Proceedings of the Fourth International Conference on Tools for Teaching Logic 9–12 June 2015

Briscoe Center for American History
circa 1961

Preface

This book constitutes the proceedings of the Fourth International Congress on Tools for Teaching Logic, TTL 2015 (http://ttl2015.irisa.fr/) held in Rennes, France, in June 2015.

The contributions were carefully reviewed and selected; each submission was reviewed by at least two members of the Programme Committee.

The congress focuses on a variety of topics including: logic teaching software, teaching deductive calculus, logic in the humanities and in computer science education, dissemination of logic courseware, methods for teaching logic at different levels of instruction, facing some difficulties concerning what to teach, visual methods for learning logic, teaching classical theorems by new methods, teaching modal logic, argumentation theory and syllogisms, logic in action, publishing and teaching at the digital age. A special session devoted the *Life and Work of Leon Henkin* [6] is also part of the present congress. Henkin was not only an excellent researcher but a social activist, having labored much of his career to boost the number of women and underrepresented minorities in the upper echelons of mathematics.

The first TTL congress took place in 2000[1]. It was the idea of an international group of logicians that in 1998 created ARACNE, an European Community ALFA (America Latina Formación Académica) network. The second congress took place in 2006 (`http://logicae.usal.es/SICTTL/`), and the third in 2011 (`http://logicae.usal.es/TICTTL/`).

The proceedings of these congresses were published in [2] and [3], and a special volume of the *Logic Journal of the IGPL* devoted to Tools for Teaching Logic was published in 2007 [4] containing selected contributions of the second congress. In 2011, selected contributions to the third congress were published in *Lecture Notes in CS* [5]. As we write this, the number of downloads of the latter are over 16 thousand.

From an educational point of view, the papers published in the current proceedings can be broadly divided into four major categories: pedagogical issues, technological tools for teaching and learning logic, logic curriculum and gender issues. Most papers can be classified as focused on technological or pedagogical issues, in this order; and after these, we can find those dealing with curricular issues. This confirms a trend already present in the previous two conferences. Furthermore, it also shows that these issues, far from having been resolved, continue to be elements of current discussion. In addition, another current educational issue, concerning the gender perspective in teaching logic is represented in this fourth congress, with two papers.

Most of us share the feeling that the teaching of an interdisciplinary field spanning logic, linguistics and computer science should be available in such a way that will facilitate further interdisciplinary research. Nevertheless, we are aware that the needs are different in those fields of study which have already been established. The overall concern is on the teaching of logic, but with special regard for addressing innovations and the systematization of educational activity. We believe the subject matters of Tools for Teaching Logic to be as diverse as the three terms in the suggested title. It is important to discuss *how* to teach, *with what* to teach, but also *what* should be taught.

Very often there is a divorce between the teaching of logic and the research in logic. Hardly any textbook gives students a good impression of the range and scope of logic today, and the same may well apply to teaching at college level. We believe that the role of logic in the shaping of the epistemology of humankind in the third millennium is crucial. Information technology is rapidly changing the world we live in. Logic is helping us to produce, distribute and process information, and also helping us to

[1] aracne.usal.es/congress/congress.html

understand how coded information can modify people's state of knowledge.

In 1995 the Association for Symbolic Logic distributed the Guidelines for Logic Education [1]. They focused on the basic logical principles of reasoning and distinguished between young children, adolescents and college students. Their recommendations cover not only a possible list of important topics to be taught at different levels, but they also give some tips to make logic courses successful. Their recommendations never go beyond the realm of classical first order logic. *Is that enough?* Some of the papers of this issue can be seen as an implementation of the ideas of the ASL committee, and some go beyond first-order logic.

The ASL Committee on Logic Education also provides software and other computer tools for teaching logic, by offering a list of logic courseware. Thus, showing how important that aspect is. In the case of computer tools for learning logic, however, the wide dispersion of systems may suggest the need for some kind of standardization, as we can find among the tools for learning mathematics. Most systems are individually produced for the use in a particular course. All this makes it difficult to share these resources. Therefore, the standardization may be one of the things to come.

Since the last Tools for Teaching Logic conference, the new actor on the block is the SIGLOG, the *ACM Special Interest Group on Logic and Computation*, which has also invested on an Education Committee. This committee is specially worried about the role of logic in Computer Science, and is now helping in the revision of the influential ACM Computer Science Curricula Recommendations. The future is to witness the effects of its present interventions.

Methodology and pedagogical issues for teaching logic are the most relevant. In the spirit of Henkin, care in how to facilitate learning has to be one of the key elements. Also, following the ASL recommendations, the introduction of basic logic courses at the university level, in a general way, is a discussion to be addressed seriously. Nowadays, universities are increasingly kidnapped by the need to provide the certificates that demand the labour market in the short term. In such situation, the scientific rigor and the return to the society should be central lines, and then, the importance of critical thinking leads to the need of maintaining and improving logic education.

References

[1] ASL Committee on Logic Education. Guidelines for Logic Education. Bull. Symb. Logic, 1(1):4-7, 1995. URL = `http://www.ucalgary.ca/aslcle/guidelines`

[2] Manzano, M. Proceedings of the First International Congress on Tools for Teaching Logic. University of Salamanca, 200. DL: S.443-2000. URL = `http://logicae.usal.es`

[3] Manzano, M. Pérez-Lancho, B. Gil, A. Proceedings of the Second International Congress on Tools for Teaching Logic. University of Salamanca, 2006. URL = `http://logicae.usal.es`

[4] Ditmarsch, H. and Manzano, M. (eds) Special Issue: Tools for Teaching Logic. Logic Journal of the IGPL. 15(4). 2007

[5] Blackburn, P. Ditmarsch, H. Manzano, M. and Soler-Toscano, E. (eds) Tools for Teaching Logic. Lecture Notes in Computer Science. 6680. 2011.

[6] Manzano, M., Sain, I. Alonso, E. (eds) The Life and Work of Leon Henkin: Essays on His Contributions. Birkauser/Springer. 2014

June 2015

M. Antonia Huertas
María Manzano
João Marcos
Sophie Pinchinat
François Schwarzentruber

Organization

Program Chairs

M. Antonia Huertas
María Manzano
João Marcos
Sophie Pinchinat
François Schwarzentruber

Program Committee

Giovanna d'Agostino
Carlos Areces
Philippe Besnard
Iliano Cervesato
Hans van Ditmarsch
Ulle Endriss
Susanna Epp
Annie Foret
María José Frápolli
Tim French
Olivier Gasquet
Patrick Girard
Hubert Marraud González
Valentin Goranko
Andreas Herzig
Colin de la Higuera
Steffen Hölldobler
Theo Janssen
Fenrong Liu
Josje Lodder
Concepción Martínez Vidal
Manuel Martins
Angelo Montanari
Angel Nepomuceno
Valeria de Paiva
Ram Ramanujam
Christian Rétoré
Giovanni Sambin
Martin Strecker
Sergio Tessaris
Helmut Veith
Audrey Yap

Additional Referees

Sarah Alice Gaggl
Christoph Wernhardt
Emmanuelle-Anna Dietz

Organization Comittee

Sophie Pinchinat
François Schwarzentruber
Élisabeth Lebret
Loïc Lesage
Agnès Cottais
M. Antonia Huertas
María Manzano
João Marcos

Invited speakers

Claude Kirschner

Publishing and Teaching at the digital age

The digital revolution strongly impacts the ways we are developing sciences as well as publishing and teaching in all academic domains. These two main activities of scientists are indeed two main pillars of the knowledge economy. They both rely also on scientific social networks allowing for the elaboration, organization, sharing, training of communities, knowledges and know-hows. This gives rise currently to the emergence and development of many initiatives like Academia.edu, Mendeley, PeerEvaluation, ResearchGate, etc.

The scientific communities contribute and benefit of these developments, making possible the development and uses of public archives like HAL or ArXiv, or of MOOC and e-learning platforms like FUN (France Université Numérique) or EdX. This allows for the development of dedicated services like overlay journals, cooperative editors or specific scientific social networks.

Using and mastering these powerful digital environments induce increasing responsibilities of scientists, teachers and learners in the context of public and private interests and competences which policies should benefit of a balanced public-private dialog.

The presentation will introduce and discuss these complementary and sometime conflicting aspects.

Nicole Schweikardt

Some reflections on teaching an introductory course on logic in computer science

I will report on experiences made while giving introductory courses on logic in computer science at Goethe-University Frankfurt and at Humboldt-University Berlin. My major goal during these courses is to show students at the very beginning of their computer science studies that logic is at the center of computer science. In my talk I will give details on the course topics, examples of "fun" exercises, and a demonstration of a "compiler" for propositional logic that was developed within my group.

Patrick Blackburn

The New Trivium

Here's a quick and dirty history of logic. First there was Aristotelian logic, and then, for a long time, not so very much else. Logic, along with grammar and rhetoric, formed part of the medieval trivium, which placed it at the heart of humanist tradition. But it was a blunt tool.

The 20th century sharpened it. Flint axe morphed into laser, and new light shone into the heart of the humanities, yielding insights about language and cognition. Ingredients for a new trivium were assembled by Richard Montague, Paul Grice, and John McCarthy.

This story is more cartoon than history, but it makes an important claim: modern logic can and should speak to the humanities. But it also poses a problem: how can this new logic be taught ? One difficulty is more or less obvious: modern logic can be technically demanding, and many students lack the mathematical background that standard introductions presuppose. Other difficulties are more blurred - which is itself part of the problem.

TTL 2015

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 1–2
Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

So: can we create a new trivium? Can we teach logic in a way that speaks to the humanities? And what is the logic we should teach?

Mordechai Ben-Ari

Logic in Computer Science Education

Computer science (CS) differs from other disciplines in that its theoretical basis is not continuous mathematics (calculus and differential equations), but discrete mathematics, in particular, logic. The talk will survey two important uses of logic in CS and describe tools for teaching them. Computers are widely used in critical systems, but programs cannot be fully tested, and in the case of concurrent and distributed programs, cannot be tested at all. Program verification uses logic to demonstrate the correctness of programs. The first approach was deductive: inductive assertions (Dijkstra and Floyd), axiomatic systems (Hoare) and temporal logic (Pnueli). The difficulty of constructing deductive proofs led to the development of model checking. Model checking is based upon an exploration of the state space of a program, in parallel with an exploration of correctness specifications based upon assertions and translations of temporal logic formulas into Büchi automata. While model checking is theoretically limited, in practice, it has proved to be extremely successful. I will present the Erigone model checker that I developed for educational purposes. SAT—the problem of deciding if a formula in CNF is satisfiable – is fundamental in CS, because it was the first problem to be proved NP-complete, and therefore, almost certainly has no efficient algorithm. Nevertheless, advances in algorithms and implementations have turned SAT-solvers based upon the simple DPLL algorithm into highly efficient practical tools. Algorithms include stochastic search and clause learning to trim the search space. I will present the LearnSAT system that I developed for educational purposes.

Johan van Benthem

LOGIC IN ACTION, a grand old topic with a new twist

What are the basic logical notions and skills that beginning students should learn, and that might stay with them as a useful cultural travel kit for their lives, even when they do not become professional logicians? "Logic in Action" tries to convey the idea that logic is about reasoning, but also much more: including information and action, both by individuals and in multi-agent settings, studied by both semantic and syntactic tools. Viewed in this way, logic sits at a crossroads of disciplines where exciting new traffic comes by every day.

I will explain the ideas behind the course, which combines predicate logic with various modal logics, its current mani- festations (and dialects) in Amsterdam, Beijing and the Bay Area, and its possible future as an EdX pilot course.
Reference: `http://www.logicinaction.org/`

Gilles Dowek

Rules and derivations in an elementary logic course

The attempt to explicitly define the notions of inference rule and derivation in an elementary logic course lead us to define two kinds of derivations - labelled by objects and by rule names. Taking some time to clearly define these notions pays off when introducing not only the notion of proof, but also that of computable function, of automaton, and the Curry-Howard correspondence.

Table of Contents

Twist your logic with TouIST

Skander Ben Slimane[1], Alexis Comte[1], Olivier Gasquet[1], Abdelwahab Heba[1], Olivier Lezaud[1], Frederic Maris[1], and Mael Valais[1]

1 **University Paul Sabatier**
Toulouse, France
{gasquet,maris}@irit.fr

Abstract

SAT provers are powerful tools for solving real-sized logic problems, but using them requires solid programming knowledge and may be seen w.r.t. logic like assembly language w.r.t. programming. Something like a high level language was missing to ease various users to take benefit of these tools. TouIST aims at filling this gap. It is devoted to propositional logic and its main features are 1) to offer a high-level logic langage for expressing succintly complex formulas (e.g. formulas describing Sudoku rules, planification problems,...) and 2) to find models to these formulas by using the adequate powerful prover, which the user has no need to know about. It consists in a friendly interface that offers several syntactic facilities and which is connected with some sufficiently powerful provers allowing to automatically solve big instances of difficult problems (such as time-tables or Sudokus). It can interact with various provers: pure SAT solver but also SMT provers (SAT modulo theories - like linear theory of reals, etc) and thus may also be used by beginners for experiencing with pure propositional problems up to graduate students or even researchers for solving planification problems involving big sets of fluents and numerical constraints on them.

1998 ACM Subject Classification K.3.2 Computer and Information Science Education ; I.2.8 Problem Solving, Control Methods, and Search

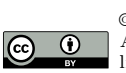
Keywords and phrases Interface High-level logic formalization SAT-prover

1 The history

O. Gasquet and F. Maris teach at University Paul Sabatier in Toulouse, France. They teach logic at different levels starting from introductory courses of propositional logic up to advanced topics for graduate students, like modal logic or logic-based planning. S. Ben Slimane, A. Comte, A. Heba, O. Lezaud and M. Valais are graduate students of the same university. They have been implementing TouIST during three months of their MSc.

Motivation of students

At the beginning of undergraduate studies, we (teachers) found that students' motivation may be increased by showing them that logic is useful and powerful for computer scientists and that computer science does not only consist in hacking C-code or JAVA. Classically, logic is motivated by abstract examples or, at the best, by toy examples. At some time, we thought that it would be preferable to show and not only tell them that with little knowledge, logic can be used to solve difficult problems whose size prevents humans from solving them by hand easily or would require rather complex programming in C or any other programming language.



4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 1–8

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

SATouLouse's genesis

In ICTTL'2011, we presented SATouLouse [2], devoted to propositional logic whose main features were 1) to offer a high-level logic language for expressing succinctly complex formulas and 2) to find models of these formulas by using a powerful SAT prover. But SATouLouse had several drawbacks to be corrected.
Of course, there are loads of logic tools like provers, proof assistants, truth table editors,... on the Internet, even PROLOG could have been used, but none fits our requirements which are:

- the tool must be very easy to install and to use, with no complex syntax;
- the prover can be used as a black box without knowing how it works;
- no normal forming, ordering on clauses, or PROLOG cut must be needed;
- only little knowledge in logic should be necessary.

As we could not find an existing tool fulfilling these requirements, in 2010 we started to implement ours, and we came to the idea of just developing an interface that allows to very comfortably use a powerful SAT-prover (namely SAT4J [1]): this tool had been called SATouLouse and is described in [2]. With this tool, students could experiment by themselves that a logical language is not only descriptive but may lead to computations that solve real-life problems. In particular, with SATouLouse, they could solve Sudokus quite easily, as well as many other combinatorial problems such as time-table, map coloring, electronic circuits design,....

Here are the main facilities that SATouLouse offered:

- Input formulas need not to be in clausal form and arbitrary connectives may be used, normal forming is done dynamically during keyboarding of the user;
- Big conjunctions and disjunctions facilities are offered like in:

$$\bigwedge_{i\in\{1..9\}} \bigvee_{j\in\{1..9\}} \bigwedge_{n\in\{1..9\}} \bigwedge_{m\in\{1..9\},m\neq n} (p_{i,j,n} \rightarrow \neg p_{i,j,m})$$

- Running the solver only consists in clicking a button;
- The tool displays a model in the syntax of the input formula.

Then it is possible to show the power of propositional logic to students that have been trained a bunch of hours to formalize sentences in logic and have acquired basic notions of validity and satisfiability to automatically solve some Sudokus.

Practical work with SATouLouse

But this is not the whole story, since the same SAT-solver may be used for solving many other combinatorial problems as easily as they just did for Sudokus: they just have to formalize the constraints. Our students are asked to do so for: time-table, map coloring,... SATouLouse has been used during three years now by about 400 students with great satisfaction. Particularly, students used it to perform long-term homeworks in the spirit of programming projects: we give them a logical problem to solve (too big to be solved by hand), they must formalize it and then use this formalization to solve the problem. For example, a problem of storage of chemicals that must be stored in same/contiguous/non-contiguous rooms according to their degree of compatibility. Students must solve a case involving a lot of chemicals.

SATouLouse's limitations and TouIST's genesis

But during these years, we noticed some painful limitations of SATouLouse: many bugs, flaws in the interface, lack of modularity (if one wishes to change the SAT prover used), ambiguity and limitations of its language, etc.

For example, problems involving pigeon-holes principle like the rules of the Takuzu game[1] which requires to count 0's and 1's could not be easily formalized: facilities to express something like "exactly 5 among 10 propositions are true" were missing.

SATouLouse do not offer the possibility to browse all the models provided by the prover, it only returns one.

Lessons learned from two years using SATouLouse are that many of our CS students clearly become aware that logic has real applications w.r.t. problem solving, and many of them gained ability in formalizing problems. But remaining flaws of SATouLouse made debugging really hard because only one model is displayed and because of the raw way the models is displayed, together with the poor editing capabilities it has. Moreover only pure combinatorial problems could be handled which heavily limitates the wide range pretention of SATouLouse w.r.t. real world problems.

Another drawback of SATouLouse not specifically linked to logic teaching, was its inability to be used from the command line: researchers or engineers who wish to use it intensively would find it tedious to type input problems. Last, extension to richer theories is also something that may interest researchers, engineers or graduate students, since SATouLouse is definitely not suited for satisfiability modulo theories or for solving planification problems though the same architecture of the software could be used by just changing the solver used.

A few months ago, we started to go for a whole new software which would fulfill all these demands. It would be called TouIST which stands for TOUlouse Integrated Satisfiability Tool and should be pronounced "twist".

TouIST is of course publicly available for download from the following site

`https://github.com/olzd/touist/releases`

To sum it up, here are the features TouIST offers that SATouLouse does not:

- definition of domain sets: $\bigwedge_{i \in A}$ vs. $\bigwedge_{i \in \{Paris, London, Roma, Madrid\}}$
- multiple binding of indexes: $\bigwedge_{i \in A, j \in B}$ vs. $\bigwedge_{i \in \dots} \bigwedge_{j \in \dots}$
- rich computations on indexes as well as on domain sets $\bigwedge_{i \in (A \cup (B \cap C))}$
- built-in pigeon-holes primitives: "atLeast" (resp. "atMost", "exact") *so many* values are true among *these values*
- predicates also may be variables ranging over domain sets: $\bigwedge_{X \in \{A,B\}, i \in \{1,2\}} X(i)$ vs. $\bigwedge_{i \in \{1,2\}} (A(i) \wedge B(i))$
- specialized literals targeting constraints between integer or real numbers
- easy browsing of models successively computed by the solvers
- regular expressions allowing filtration of literals under interest
- possibility to use the software on command line and/or batch
- many editing facilities and improvements

2 Quick survey of TouIST

TouIST is made of three modules, but the standard user will only see one of them: the interface. In the sequel we mainly insist on the latter rather than on the translator and the solver. The global architecture looks as pictured in figure 1:

With TouIST one accesses a powerful and friendly editor for editing complex logical formulas and various constraints like:

[1] Also known as Binero. `http://fr.wikipedia.org/wiki/Takuzu`

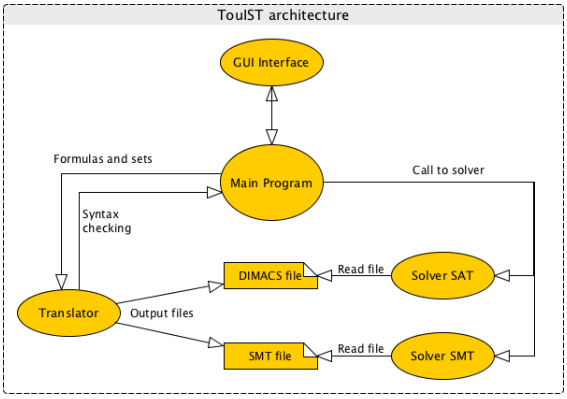


Figure 1 TouIST architecture

$$\bigwedge_{i\in\{1..9\}} (P_i \longrightarrow Q_{i+1}),$$

which comfortably abbreviates $(P_1 \longrightarrow Q_2) \wedge (P_2 \longrightarrow Q_3) \wedge \ldots \wedge (P_9 \longrightarrow Q_{10})$.

Once it has been given to the interface, a set of formulas may be checked for satisfiability: the interface would send it to the provers which would send back a satisfying model, displayed as shows figure 2 if such models exist. Then through the interface, the user can for example ask for other models (button "Next" of the interface).

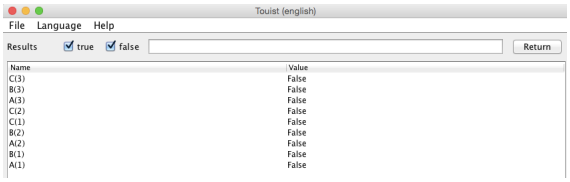


Figure 2 Model display

Models returned by the prover are "total" ones: each variable appearing in the formulas sent to the prover is assigned a value. The user may select only True propositions or only False ones. She can also select subsets of the variables under interest by typing a regular expression filtering them.

3 Details of what can be done with TouIST

3.1 Domain sets

With time, we noticed that we often need to write things like

$$\bigwedge_{i\in\{1..9\}} \bigwedge_{j\in\{1..9\}} \bigwedge_{m\in\{A,B,C,D,E,F,G,H,I\}} P_{i,j,m} \longrightarrow \bigwedge_{n\in\{A,B,C,D,E,F,G,H,I\}|m\neq n} \neg P_{i,j,n}$$

If one read $P_{i,j,m}$ as "there is a letter m in cell (i,j)" of some 9×9 grid, the above formula expresses that there is *at most* one letter among 'A' ... 'I' in each cell.

These sets $\{1..9\}$ and $\{A,B,C,D,E,F,G,H,I\}$ are *domain sets*, with TouIST the user may define as many domain sets she wants, e.g.:

```
N=(1..9)   L=(A,B,C,D,E,F,G,H,I)
```

and then write the above formula as $\bigwedge_{i\in N}\bigwedge_{j\in N}\bigwedge_{m\in L} P_{i,j,m} \longrightarrow \bigwedge_{n\in L|m\neq n} \neg P_{i,j,n}$
Moreover, usual operations on sets ($\cup$, $\cap$, $\setminus$, ...) can be used to define other sets.

3.2 Propositional formulas

The formulae of TouIST are based on propositional variables (that can have indices) and usual logical operators ($\wedge$, $\vee$, $\longrightarrow$, $\neg$, $\leftrightarrow$). Thus one can type usual simple formulas like $Rain \longrightarrow Clouds$. But in addition, we provide high-level logical operators that allow to express complex statements in a very compact form.

Generalized conjunctions and disjunctions

They allow to express conjunctions and disjunctions over formulas containing parameters that vary, e.g.

- $\bigwedge_{i\in N} P_i$, where N is the domain set defined above. It represents $P_1 \wedge P_2 \wedge \ldots \wedge P_9$.
- $\bigvee_{i\in E} P_i$.

Of course, these operators may be nested, as in $\bigwedge_{i\in N}\bigwedge_{j\in N}\bigvee_{m\in L} P_{i,j,m}$ stating that in each cell there is at least one letter.

Pigeon-hole statements

They were one of the "left-to-the-future" topic of [2]. These less classical logical operators are available in TouIST: they allow to drastically lower the size of some formulas, they are: $\leqslant$, $\geqslant$ and $<>$.
The following examples will describe their meanings:

- $\leqslant^2_{i\in N} P_i$ represents "for at most two values of $i \in N$ $P(i)$ is true;
- $\geqslant^2_{i\in N} P_i$ represents "for at least two values of $i \in N$ $P(i)$ is true;
- $<>^2_{i\in N} P_i$ represents "for exactly two values of $i \in N$ $P(i)$ is true;

Generalized disjunction is in fact a special case of those: at least one is true, conjuction too: at most 0 are false, and exclusive or may be viewed as: exactly one among two is true.

Let us recall that with basic logical operators and with N containing 9 elements, $\leqslant^3_{i\in N} P_i$ would necessitate a formula containing 84 propositions P_i since it amounts to choosing 3 among 9 which yields $\binom{9}{3}$ possibilities, and neither $\bigwedge$ and $\bigvee$ would help a lot.

Constraints and calculus on indexes

Often we need to add constraints on indexes, for example:

$$\bigwedge_{i\in E}\bigwedge_{j\in E|i\neq j} P_{i,j}$$

which means that $P_{i,j}$ is true whenever $i \neq j$.

This was the only constraint available in SATouLouse, now in TouIST the range of possibility has been widely enriched. Constraints may include usual comparaison operators like $<$, $>$, $\leq$, $\geq$, $\neq$, $=$ and these comparisons may not only apply to indices but to any arithmetic expressions involving indexes and $+$, $-$, $*$, $/$, $\mod$, $\sqrt{\ }$. Expressing a sentence like "each cell (i, j) contains a number which is not equal to $i + j$" will give:

$$\bigwedge_{i\in N}\bigwedge_{j\in N}\bigvee_{k\in N|k\neq i+j} P_{i,j,k}$$

Of course, *all these sentences* may be expressed with usual plain logical operators, but this would be an aweful work to do. Nevertheless, students must know what is behind the scene, and that such a compact formula abbreviates something long and dull like:

$$P_{1,1,1} \vee P_{1,1,3} \vee P_{1,1,4} \ldots P_{1,2,1} \vee P_{1,1,2} \vee P_{1,2,4} \vee \ldots$$

3.3 Technical aspects

Input language vs display language

Formulas as seen above are written in the *display* language (LaTeX-style), but all those symbols are not available on keyboards, thus for writing formula and domain sets, the user will use the input language. For example, the above formula together with the associated set N will be typed as (variables are prefixed with $):

```
bigand $i in $N , $j in $N
  bigor $k in $N when $k < $i+$j :
    P($i,$j,$k))
  end
end
```

But TouIST displays it in LaTeX-style as seen in the right panel shown in figure 3. The definition of the set N is done in the Sets tab.

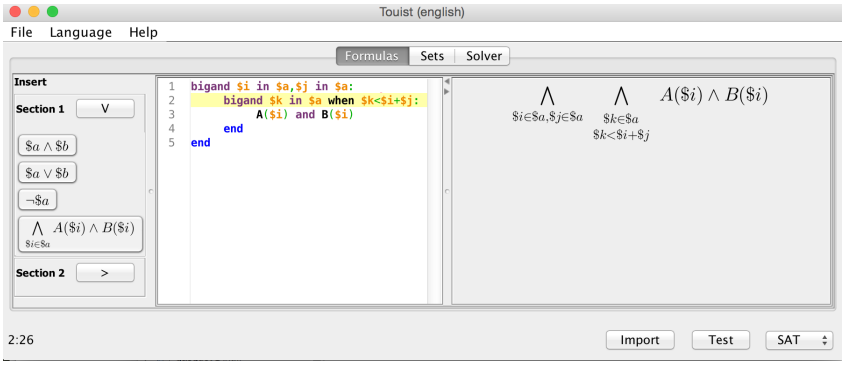


Figure 3 LaTeX style display

Also, formulas may either be hand-typed in the editor window, or introduced in a sort of syntax-directed editor, by progressively refining the syntax tree, or else they can be imported from some external file.

4 Advanced topics for graduate students

In what follows, we very briefly present some advanced features of TouIST. They may rather interest researchers, engineers, graduate students and their teachers. They concern SMT (SAT modulo theories), Planning as SAT and their combination Planning as SMT.

4.1 SMT: SAT modulo theories

Some combinatorial problems require nevertheless to deal with some calculus over natural or real numbers. This can be done using only propositional logic (e.g. $2+3=5$ may be encoded

by $ADD_{2,3,5}$), but it is very uncomfortable as soon as there are more than a few additions to be made. Do not even mention products or more complex operations. The idea behind SMT genesis has been to combine SAT solvers with arithmetic solver in order to improve the treatment made to the arithmetic part of reasoning. In many cases, it will not only improve the efficiency of the prover, but will also allow to express arithmetic constraints of problems in a drastically more compact way.

Think of the Kamaji game[2] where the player must group adjacent numbers in a grid so that their sum is equal to some fixed number. Solving the game essentially requires logical reasoning but still needs a few arithmetic (addition).

Then if $x_{i,j}$ for each cell (i, j) is an integer and $G(i, j, i, k)$ represents the fact that cells (i, j) to (i, k) of line i form a group, the sum constraint may be expressed as: $\Sigma_{m\in E} x_{i,m} = N$ where N is the fixed number and E is $\{j, j + 1, \ldots, k\}$. Pure propositional logic is definitely unsuited for such sentences!

4.2 TouIST for classical planning as SAT

In Artificial Intelligence, *planning* is a cognitive process to automatically generate, through a formal procedure, an articulated result in the form of an integrated decision-making system called *plan*. The plan is generally in the form of an organized collection of *actions* and it must allow the universe to evolve from the *initial state* to a satisfactory state, the *goal*. Propositional planning as SAT has been introduced by Kautz and Selman in [4].

One important difference of TouIST compared with SATouLouse is its ability to take into account both logic formulas and domain sets. For example, if one wants to solve a particular planning problem, SATouLouse is easy to use for describing the problem and solving it via a SAT solver. But in order to solve several generic planning problems, we can take advantage from the flexibility of TouIST which will allow the user to describe a generic solving method with rules encoded as formulas and to use domains sets to describe each particular planning problem. Numerous encoding rules for planning problem resolution have already been proposed [4, 5, 7]. As an example of such a rule we give below an encoding of frame-axioms. If a fact is false at step i-1 of a solution plan and becomes true at step i, then the disjunction of actions that can establish the fact at step i of the plan is true. That is, at least one of the actions that can establishes the fact should have been applied.

$$\bigwedge_{i\in\{1..length\}} \bigwedge_{f\in Facts} \left((\neg f(i-1) \wedge f(i)) \Rightarrow \bigvee_{a\in Actions/f\in Effects(a)} a(i) \right)$$

4.3 TouIST for temporal planning as SAT (modulo unquantified rational difference logic)

Moreover, in addition to SAT, our new platform TouIST is able to handle theories like difference logic or linear arithmetic on integer or real numbers, and call a SMT solver to find a solution. To be solved, real world temporal planning problems require to represent continuous time, and so, the use of real numbers in logic encodings. TouIST can also be used to solve such problems involving durative actions, exogenous events and temporally extended goals, for example with encoding rules proposed in [6]. We give below an encoding of temporal

[2] http://fr.wikipedia.org/wiki/Kamaji

mutual exclusion of actions. If two actions respectively producing a proposition p and its negation are active in the plan, then the time interval $[\tau(a \mid\!\rightarrow p), \tau(a \rightarrow\!\mid p)]$ corresponding to the activation of p, and the time interval $[\tau(b \mid\!\rightarrow \neg p), \tau(b \rightarrow\!\mid \neg p)]$ corresponding to the activation of $\neg p$ are disjoint.

$$\bigwedge_{a \in Actions} \bigwedge_{b \in Actions} \bigwedge_{f \in Facts \mid f \in Effects(a) \wedge \neg f \in Effects(b)}$$

$$[(a \wedge b) \Rightarrow [(\tau(b \rightarrow\!\mid \neg f) < \tau(a \mid\!\rightarrow f)) \vee (\tau(a \rightarrow\!\mid f) < \tau(b \mid\!\rightarrow \neg f))]]$$

5 Conclusion

As far as we are aware, there is no other tool targeted at the same large audience, neither at the same wide class of problems, neither with the same comfort. Most existing pedagogical tools (either implementation of truth-tables or semantic tableaux) that could do the job of searching a model cannot efficiently handle big problems, and real tools able to deal with them are definitely not designed to be used by beginners in logic, and not even by most graduate students. Advanced tools designed for graduate topics, like Mozart [8] or Alloy [3] have a steep learning curve that may dissuade beginners and non-specialist users.

We believe TouIST will be useful for beginners in logic as well as for advanced users thanks to its large scope of applications and to its ease of use.

References

1 Daniel Le Berre and Anne Parrain. The sat4j library, release 2.2. *JSAT*, 7(2-3):59–6, 2010.

2 Olivier Gasquet, François Schwarzentruber, and Martin Strecker. Satoulouse: the computational power of propositional logic shown to beginners. In P. Blackburn, H. van Ditmarsch, M. Manzano, and F. Soler-Tosca, editors, *Third International Congress on Tools for Teaching Logic (ICTTL'2011)*, volume 6680 of *Lecture Notes in Computer Science*, pages 77–84. Springer, 2011.

3 Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

4 Henry Kautz and Bart Selman. Planning as satisfiability. In *IN ECAI-92*, pages 359–363. Wiley, 1992.

5 Amol Dattatraya Mali and Subbarao Kambhampati. On the utility of plan-space (causal) encodings. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, 1999*, pages 557–563, 1999.

6 Frédéric Maris and Pierre Régnier. Tlp-gp: New results on temporally-expressive planning benchmarks. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, volume 1, pages 507–514. IEEE Computer Society, 2008.

7 Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Planning as satisfiability: Parallel plans and algorithms for plan search. *Artif. Intell.*, 170(12):1031–1080, 2006.

8 Peter Van Roy, editor. *Multiparadigm Programming in Mozart/Oz, Second International Conference, MOZ 2004, Charleroi, Belgium, October 7-8, 2004, Revised Selected and Invited Papers*, volume 3389 of *Lecture Notes in Computer Science*. Springer, 2005.

dasasap, an App for Syllogisms

J. Martín Castro-Manzano[1], Verónica Reyes-Meza[2], and Jorge Medina-Delgadillo[3]

1 **Faculty of Philosophy and Humanities, UPAEP**
`josemartin.castro@upaep.mx`
2 **Faculty of Psychology, UPAEP**
`veronica.reyes@upaep.mx`
3 **Faculty of Philosophy and Humanities, UPAEP**
`jorge.medina@upaep.mx`

Abstract

The main goal of this contribution is to introduce a cross-platform application to learn-teach syllogistic. We call this application *dasasap* for *develop all syllogisms as soon as possible*. To introduce this application we show the logical foundations for the game with a system we call $\mathcal{L}_{\Box}$, and its interface developed with LiveCode.

1998 ACM Subject Classification F.4.1 Mathematical Logic, K.8 Personal Computing

Keywords and phrases Application, Software, Syllogistic, LiveCode

1 Introduction

Syllogistic has a remarkable impact in our education and culture: most undergraduate logic courses, books and manuals around the world cover or include a fragment of syllogistic not just to teach logic, but to provide *formæ mentis*. Therefore, whether studying syllogistic is relevant or not is beyond doubt; the problem, if any, has to do with the different treatments used to learn-teach syllogistic.

The main goal of this contribution is to introduce a cross-platform application to learn-teach syllogistic. We call this application *dasasap* for *develop all syllogisms as soon as possible*. To introduce this application we show the logical foundations for the game with a system we call $\mathcal{L}_{\Box}$ (Section 2), and its interface developed with LiveCode (Section 3). We close this work with comments about future work (Section 4).

2 Logical foundations

2.1 Syllogisms

A *categorical proposition* is a proposition with a defined quantity (universal or particular), a subject S, a quality (affirmative or negative), and a predicate P. The adequate combinations of these elements produce four categorical propositions, each one with a name:

- Proposition A states *All* S *is* P.
- Proposition E states *No* S *is* P.
- Proposition I states *Some* S *are* P.
- Proposition O states *Some* S *are not* P.

A *categorical syllogism* is a sequence of three categorical propositions ordered in such a way that the two first propositions are called *premises* (major and minor) and the last one is called *conclusion*. Within the premises there is a term M, known as *middle term*, that works as a *link* that gathers the remaining terms, namely, S and P:

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 9–16


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

▶ Example 1.

1. Major premise: *All* M *is* P.
2. Minor premise: *All* S *is* M.
3. Conclusion: *All* S *is* P.

The previous example can be represented as a string of characters using the terms and proposition names stated previously in an infix notation: MAPSAM ∴ SAP, where ∴ indicates that the premises are to the left and the conclusion to the right. According to the position of the middle term M we can build four *figures* of the syllogism that encode all the valid and only the valid syllogisms. These have been named using mnemonics [1] (Table 1).

Table 1 Valid categorical syllogisms

Figure 1	Figure 2	Figure 3	Figure 4
Barbara	Cesare	Disamis	Camenes
MAPSAM ∴ SAP	PEMSAM ∴ SEP	MIPMAS ∴ SIP	PAMMES ∴ SEP
Celarent	Camestres	Datisi	Dimaris
MEPSAM ∴ SEP	PAMSEM ∴ SEP	MAPMIS ∴ SIP	PIMMAS ∴ SIP
Darii	Festino	Bocardo	Fresison
MAPSIM ∴ SIP	PEMSIM ∴ SOP	MOPMAS ∴ SOP	PEMMIS ∴ SOP
Ferio	Baroco	Ferison	
MEPSIM ∴ SOP	PAMSOM ∴ SOP	MEPMIS ∴ SOP	

2.2 Jigsaw puzzles

A jigsaw puzzle is a tiling array composed by a finite set of tessellating pieces that require assembly by way of the *interlocking* of knobs and sockets (Figure 1). When finished, a jigsaw puzzle mechanically produces a complete picture by the adequate interlocking of all the pieces; when the pieces cannot be interlocked, the jigsaw puzzle cannot be completed, and thus the picture cannot be mechanically produced.

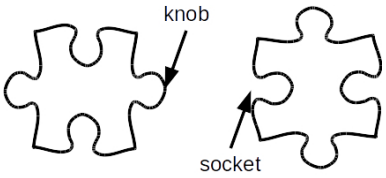


Figure 1 Pieces of a typical jigsaw puzzle

These types puzzles date back as far as Archimedes [2], so the historical paths of logic and jigsaw puzzles have a long tradition, although the typical pictorial jigsaw puzzles we are familiar with have their roots in Europe when John Spilsbury's first dissected maps in the 1760s primarily for educational purposes [3]. We suggest that syllogisms can be developed as jigsaw puzzles that require assembly by way of *interlocking*.

2.3 The system $\mathcal{L}_{\square}$

$\mathcal{L}_{\square}$ is a system that uses adequate combinations of polygons as jigsaw puzzles in order to decide the (in)validity of categorical syllogisms. Its vocabulary is defined by two elementary diagrams (i.e., pieces), socket and knobs (Figure 2).

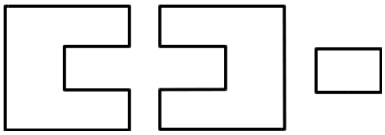


Figure 2 Vocabulary for $\mathcal{L}_{\square}$

Syntax is given by a single rule: if a diagram belongs to the vocabulary of $\mathcal{L}_{\square}$, the diagrams depicted in Figure 3, and only those, are well formed diagrams.

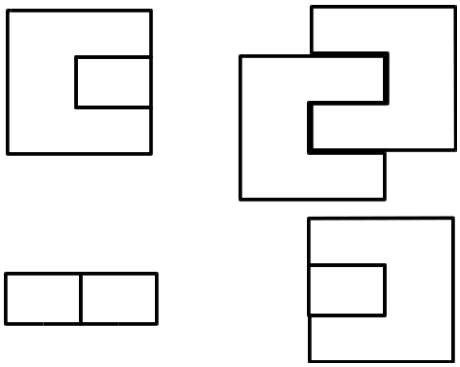


Figure 3 Well formed diagrams for $\mathcal{L}_{\square}$

Semantics is given by Figure 4.

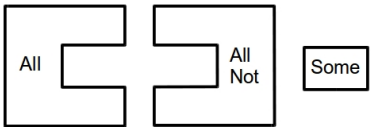


Figure 4 Semantics for $\mathcal{L}_{\square}$

With this system we can represent the categorical propositions as in Figure 5.

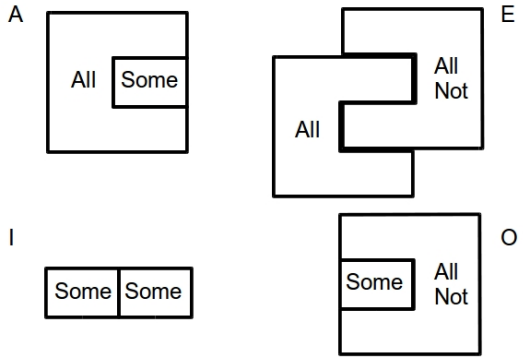


Figure 5 Categorical propositions in $\mathcal{L}_{\Box}$

We can arrange this system in a Square of Opposition (Figure 6). It should be clear that the only rules preserved in this Square are the rules for *contradiction* between A (E) and O (I); the rules for *contraries*, *subalterns*, and *subcontraries* do not work. Thus, this system's Square behaves as a Modern Square of Opposition rather than a Traditional Square. Also, *conversion*, *contraposition*, and *obversion* are valid transformations in this Square.

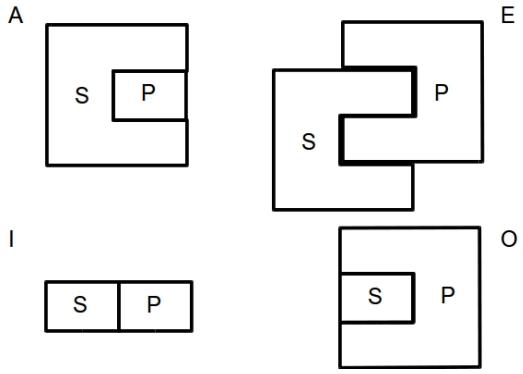


Figure 6 Square of Opposition in $\mathcal{L}_{\Box}$

2.4 Decision Algorithm in $\mathcal{L}_{\Box}$

Using a single term, in this case the middle term, we can represent the Identity Principle (IP) in this system:

- Proposition A states *All* M *is* M.
- Proposition E states *No* M *is* M.
- Proposition I states *Some* M *are* M.
- Proposition O states *Some* M *are not* M.

Figure 7 shows the four categorical propositions using the middle term M. We can observe that from these four cases only A encodes a logical truth, that is, a true statement for any structure, provided it is a weak version of the IP as in $\forall x(\mathsf{M}x \Rightarrow \exists y \mathsf{M}y)$.

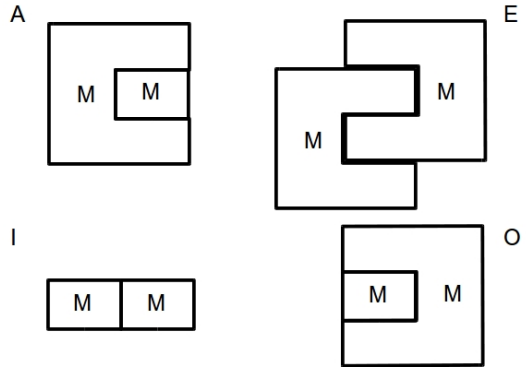


Figure 7 Identity Principle in $\mathcal{L}_{\square}$

Using the IP we suggest a decision algorithm for syllogistic in $\mathcal{L}_{\square}$ with complexity of $O(N)$ (Algorithm 1).

Algorithm 1: Decision algorithm in $\mathcal{L}_{\square}$

```
Input: syllogism σ
Output: decision δ
for σ do
    if middle terms of σ == IP then
        δ ← valid;
    else
        δ ← invalid;
    end
end
```

Hence, this algorithm not only is time efficient but sound and complete:

▶ Proposition 1. (Soundness) If $\mathcal{A}(\sigma_{i,j}) = valid$ then $\sigma_{i,j}$ is valid.

Proof. Let us denote the application of Algorithm 1 to a given syllogism $\sigma_{i,j}$ from figure $i \in \{1,2,3,4\}$ and row $j \in \{1,2,3,4\}$ in Table 1 by $\mathcal{A}(\sigma_{i,j})$ (thus, for instance, the application of Algorithm 1 to a *Dimaris* syllogism is $\mathcal{A}(\sigma_{4,2})$). We prove soundness by cases. Since there are four figures, we need to cover each valid syllogism from each figure. Here we only show the first case that covers the syllogisms of figure 1 (*Barbara*, *Celarent*, *Darii*, and *Ferio*) in Section 2.5. The remaining syllogisms can be proven similarly. Thus, we have that for every $\sigma_{i,j}$, when $\mathcal{A}(\sigma_{i,j}) = valid$, $\sigma_{i,j}$ is valid. ■

▶ Proposition 2. (Completeness) If $\sigma_{i,j}$ is valid then $\mathcal{A}(\sigma_{i,j}) = valid$.

Proof. We prove this by contradiction. Suppose that for all i, j, the syllogism $\sigma_{i,j}$ is valid but for some valid syllogism $\sigma_{k,j}$, $\mathcal{A}(\sigma_{k,j}) = invalid$. Now, we know $\sigma_{1,j}$ is valid and if we apply $\mathcal{A}(\sigma_{1,j})$ we obtain $\mathcal{A}(\sigma_{1,j}) = valid$, as we can see in Section 2.5. So, since all valid syllogisms $\sigma_{n>1,j}$ can be reduced to the valid syllogisms of figure 1 by the rules of equivalence (conversion, contraposition, and obversion), it follows that $\mathcal{A}(\sigma_{n>1,j}) = valid$, and thus, for all valid syllogisms k, $\mathcal{A}(\sigma_{k,j}) = valid$, which contradicts our assumption. ■

2.5 Example

As an example of how $\mathcal{L}_{\square}$ works we prove the four valid syllogisms of figure 1 (Figure 8), but it should be clear that the other syllogisms can be proven similarly.

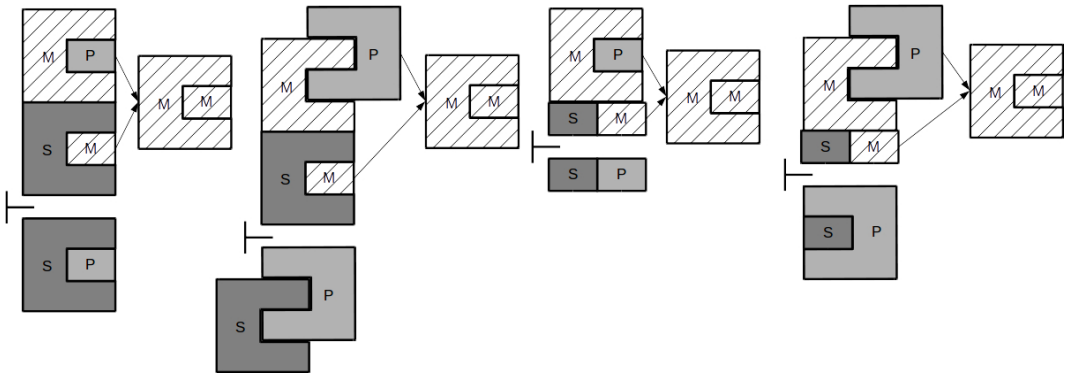


Figure 8 *Barbara*, *Celarent*, *Darii*, *Ferio*. In order to represent and build a categorical syllogism in $\mathcal{L}_{\square}$ we just stack up two well formed diagrams that represent the premises. When the middle terms *interlock* each other forming the IP in a jigsaw puzzle style (which is a step denoted by the arrows), the inference is valid, thus allowing the terms S and P interlock in the third diagram, i.e., the conclusion, which is below the turnstile.

3 Interface

Our application, *dasasap*, is motivated by the impact of syllogistic in our education and culture, is inspired upon $\mathcal{L}_{\square}$, and is implemented with LiveCode, which is an event-oriented programming language that allows the development of cross-platform applications.

At this moment *dasasap* comes with two modes: *arcade* and *learning*. In the arcade mode the user tries to develop valid syllogisms as quick as possible thinking in a jigsaw puzzle style. The objective of this mode is to fasten the recognition of valid syllogisms. The learning mode is a tutorial mode that helps users understand how to develop syllogisms in a jigsaw puzzle style assuming the ideas of $\mathcal{L}_{\square}$ and thus showing the basic concepts of logic (propositions, truth values, form-content distinction, and so on).

In Figure 9 we can see the game's home screen running in Ubuntu 12.04. We also depict fragments of the learning and arcade modes (Figures 10 and 11).

Figure 9 *dasapap*'s home snapshot

The learning mode includes four options. In the option *What's logic about?* the application explains the basic concepts of logic in order to provide an operative definition of logic: proposition, argument, form-content distinction, and truth-valid distinction are some of the concepts included. The option *So you think you are logical?* leads to a mini-game

that produces a ranking: the user is given random syllogisms and the user must decide whether they are valid or invalid. The options *What's a syllogism* and *What's dasasap?* are self-explanatory.

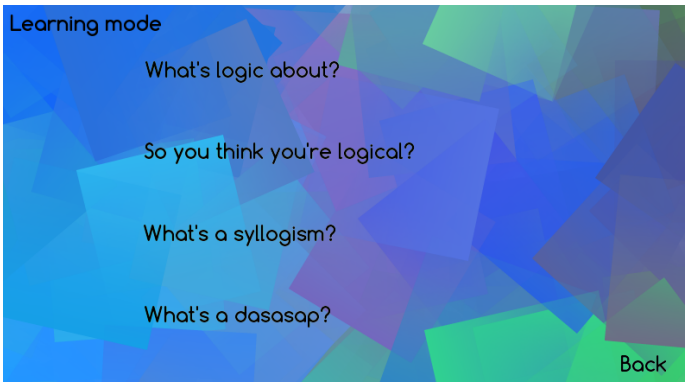


Figure 10 Learning mode snapshot

The arcade mode keeps track of the ranking of the user and allows the construction of syllogisms by measuring time efficiency and accuracy in the development of syllogisms.

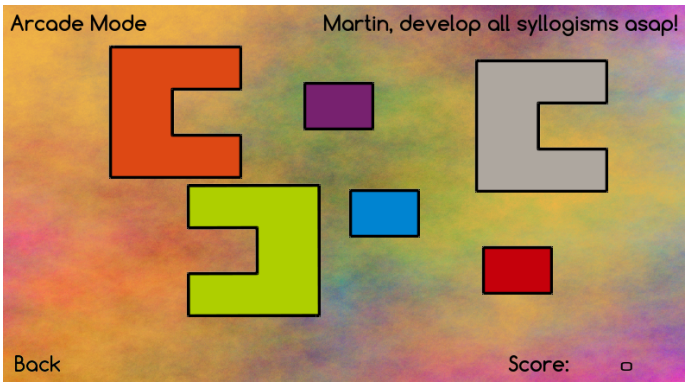


Figure 11 Arcade mode snapshot

4 Conclusion

We have introduced an application we call *dasasap* for *develop all syllogisms as soon as possible*. We showed its logical foundations with a system we call $\mathcal{L}_{\square}$, which is a sound and complete system that uses adequate combinations of polygons as jigsaw puzzles in order to decide the (in)validity of categorical syllogisms. We have implemented the mechanics of this system with LiveCode in order to develop a cross-platform application to aid the teaching-learning process of basic logic.

Since this application is a small fragment of a larger project we are currently developing, at the moment we are refining the implementation and the overall design. We are also adding a reward system and different match-three puzzle styles.

Acknowledgements The authors would like to thank the reviewers for their precise corrections and useful comments. Financial support given by UPAEP Grant 30108-1008.

References

1. Peter of Spain, *Tractatus called afterwards Summule logicales*, first critical edition from the manuscripts, with an Introduction by L.M. de Rijk. Assen, van Gorcum & Co., 1972
2. Slocum, J., Botermans, J., *Puzzles Old and New*. Seattle University of Washington Press, 1986
3. Williams, A.D., Shortz, W., *The jigsaw puzzle: piecing together a history*, Berkley Books, 2004

Gender Relations in the XIth Mexican Logic Olympiad

José Martín Castro-Manzano[1], Verónica Reyes-Meza[2], César López-Pérez[3], and Karen González-Fernández[4]

1 **Faculty of Philosophy and Humanities, UPAEP**
`josemartin.castro@upaep.mx`
2 **Faculty of Psychology, UPAEP**
`veronica.reyes@upaep.mx`
4 **Chair of the Logic Olympiad, AML**
`rudoytecnico@gmail.com`
4 **Faculty of Philosophy, UP**
`karengf@gmail.com`

Abstract

Science Olympiads are academical competitions with social impact due to the fact that they allow the detection of talent and promote science and critical thinking. In this work we portray the Mexican Logic Olympiad and we describe the proportion of women and men within its XIth edition with special focus on the issue of women's underrepresentation.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Science Olympiad, Competition, Classical Logic, Informal Logic, Problem Solving

1 Introduction

Science Olympiads are academical competitions with social impact because they allow the detection of talent, due to selection criteria; and increase interest in science, both in students and professors, for they promote the further study of science and critical thinking. The relevance of Science Olympiads, thus, is not only associated with higher order education, but with fairness and social justice.

In Mexico the Science Olympiads cover a wide range of disciplines: mathematics,[1] chemistry,[2] physics,[3] informatics,[4] history[5] and philosophy[6] Olympiads are well known; however, in this divergent context where these sciences seem to be disjoint due to the lack of a common object of study, a unique methodology, or similar applications, there is an Olympiad that looks for unity within diversity: the Mexican Logic Olympiad (MLO),[7] since its object of study is what the other sciences, albeit different, have in common: reasoning.

Our main goal in this work is to portray the MLO and describe the proportion of women and men within its XIth edition with special focus on the issue of women's underrepresentation.

[1] Olimpiada Mexicana de Matemáticas.
[2] Olimpiada de Química del Distrito Federal.
[3] Olimpiada Mexicana de Física.
[4] Olimpiada Mexicana de Informática.
[5] Olimpiada Mexicana de Historia.
[6] Olimpiada Mexicana de Filosofía.
[7] Olimpiada Mexicana de Lógica.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 17–26

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

In order to accomplish these goals we review the concept of logic (Section 2), we give details about how the MLO works (Section 3) and we discuss the data obtained during the MLO (Section 4). Finally, we close with a summary and remarks about future work (Section 5).

2 General aspects of logic

Reasoning is a process that produces new information given previous data by following certain norms that can be defined mathematically. These norms allow us to describe inference as the unit of measurement of reasoning: inference may be correct or incorrect depending on those norms. The science that studies such norms is Logic.

The understanding of these norms depends on three equivalent approaches: the semantical, the syntactical, and the abstract one. From the semantical standpoint the central concept of such norms is that of *interpretation* and defines our notions of *satisfaction*, *model* and *logical truth* as denoted by $\models$ and introduced by Tarski's *On the Concept of Logical Consequence* [19]. From the syntactical point of view the main concept of those norms is that of *deducibility* and characterizes our intuitions of *proof*, *demonstration* and *theorem* as denoted by $\vdash$ and introduced by Carnap's *The Logical Syntax of Language* [2]. Finally, from the abstract standpoint the idea behind those norms is the *relation of consequence* that generalizes the previous accounts as in Tarski's *On some Fundamental Concepts of Metamathematics*, LSM, art. III [18], hence rendering logical consequence as the *proprium* of logic.

This relation has several uses and representations. The typical use is in argumentation, the typical representation is in the argument, usually defined as a finite sequence of propositions $\phi_1, \ldots, \phi_n$ ordered in such a way that $\phi_1, \ldots, \phi_{n-1}$ are *premises* and ϕ_n, the *conclusion*, is obtained by such relation of consequence. Consider a couple of examples:

▶ Example 1. Either Jones owns a Ford, or Brown is in Boston. But Brown is not in Boston. Thus, Jones owns a Ford.

▶ Example 2. All persons against unfair laws are rational. Every rational being is free. Thus, every free person is against unfair laws.

Example 1 encodes a correct argument, Example 2 does not. Justifying the (in)correction of these examples is beyond the scope of this work, but we can briefly show why Example 2 is incorrect by using (standard) Logic. Let us suppose Example 2 is a correct argument, then it must be impossible to find a counter-example, that is to say, it must be impossible to find a set of true premises that satisfies the argument's structure but yields a false conclusion; however, if we use the substitution set $\Theta = \{$*persons against unfair laws/positive number, rational being/number greater than 0, free person/real number*$\}$ we obtain true premises and a false conclusion, hence showing the argument is not correct.

3 Details of the MLO

In an effort to promote the further study of the norms of Logic in order to develop scientific and critical thinking, in 2004 a group of graduate students from the First Certificate in Logic (*Primer Diplomado en Lógica*) offered by the National Autonomous University's Institute for Philosophical Research (*Instituto de Investigaciones Filosóficas*) and the Mexican Academy of Sciences (*Academia Mexicana de Ciencias*) (Cristian Diego Alcocer, María del Carmen Cadena Roa y Ricardo Madrid†), and thanks to the support by the Mexican Academy of Logic (*Academia Mexicana de Lógica*), called the First National Logic Olympiad (*Primera Olimpiada Nacional de Lógica*). Since then there have been eleven consecutive editions that

have gathered students, professors, and researchers from all over the country, Argentina, Colombia, and Paraguay. Up next we give details about how the MLO works.

3.1 Divisions and stages

The MLO is divided into three divisions:

- High school (*Bachillerato*).
- Undergraduate (*Licenciatura*).
- Masters.

The High School and Undergraduate divisions are self-explanatory; the Masters division, introduced during the XIth edition for the first time, includes advanced undergraduate students that have been winners in previous editions and graduate students.

These divisions enter the competition in two stages:

- Knock-out round.
- Final round.

3.2 Exams

The exams are composed by 30 multiple choice questions designed and reviewed by the Academic Committe of the MLO composed by researchers, professors, and winners of previous editions. Since the MLO is a competition designed to measure logical skills, and not knowledge about logic, it requires logical training in three big areas that we are going to illustrate with references and toy examples (the examples, of course, do not fully express all the complexity and quality of the exams, but we hope they will give a general impression of how regular exams look like):

- Classical First Order Logic.
- Informal Logic.
- Problem Solving.

Classical First Order Logic captures deductive reasoning skills and requires, for instance, contents similar to Copi's *Symbolic Logic* [3], Enderton's *Mathematical Introduction to Logic* [4], Gamut's first volume of *Logic, Language, and Meaning* [7], Smullyan's *First-Order Logic* [16], or Mendelson's *Introduction to Mathematical Logic* [12]. The next one is a typical decision example:

▶ Example 3. *Since some animals are vertebrate and some cats are vertebrate, it follows that some animals are cats.* How is the previous argument?

- a) Valid, with true premises, and true conclusion
- b) Invalid, with true premises, and true conclusion
- c) Invalid, with false premises, and false conclusion
- d) Valid, with true premises, and false conclusion
- e) Invalid, with true premises, and false conclusion

Informal Logic tries to capture non-deductive skills related to argumentation and supposes, for instance, Perelman & Olbrechts-Tyteca's *Treatise on Argumentation* [13], Toulmin's *The Uses of Argument* [20] or Walton's *A Pragmatic Theory of Fallacy* [21]. The next one is a typical fallacy detection example:

▶ Example 4. What fallacy best suits the next argument? *If Europe is a Jupiter's satellite, then Europe spins around Jupiter. Since there are goats living in Europe, it must be the case that there are goats spinning around Jupiter.*

- a) False cause
- b) Ambiguity
- c) *Ad populum*
- d) Hasty generalization
- e) *Ad hominem*

Problem Solving tries to find combined reasoning skills to solve problems, riddles, and puzzles: it assumes contents similar to that of Smullyan's classics such as *What is the name of this book?* [16]; the next example, however, is due to Gardner:

▶ Example 5. You've got three boxes. Each one contains two balls. One box contains two black balls; another box contains two white balls; the last box contains one black ball and one white ball. The boxes are labelled according to the color of the balls it contains: BB, WW, BW. Now, someone has changed the labels of the boxes in such a way that every box is now incorrectly labelled. You can extract one ball at a time from any box without looking inside and, by this process, you must determine the content of each box. What is the smallest number of drawings needed to accomplish this task?

- a) 4
- b) 1
- c) 3
- d) 5
- e) 2

4 Results

The XIth edition of the MLO took place in Puebla City on may 31, 2014, at the facilities of UPAEP, and was organized by the Mexican Academy of Logic (AML) and the Faculty of Philosophy and Humanities at UPAEP. With the available data, thanks to the AML, we have pointed out some interesting facts regarding the participation of women.

The first fact we notice is that *division* and *gender* are inversely proportional variables: in High School the slight majority was composed by women (51%), while in the Undergraduate and Masters division the presence was smaller, 27%, and 24% respectively (Figure 1).

The second fact we observe is that when we consider the set of medallists (i.e, three first places) the High School and Undergraduate divisions were dominated by men (100%), while in the Masters division a woman achieved a third place (Figure 2). Note that, since the MLO grants a medal to each and every winner, the number of medallists is greater than three; the same happens with the first ten places by division.

Thirdly, given that the MLO also acknowledges the first ten places by division, we report that 25%, 26%, and 24% were women, respectively (Figure 3).

Finally since, as we said previously, Science Olympiads also target professors, we report the participation of coaches by gender and division: 19%, 27%, and 25% were women, respectively (Figure 4).

These descriptive results around the variables of *division* and *gender* are quite interesting, but we have also looked into some other variables such as the coaches' gender w.r.t students' gender (Figure 5), state (i.e., *city*) (Figure 6), and current major/background (Undergraduate/Masters) (Figure 7).

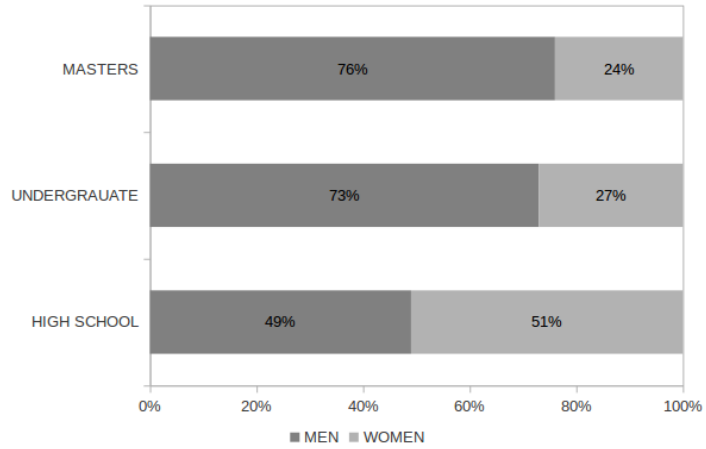


Figure 1 Proportion of gender by division

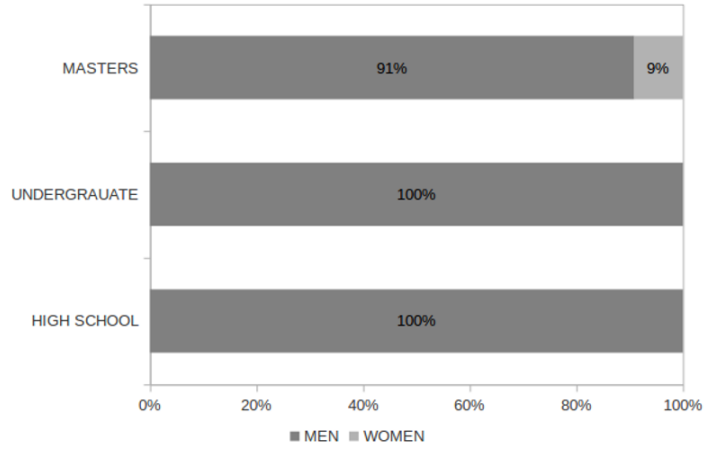


Figure 2 Proportion of medallists by division

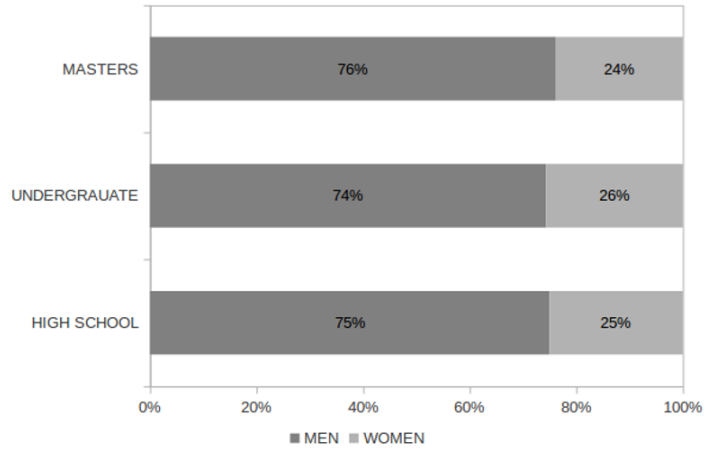


Figure 3 Proportion of first ten places by division

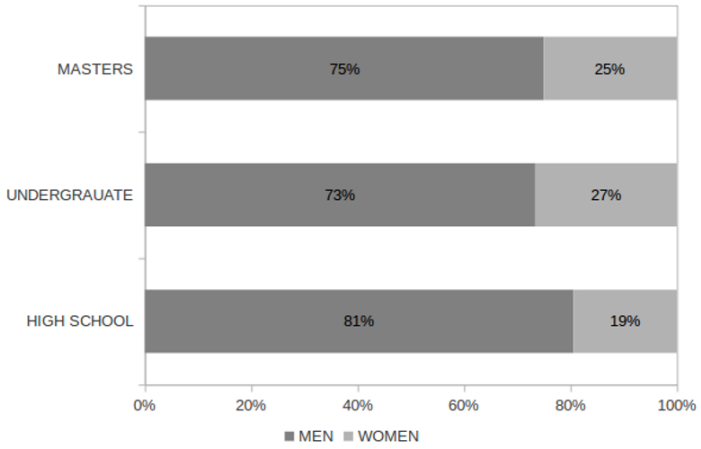


Figure 4 Proportion of coaches by division

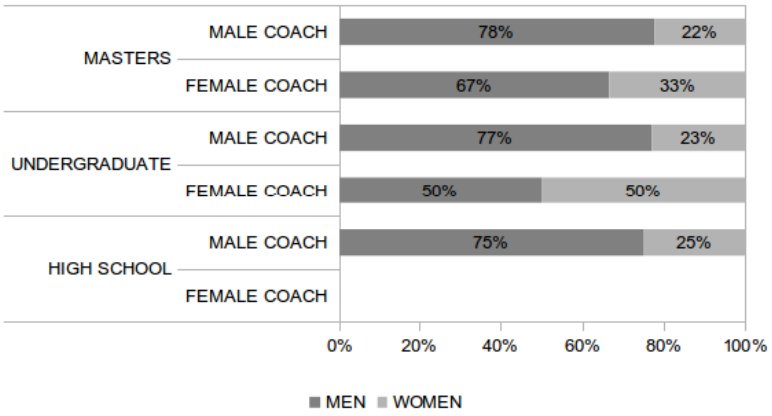


Figure 5 Proportion of coaches' gender w.r.t students' gender

5 Conclusion

Although we are currently studying more data and testing some other correlations with special focus on the participation of women, we can momentarily conclude that our report indicates that the participation of women seems to be inversely proportional to the division, despite they have been successful in the competition, both as students and coaches. Nevertheless, the cause of this underrepresentation still remains unanswered.

We know there are certain disciplines in which women's participation is significantly lesser than men's (the inverse seems to be true as well), which had led some researchers to consider sexual dimorphism [11, 6], hormonal differences [9, 14, 8] or environmental conditions [1, 5] to provide an answer: their results, albeit interesting, are prone to objections and are quite controversial to say the less [17]. But, as our report shows, it seems clear that there is a serious loss of women when they go from High School to Undergraduate level, and we need an explanation of this phenomenon. We can also observe this loss has an evident effect on the whole competition, as our two-way ANOVA analysis shows (Figure 8), and so, it is clear that we need new policies to improve this situation.

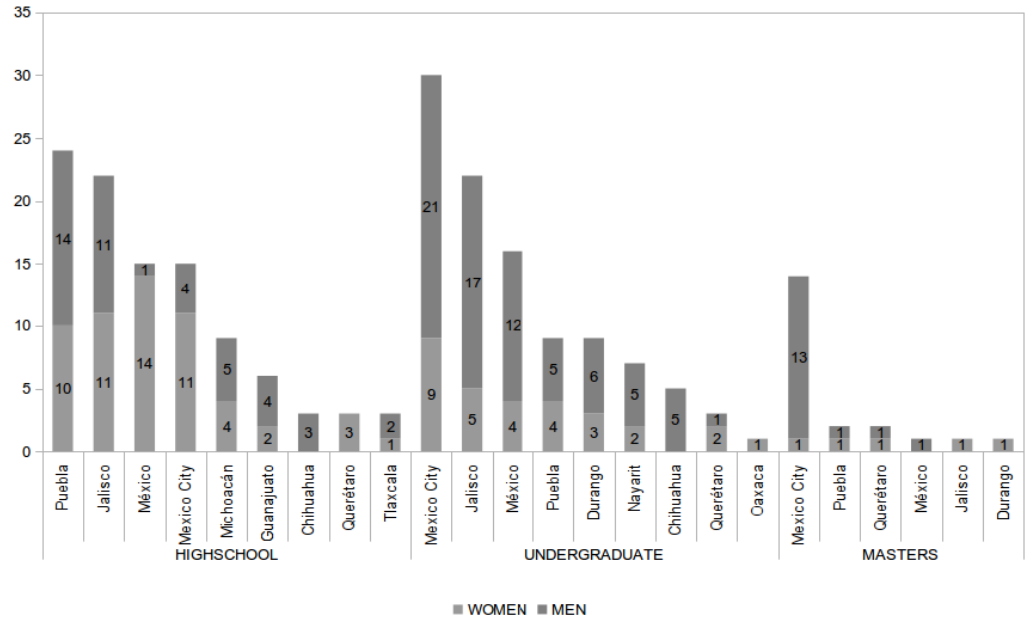


Figure 6 Proportion by state

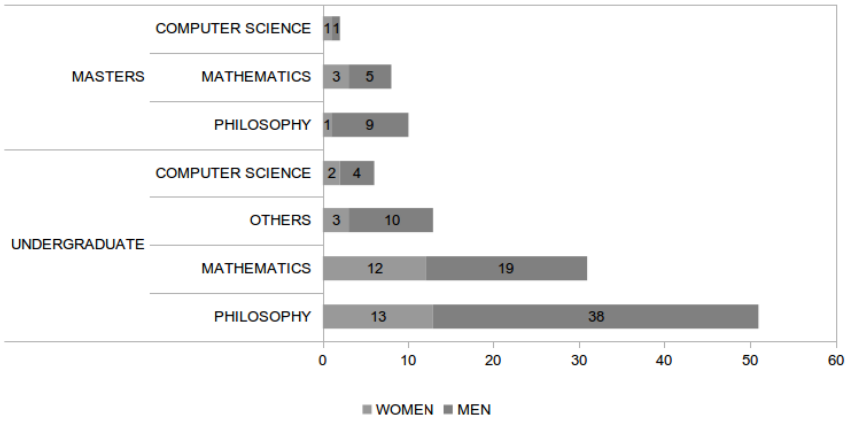


Figure 7 Proportion by major/background

Finally, we would like to add that the answers to Examples 3, 4, and 6 are the options *b*.

Acknowledgements The authors would like to thank the reviewers for their precise corrections and useful comments. Financial support given by UPAEP Grants 30108-1030 and 30108-1008.

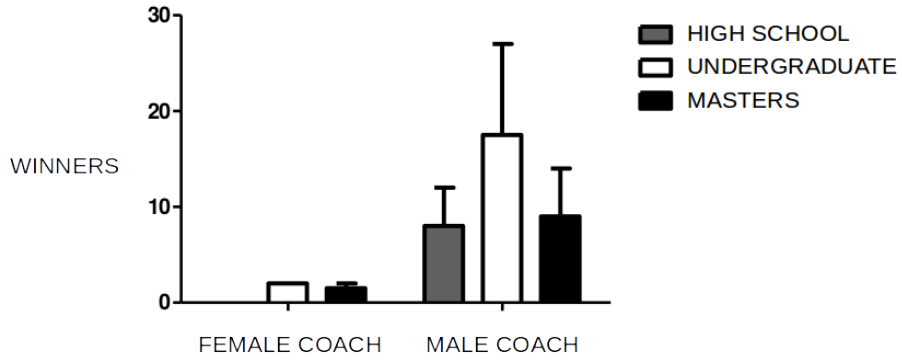


Figure 8 We have checked whether *coaches' gender* has an effect in the results and it turns out this factor has significance. We assessed *coaches' gender vs division* and we observed *coaches' gender* accounts for 45.98% of the total variance giving $F = 7.31$, $DFn = 1$ and $DFd = 6$ with a $P\ value = 0.0354$

References

1 Amunts, K, Jankce, L., Mohlberg, H., Seinmetz, H. Y Zilles, K. "Interhemispheric asymmetry of the human motor cortex related to handedness and gender." *Neuropsychologia*, 38, 304-312, 2000

2 Carnap, R. *The Logical Syntax of Language*. Open Court Publishing, USA, 2002

3 Copi, I. *Symbolic Logic*. Macmillan, 1954

4 Enderton, H.B. *A Mathematical Introduction to Logic*. Academic Press, 2001

5 Fisher, H. *The first sex*. Random House, NY, 1999

6 Frith, U., Vargha-Khadem, F. "Are there sex differences in the brain basis of literacy related skills? Evidence from reading and spelling impairments after early unilateral brain damage." *Neuropsychologia*, 39, 1485-1488, 2001

7 Gamut, L.T.F. *Logic, Language and Meaning*. University of Chicago Press, 1991

8 Halpern, D.F., Tan, U. "Stereotypes and steroids: using a psychobiosocial model to understand cognitive sex differences." *Brain and Cognition*, 45, 392-414, 2001

9 Hampson, E., Kimura, D. "Reciprocal effects of hormonal fluctuations on human motor and perceptual-spatial skills." *Behavioral Neuroscience*, 102, 456-459, 1998

10 Hyde, J.S., Linn, M.C. "Diversity. Gender similarities in mathematics and science." *Science*, 314:599–600, 2006

11 MacCoby, E., Jacklin, C. *The psychology of sex differences*. Stanford University Press, Stanford, 1974

12 Mendelson, E. *Introduction to Mathematical Logic, Fifth Edition*. Chapman and Hall/CRC, 2009

13 Perelman, C., Olbrecths-Tyteca, L. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press, Notre Dame, 1989

14 Postma, A., Winkel, J., Tuiten, A., Vanhonk, J. "Sex differences and menstrual cycle effects in human spatial memory." *Psychoneuroendocrinology*, 24, 175-192, 1999

15 Smullyan, R. *First Order-Logic*. Dover, 1995

16 Smullyan, R. *What Is the Name of This Book? The Riddle of Dracula and Other Logical Puzzles*. Dover, 2011

17 Spelke, E.S. "Sex differences in intrinsic aptitude for mathematics and science?: A critical review." *Am Psychol*, 60:950–958, 2005

18 Tarski, A. "On some fundamental concepts of metamathematics" in *Logic, semantics, metamathematics. Papers from 1923 to 1938*. Hackett Publishing Company, 1983

19 Tarski, A. "On the concept of logical consequence" in *Logic, semantics, metamathematics. Papers from 1923 to 1938*. Hackett Publishing Company, 1983
20 Toulmin, S. *The Uses of Argument*. Cambridge University Press, 1993
21 Walton, D. *A Pragmatic Theory of Fallacy*. University Alabama Press, 2003

Easy Proofs of Löwenheim-Skolem Theorems by Means of Evaluation Games

Jacques Duparc[1,2]

1 **Department of Information, Systems Faculty of Business and Economics**
University of Lausanne, CH-1015 Lausanne Switzerland
Jacques.Duparc@unil.ch

2 **Mathematics Section, School of Basic Sciences**
Ecole polytechnique fédérale de Lausanne, CH-1015 Lausanne Switzerland
Jacques.Duparc@epfl.ch

Abstract

We propose a proof of the downward Löwenheim-Skolem that relies on strategies deriving from evaluation games instead of the Skolem normal forms. This proof is simpler, and easily understood by the students, although it requires, when defining the semantics of first-order logic to introduce first a few notions inherited from game theory such as the one of an evaluation game.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Model theory, Löwenheim-Skolem, Game Theory, Evaluation Game, First-Order Logic

1 Introduction

Each mathematical logic course focuses on first-order logic. Once the basic definitions about syntax and semantics have been introduced and the notion of the cardinality of a model has been exposed, sooner or later at least a couple of hours are dedicated to the Löwenheim-Skolem theorem. This statement holds actually two different results: the downward Löwenheim-Skolem theorem (**LS**$\downarrow$) and the upward Löwenheim-Skolem theorem (**LS**$\uparrow$).

▶ **Theorem 1** (Downward Löwenheim-Skolem). *Let $\mathcal{L}$ be a first-order language, T some $\mathcal{L}$-theory, and $\kappa = \max\{card(\mathcal{L}), \aleph_0\}$.*
If T has a model of cardinality $\lambda > \kappa$, then T has a model of cardinality κ.

▶ **Theorem 2** (Upward Löwenheim-Skolem). *Let $\mathcal{L}$ be some first-order language with equality, T some $\mathcal{L}$-theory, and $\kappa = \max\{card(\mathcal{L}), \aleph_0\}$.*
If T has an infinite model, then T has a model of cardinality λ, for any $\lambda > \kappa$.

The proof of the second theorem (**LS**$\uparrow$) is a simple exercise that relies on an easy application of the compactness theorem joined with a straightforward utilization of **LS**$\downarrow$. The proof of the first theorem is more involved and not that easy to understand for undergraduate students at EPFL. For some basic background and notations we refer the reader to [1, 5].
The usual approach to proving **LS**$\downarrow$ goes through several steps which involve reducing the original theory T to another theory T' on an extended language $\mathcal{L}'$ where all statements are in Skolem normal form. Then obtaining some $\mathcal{L}'$-structure of the right cardinality that satisfies T', from which one goes back to a model that satisfies T.
The burden of going through the Skolem normal forms is regarded as bothersome by the student.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 27–34


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We would like to advocate that the use of *evaluation games* greatly simplifies the proof of **LS**↓. Of course, this requires to talk about finite two-player perfect games and the related notions of player, strategies, winning strategies, etc. But it is worth the candle, especially if these games are introduced for explaining the semantics of first-order logic. Anyhow, determining whether a first-order formula holds true in a given structure is very similar to solving the underlying evaluation game as put forward by Jaako Hintikka [4]. We refer the readers unfamiliar with these notions to [6, 7], where the tight relations between logic and games are disclosed.

2 Evaluation games for first-order logic

We noticed that when presented with both the classical semantics of first-order logic and the semantics that makes use of evaluation games, the audience gets much more involved in the second approach. Indeed, most students are more eager to solving games than to checking whether a formula holds true.

Moreover, introducing first-order formulas not as (linear) sequences of symbols, but rather as trees (usually denoted as decomposition tree) makes it even easier to give evidence in support of the game-theoretical way of dealing with satisfaction. The reason is twofold:

1. the whole arena of the evaluation is very similar to the tree decomposition of the formula. It has the very same height and the same branching except when quantifiers are involved, where it depends on the cardinality of the domain of the model.

2. the task of pointing out the occurrences of variables that are bound by a given quantifier – in order to replace them by an element of the domain chosen by one of the players – is easily taken care of by taking the path along the unique branch that leads from a given leaf of the tree where the occurrence of the variable is situated, to the root of the tree. The first – if any – quantifier acting on this variable that is encountered is the one that bounds it.

We recall the definition of the evaluation game for first-order logic.

▶ **Definition 3.** Let $\mathcal{L}$ be a first-order language, ϕ some *closed* formula whose logical connectors are among $\{\neg, \vee, \wedge\}$, and $\mathcal{M}$ some $\mathcal{L}$-structure.

The evaluation game $\mathbb{E}v\,(\mathcal{M}, \phi)$ is defined as follows:

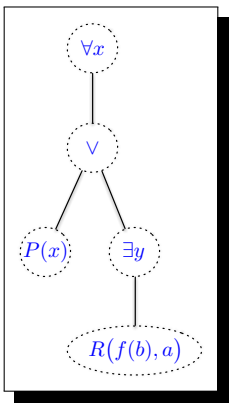
1. there are two players, called ***V**erifier* and ***F**alsifier*. ***V**erifier* (**V**) has incentive to show that the formula holds in the $\mathcal{L}$-structure ($\mathcal{M} \models \phi$), whereas the goal of ***F**alsifier* (**F**) is to show that it does not hold ($\mathcal{M} \not\models \phi$).

   The moves of the players essentially consist of pushing a token down the tree decomposition of the formula ϕ – as a way to choose sub-formulas – and must comply with the rules below:

if the current position is...	whose turn ...	the game continues with...
$\varphi_0 \vee \varphi_1$	**V** chooses $j \in \{0,1\}$	φ_j
$\varphi_0 \wedge \varphi_1$	**F** chooses $j \in \{0,1\}$	φ_j
$\neg\varphi$	**F** and **V** switch roles	φ
$\exists x_i \varphi$	**V** chooses $a_i \in \|\mathcal{M}\|$	$\varphi[a_i/x_i]$
$\forall x_i \varphi$	**F** chooses $a_i \in \|\mathcal{M}\|$	$\varphi[a_i/x_i]$
$R(t_1,\cdots,t_k)_{[a_1/x_1,\cdots,a_n/x_n]}$	the game stops	**V** wins $\iff$ $\mathcal{M}, a_1/x_1,\cdots,a_n/x_n \models R(t_1,\cdots,t_k)$

2. The winning condition arises when the remaining formula becomes atomic, i.e. of the form $R(t_1,\cdots,t_k)_{[a_1/x_1,\cdots,a_n/x_n]}$. Notice that the rules guarantee that, since the initial formula is closed, one always ends up with an atomic formula that does not contain any more variable as each of them has been replaced by some element of the domain of the model [1].
   Player **V** wins if $R(t_1,\ldots,t_k)$ is satisfied in the extended $\mathcal{L}$-structure $\mathcal{M}, a_1/x_1,\ldots,a_n/x_n$ ($\mathcal{M}, a_1/x_1,\ldots,a_n/x_n \models R(t_1,\ldots,t_k)$); **F** wins otherwise.

▶ **Example 4.** Suppose we have a language that contains a unary relation symbol P, a binary relation symbol R, and a unary function symbol f. We then consider the formula $\forall x\Big(P(x) \vee \exists y R\big(f(x),y\big)\Big)$ whose tree decomposition is

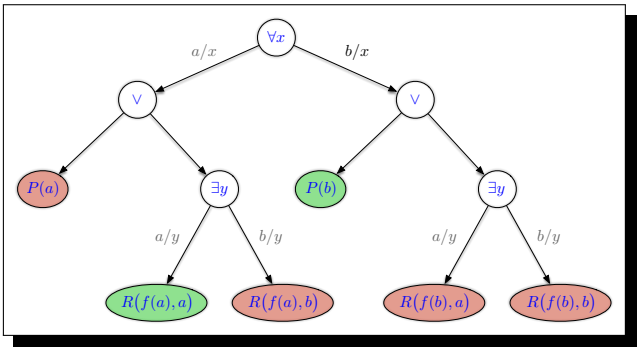


[1] Formally we should not say that we replace each variable x_i by some element a_i, but rather that we replace x_i by some *brand new* constant symbol c_{a_i}, whose interpretation is precisely this element a_i and the formula we reach at the end is of the form $R(t_1,\ldots,t_k)_{[\mathsf{c}_{a_1}/x_1,\ldots,\mathsf{c}_{a_n}/x_n]}$.

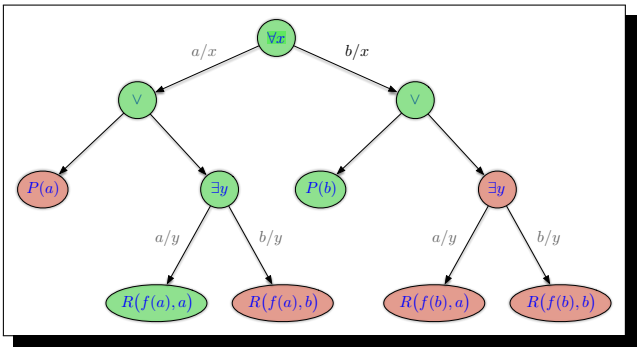
the model $\mathcal{M}$ is defined by:

- $|\mathcal{M}| = \{a, b\}$,
- $f^{\mathcal{M}}(a) = b$.
- $f^{\mathcal{M}}(b) = a$
- $R^{\mathcal{M}} = \{(b, a)\}$.
- $P^{\mathcal{M}} = \{b\}$,

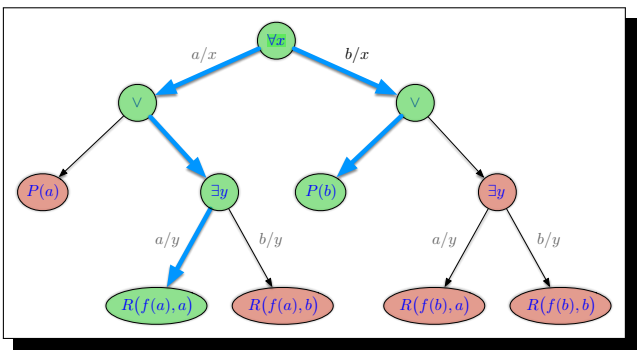
The game tree that represents the arena for the evaluation game $\mathbb{E}v\,(\mathcal{M}, \forall x\,(P(x) \vee \exists y R(f(x), y)))$ is played on the arena represented by the following game tree:



The green leaves are the ones where the atomic formula holds true in the model, and the opposite for the red ones.
We then proceed by *backward induction* and assign either the colour green or the colour red to every node depending on whether the ***V****erifier* or the ***F****falsifier* has a winning strategy if the game were to start from that particular node.



We end up this way with the root being coloured green which shows that the ***V****erifier* has a winning strategy. We indicate below by blue arrows such a winning strategy for the ***V****erifier*.



3 The classical proof of LS↓ with Skolemization

Given any theory T as in Theorem 1, without loss of generality, one first assumes that every formula $\phi \in T$ is in *prenex normal form* i.e. $\phi = Q_1 x_1 \dots Q_k x_k \psi$, where, for all $1 \leq i < j \leq k$, $Q_i, Q_j \in \{\forall, \exists\}$, $x_i \neq x_j$ and ψ is quantifier free. Then, the usual proof of **(LS↓)** goes through the following steps

1. The *skolemization* of each such ϕ, which consists of
   - **a.** first extending the language by adding, for each existential quantifier that ϕ contains, a new function symbol $f_k^{\phi(q_k)}$ of arity q_k.

     $$\mathcal{L}' = \mathcal{L} \cup \left\{ f_k^{\phi(q_k)} \mid Q_k = \exists, q_k = card\left(\{m \mid 1 \leq m < k \wedge Q_m = \forall\}\right) \right\}.$$

     With this definition q_k coincides with the number of universal quantifiers which precede the existential Q_k.
   - **b.** For each existentially quantified variable x_k, replacing inside ψ, each occurrence of x_k by the term[2] $f_k^{\phi}\left(x_{p_1}, \dots x_{p_{q_k}}\right)$, where $x_{p_1}, \dots, x_{p_{q_k}}$ are the universally quantified variables that precede Q_k. More formally:

     $$t_k = f_k^{\phi}\left(x_{p_1}, \dots x_{p_{q_k}}\right)$$

     where $\{x_{p_1}, \dots x_{p_{q_k}}\} = \{m \mid 1 \leq m < k \wedge Q_m = \forall\}$ and the sequence of subscripts $(p_i)_{1 \leq i \leq q_k}$ is strictly increasing. We then obtain

     $$\tilde{\psi} = \psi_{\left[t_{k_1}/x_{k_1}, \cdots, t_{k_m}/x_{k_m}\right]}$$

     where $\{x_{k_i} \mid 1 \leq i \leq m\}$ is the set of all existentially quantified variables of ϕ.
   - **c.** Removing all universal quantifiers from ϕ. The Skolem normal form of ϕ – denoted σ_ϕ – becomes:

     $$\sigma_\phi = Q_{i_1} x_{i_1} \dots Q_{i_t} x_{i_t} \tilde{\psi}$$

     where the sequence of subscripts $(x_i)_{1 \leq i \leq t}$ runs through all universally quantified variables of ϕ.

   This way, any $\mathcal{L}$-theory T is turned into its skolemized version: some $\mathcal{L}'$-theory $\sigma_T = \{\sigma_\phi \mid \phi \in T\}$.
2. One shows that the cardinality of the set of new function and constant symbols that have been added to the language is either countable if the language is finite, and it is the same as the one of the original language $\mathcal{L}$ if it is infinite. Therefore the extended language $\mathcal{L}'$ has cardinality $\max\{card(\mathcal{L}), \aleph_0\}$.
3. One takes any $\mathcal{L}$-structure $\mathcal{M}$ of cardinality $\lambda > \kappa$ such that $\mathcal{M} \models T$ and construct some $\mathcal{L}'$-structure $\mathcal{M}'$ by extending $\mathcal{M}$ from $\mathcal{L}$ to $\mathcal{L}'$. This is done by providing for every new symbol $f_k^{\phi(q_k)}$ an interpretation

   $$f_k^{\phi(q_k)^{\mathcal{M}}} : |\mathcal{M}|^{q_k} \mapsto |\mathcal{M}| \text{ such that it satisfies } \mathcal{M}' \models \sigma_T.$$

   With the classical approach, the description of the extension $\mathcal{M}'$ is usually messy, whereas it simply does not exist with the game-theoretical approach.

[2] A function symbol whose arity is zero is simply a constant symbol.

4. One constructs some $\mathcal{L}'$-structure $\mathscr{N}'$ of cardinality κ such that $\mathscr{N}' \models \sigma_T$ holds. So, one selects some sub-domain of $|\mathcal{M}'|$ which is closed under all interpretations of functions [3] of $\mathcal{L}'$. For this purpose, one starts with any subset $N_0 \subseteq |\mathcal{M}'|$ of cardinality κ which contains all interpretations of constant symbols from $\mathcal{L}'$. We let $\mathscr{F}_k(\mathcal{L}')$ denote the set of function symbols of $\mathcal{L}'$ of arity k. By induction, one defines

$$N_{n+1} = N_n \cup \{f^{\mathcal{M}'}(a_1, \ldots, a_k) \mid k \in \mathbb{N}, f \in \mathscr{F}_k(\mathcal{L}'), a_1, \ldots, a_k \in N_n\},$$

and one sets $N_\omega = \bigcup_{n\in\mathbb{N}} N_n$. One observes that:

a. $N_0 \subseteq N_1 \subseteq \ldots \subseteq N_\omega$.

b. For all $n \in \mathbb{N}$, $card(N_{n+1}) = card(N_n) = \kappa$, since the induction yields

$$\begin{array}{rcl} \kappa & \leq & card\Big(N_n \cup \{f^{\mathcal{M}'}(a_1, \ldots, a_k) \mid k \in \mathbb{N}, f \in \mathscr{F}_k(\mathcal{L}'), a_1, \ldots, a_k \in N_n\}\Big) \\ & \leq & card(\mathcal{L}' \times N_n^{<\omega}) \\ & \leq & card(\kappa \times \kappa^{<\omega}) \\ & \leq & \kappa \cdot \kappa \\ & \leq & \kappa \end{array}$$

c. Hence $card(N_\omega) = \kappa$ holds, since

$$\kappa \leq card(N_\omega) \leq card\left(\prod_{n\in\omega} N_n\right) \leq card(\aleph_0 \times \kappa) \leq \aleph_0 \cdot \kappa \leq \kappa.$$

d. N_ω is closed under all interpretations of function symbols from $\mathcal{L}'$. For if $k \in \mathbb{N}$, $f \in \mathscr{F}_k(\mathcal{L}')$, and $a_1, \ldots, a_k \in N_\omega$, there would then exist some $n \in \mathbb{N}$ such that $a_1, \ldots, a_k \in N_n$, henceforth $f^{\mathcal{M}'}(a_1, \ldots, a_k) \in N_{n+1} \subseteq N_\omega$.

Therefore $\mathscr{N}'$ is defined by:

- $|\mathscr{N}'| = N_\omega$;
- for all constant symbols c, $c^{\mathscr{N}'} = c^{\mathcal{M}'}$;
- for all fonction symbols f of arity k, $f^{\mathscr{N}'} = f^{\mathcal{M}'}|_{N_\omega{}^k}$;
- for all relation symbols R of arity k, $R^{\mathscr{N}'} = R^{\mathcal{M}'} \cap (N_\omega)^k$.

5. Finally, one shows that the model $\mathscr{N}$ defined as the restriction of $\mathscr{N}'$ to the language $\mathcal{L}$ satisfies $\mathscr{N} \models T$. This requires once again to review the construction of $\mathscr{M}'$, by backward this time: another wearisome moment for the students.

4 An alternative game-theoretical proof of LS↓

Compared to the previous proof, the game-theoretical proof that we advocate is simpler for it only focuses on the original model and fixed winning strategies that witness that the theory holds in this model. Indeed, this proof contains

- no Skolem normal form
- no extended language $\mathcal{L}'$
- no extension $\mathcal{M}'$ of $\mathcal{M}$
- no restriction $\mathcal{N}$ of $\mathcal{N}'$

In this proof, we start with any model $\mathscr{M}$ such that both $card(|\mathscr{M}|) \geq \kappa$ and $\mathscr{M} \models T$ hold. We also assume that every formula $\phi \in T$ is in prenex normal form[4].

---

[3] Including the constants which are the ones of arity 0.

[4] We suppose that $T \neq \emptyset$ holds; otherwise, the result is simply straightforward.

1. For each $\phi \in T$ we pick any winning strategy σ_ϕ for the ***V**erifier* in $\mathbb{E}v\,(\mathcal{M}, \phi)$. By looking at the very rules of the game, every student realises immediately that for each existential $Q_i x_i$ in ϕ, the given strategy secures one element from $|\mathcal{M}|$ that *only depends on* the previous choices made by her opponent (***F**alsifier*). Since ϕ is in prenex normal form, these choices made by ***F**alsifier* correspond precisely to the universal quantifiers preceding $Q_i x_i$. For instance, if ϕ is of the form

$$\forall x_1 \exists x_2 \forall x_3 \forall x_4 \forall x_5 \psi$$

then the choices that ***V**erifier* makes – following σ_ϕ – of an element $a_2 \in |\mathcal{M}|$ for x_2 and an element $a_5 \in |\mathcal{M}|$ for x_5 depend respectively of the choices made by ***F**alsifier* of an element $a_1 \in |\mathcal{M}|$ for x_1, and of elements $a_1 \in |\mathcal{M}|$ for x_1, $a_3 \in |\mathcal{M}|$ for x_3, $a_4 \in |\mathcal{M}|$ for x_4. In other words, the winning strategy picks for x_i an element that is *function of* – meaning that it only depends on – the choices made for the universally quantified variables that come before x_i. Assuming there is k-many such universally quantified variables, this induces a unique function $f_i^{\sigma_\phi} : |\mathscr{M}|^k \mapsto |\mathscr{M}|$. So we come up with a set $\mathscr{F} = \{f_i^{\sigma_\phi} \mid \phi \in T\} \cup \{f^{\mathcal{M}} \mid f \in \mathcal{L}\}$ of functions of different arities[5] whose cardinality is at most $\kappa = \max\{card\,(\mathcal{L}), \aleph_0\}$.
2. We take any subset $N_0 \subseteq |\mathscr{M}|$ of cardinality κ and proceed as in (4)(a-d) to obtain the least (for inclusion) subset $N \subseteq |\mathscr{M}|$ of cardinality κ that satisfies both $N_0 \subseteq N$ and N is closed under all functions in $\mathscr{F}$.
3. We form $\mathcal{N}$ as the restriction of $\mathcal{M}$ from $|\mathcal{M}|$ to N, and show that $\mathcal{N} \models T$ in a straightforward manner this time, since for every formula $\phi \in T$ the very same strategy[6] σ_ϕ which is winning for the ***V**erifier* in $\mathbb{E}v\,(\mathcal{M}, \phi)$ is also winning for the ***V**erifier* in $\mathbb{E}v\,(\mathcal{N}, \phi)$.

5 An even simpler proof of LS↓

This proof does not even require to go through the formulas of T to be in prenex normal form.

1. for each $\phi \in T$, pick any winning strategy σ_ϕ for the ***V**erifier* in $\mathbb{E}v\,(\mathcal{M}, \phi)$ and for any $A \subseteq |\mathcal{M}|$ consider all possible plays in $\mathbb{E}v\,(\mathcal{M}, \phi)$ such that
   a. ***F**alsifier* restricts his $\forall$-moves to choosing elements of A, and
   b. ***V**erifier* applies her winning strategy σ_ϕ.
   set $(A)^{\sigma_\phi}$ as the subset of $|\mathcal{M}|$ formed of all the $\exists$-moves made by ***V**erifier*. Set also $(A)^{\mathscr{F}} = \left\{ f^{\mathcal{M}}(\vec{a}) \mid f \in \mathcal{L} \text{ a function symbol with arity } k, \vec{a} \in {N_n^{\mathbf{V}}}^k \right\}$,
2. inductively define N by:
   a. $N_0^{\mathbf{F}} \subseteq |\mathcal{M}|$ any set s.t. $card\left(N_0^{\mathbf{F}}\right) = \kappa$, $\{c^{\mathcal{M}} \mid c \text{ constant } \in \mathcal{L}\} \subseteq N_0^{\mathbf{F}}$,
   b. $N_{n+1}^{\mathbf{F}} = N_n^{\mathbf{F}} \cup \left(N_n^{\mathbf{F}}\right)^{\mathscr{F}} \cup \bigcup_{\phi \in T} \left(N_n^{\mathbf{F}}\right)^{\sigma_\phi}$, and $N = \bigcup_{n \in \mathbb{N}} N_n^{\mathbf{F}}$. Then,
   - $card\,(N) = \kappa$
   - $(N)^{\sigma_\phi} \subseteq N$
   - $(N)^{\mathscr{F}} \subseteq N$.
3. Form $\mathcal{N}$ as the restriction of $\mathcal{M}$ to N, and easily verify that $\mathcal{N} \models T$, since the very same σ_ϕ is winning for ***V**erifier* in $\mathbb{E}v\,(\mathcal{N}, \phi)$.

[5] Functions of arity 0 being identified with elements of the domain $|\mathscr{M}|$.
[6] Strictly speaking this is the restriction of this strategy to N.

6 Conclusion

We tried these proofs on our own students at EPFL. It turns out that they understand much better the proof of LS↓ that we recommend than they buy the classic one. It also requires much less time to teach, and above all we deeply believe that this proof highlights what is essential in this result now devoid of all the technical details of the skolemization. On the other hand, there is a price to pay in doing so: one has to present the semantics of first-order logic through evaluation games. But here also we have noticed that the students learn more easily this other part of the course. Another advantage of the game-theoretical approach is also that it paves the way for the *back-and-forth* method [3] or the Ehrenfeucht-Fraïssé games [2] that are intensively used in model theory [5].

References

1 René Cori and Daniel Lascar. *Mathematical Logic: Part 1: Propositional Calculus, Boolean Algebras, Predicate Calculus, Completeness Theorems*. Oxford University Press, 2000.

2 Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math*, 49(129-141):13, 1961.

3 Roland Fraïssé. Sur une nouvelle classification des systèmes de relations. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 230(11):1022–1024, 1950.

4 Jaakko Hintikka and Esa Saarinen. *Game-theoretical semantics: Essays on semantics*, volume 5. Springer, 1979.

5 David Marker. *Model theory: an introduction*. Springer, 2002.

6 Jouko Väänänen. *Models and games*, volume 132. Cambridge University Press, 2011.

7 Johan van Benthem. Logic and game theory: Close encounters of the third kind. In Gregory Mints and Reinhard Anton Muskens, editors, *Games, logic, and constructive sets*. CSLI Publications, 2003.

The Sequent Calculus Trainer – Helping Students to Correctly Construct Proofs

Arno Ehle, Norbert Hundeshagen, and Martin Lange

School of Electrical Engineering and Computer Science
University of Kassel, Germany

Abstract

We present the Sequent Calculus Trainer, a tool that supports students in learning how to correctly construct proofs in the sequent calculus for first-order logic with equality. It is a proof assistant fostering the understanding of all the syntactic principles that need to be obeyed in constructing correct proofs. It does not provide any help in finding good proof strategies. Instead it aims at understanding the sequent calculus on a lower syntactic level that needs to be mastered before one can consider proof strategies. Its main feature is a proper feedback system embedded in a graphical user interface.

We also report on some empirical findings that indicate how the Sequent Calculus Trainer can improve the students' success in learning sequent calculus for full first-order logic.

1 Introduction

Many courses in theoretical computer science suffer from high failure rates and it is common among students to alienate themselves from such courses, c.f. [9, 11]. The reasons are manifold and vary from the way mathematics is taught in school, to lack of general problem solving competences, and not least, to a diversity in skills for adapting new knowledge.

The BSc computer science curriculum at the University of Kassel contains a 2nd-year mandatory course on logic in computer science, as it can be found in typical computer science university programs. This course has been re-organised in recent years with the aim of improving learning outcomes and therefore reducing failure rates. A central point of this re-organisation is the use of constructivistic learning theory, in particular the inverted-classroom model (c.f. [8]). This model focuses on learning, literally, as a self-organised activity; consequently, the logic course should engage students with methods and tools to assist and self-assess the use of formal logic and the calculi taught with them. One of these tools, developed for such purposes, is the *Sequent Calculus Trainer* whose design and use will be described in this paper.

We start by explaining typical problems that students encounter when being faced with a standard exercise in learning Gentzen's sequent calculus [5]: to find a proof for a given sequent, formally expressing that some formula is a logical consequence of a set of formulas. We assume the reader to be entirely familiar with first-order logic with equality [3] and proof calculi in general. Familiarity with sequent calculus in particular is not strictly necessary to follow those explanations; the principles should become clear from the examples we use. We give a brief description of the Sequent Calculus Trainer and explain its aims. At last we provide some empirical data that supports the claim that the Sequent Calculus Trainer can effectively aid the learning of the sequent calculus for first-order logic.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 35–44

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

$$\frac{\Gamma, \varphi, \psi \Longrightarrow \Delta}{\Gamma, \varphi \wedge \psi \Longrightarrow \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi, \Delta \qquad \Gamma \Longrightarrow \psi, \Delta}{\Gamma \Longrightarrow \varphi \wedge \psi, \Delta} \qquad \frac{\Gamma, \varphi \Longrightarrow \Delta}{\Gamma \Longrightarrow \neg\varphi, \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi, \Delta}{\Gamma, \neg\varphi \Longrightarrow \Delta}$$

$$\frac{\Gamma, \varphi \Longrightarrow \Delta \qquad \Gamma, \psi \Longrightarrow \Delta}{\Gamma, \varphi \vee \psi \Longrightarrow \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi, \psi, \Delta}{\Gamma \Longrightarrow \varphi \vee \psi, \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi, \varphi, \Delta}{\Gamma \Longrightarrow \varphi, \Delta} \qquad \frac{\Gamma, \varphi, \varphi \Longrightarrow \Delta}{\Gamma, \varphi \Longrightarrow \Delta}$$

$$\frac{\Gamma, \psi \Longrightarrow \Delta \qquad \Gamma \Longrightarrow \varphi, \Delta}{\Gamma, \varphi \to \psi \Longrightarrow \Delta} \qquad \frac{\Gamma, \varphi \Longrightarrow \psi, \Delta}{\Gamma \Longrightarrow \varphi \to \psi, \Delta} \qquad \frac{}{\Gamma, \varphi \Longrightarrow \varphi, \Delta} \qquad \frac{}{\Gamma \Longrightarrow s = s, \Delta}$$

$$\frac{\Gamma, \varphi[c/x] \Longrightarrow \Delta}{\Gamma, \exists x\, \varphi \Longrightarrow \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi[t/x], \Delta}{\Gamma \Longrightarrow \exists x\, \varphi, \Delta} \qquad \frac{\Gamma, \varphi[t/x] \Longrightarrow \Delta}{\Gamma, \forall x\, \varphi \Longrightarrow \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi[c/x], \Delta}{\Gamma \Longrightarrow \forall x\, \varphi, \Delta}$$

$$\frac{\Gamma, s = s \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta} \qquad \frac{\Gamma, \varphi[s'/x] \Longrightarrow \Delta}{\Gamma, s = s', \varphi[s/x] \Longrightarrow \Delta} \qquad \frac{\Gamma \Longrightarrow \varphi[s'/x], \Delta}{\Gamma, s = s' \Longrightarrow \varphi[s/x], \Delta}$$

Figure 1 The proof rules of the sequent calculus.

2 The Sequent Calculus

2.1 The Proof Rules

The sequent calculus is a system of proof rules that operate on *sequents* which are pairs of multi-sets of formulas, written $\Gamma \Longrightarrow \Delta$. The intended meaning of such a sequent is that the conjunction over Γ logically implies the disjunction over Δ. The rules listed in Figure 1 operate on formulas in the antecedent Γ or succedent Δ of the rule's conclusion below the line, possibly producing new premisses shown above the line. A proof for a sequent $\Gamma \Longrightarrow \Delta$ is, as usual, a finite tree of sequents formed from $\Gamma \Longrightarrow \Delta$ at its root by successively applying these rules. Each branch must end in an instance of an axiom, i.e. a rule with no premisses.

In the rules for quantified formulas, c must always be a fresh constant – called Skolem constant – that does not occur in the conclusion of this rule already; t must be a ground term over the symbols that occur in the conclusion.

2.2 A Didactic Perspective

A standard exercise in sequent calculus asks for a proof of a given sequent, e.g. $\mathcal{S}_0 :=$

$$\forall x \forall y.\ E(x,y) \to x = f(y) \implies \forall x \forall y \forall z.\ E(x,z) \wedge E(y,z) \to x = y\ .$$

Difficulties and mistakes can generally be put into two categories.

(1) The first one is about *constructing a correct proof*: many students are not able to handle formalisms well; often they can barely parse sequents and apply rules correctly. In this example, one has to introduce new names for the universally quantified variables x, y, z in this order and then decompose the Boolean operators on the right side, yielding $\mathcal{S}_1 :=$

$$\forall x \forall y.\ E(x,y) \to x = f(y), E(a,c), E(b,c) \implies a = b\ .$$

Typical mistakes at this syntactic level are concerned with wrong rule applications and include

- *confusing* rules, for instance applying the rule for conjunctions to a disjunction;
- *misplacing* rules, usually by applying a rule to a genuine subformula rather than a formula in the sequent; in other words not understanding the structure of a sequent;

- *wrong first-order instantiations*, for instance not choosing a fresh Skolem constant when needed;
- *wrong rule instantiations*, for instance by adding the symbols Γ and Δ to the sequent at hand;

and so on.

(2) There are other ways of applying rules in a syntactically correct way in this example, for instance by operating on the formulas in the premiss instead. This, however, is unwise for *finding the right proof*. For this one must proceed as described above and then have the inspiration to double one of the formulas in the premiss and instantiate both copies differently yielding

$$E(a,c) \to a = f(c), E(b,c) \to b = f(c), E(a,c), E(b,c) \implies a = b\ .$$

The rest of this proof task is relatively easy using Boolean elimination rules and very simple reasoning about equalities.

There is a clear dependency between these two challenges: the ones described in (1) need to be met before those in (2) can be met; it is impossible to find a proof unless one is able to construct correct proofs at all. The latter is clearly a very difficult task for students who already struggle with uncertainties like "am I allowed to apply this rule here?", "was the application correct?", "should I introduce a new name or instantiate with an already existing term?", etc.

This gap between syntactical and semantical understanding has been addressed in many fields like teaching programming or simply mathematics (e.g. in [10]). This phenomenon is accurately described in [4] as the ability to "write rather rigorously a simple C program", while they cannot "rigorously write down a mathematical proof of the kind needed in graph theory, formal logic, [...]" There is a hidden hint in this observation on how to tackle the problem of teaching proof calculi. Students seem to easily understand syntactic principles as long as there is a mechanism – like a compiler – which allows them to learn the formalism in a trial-and-error way.

This is where the Sequent Calculus Trainer comes into play. It is supposed to aid the constructing of correct proofs but does not help at all in finding the right proofs. From a logical point of view it is merely a proof assistant, not a theorem prover. From a didactic point of view, its use is supposed to address the students' rote memory such that they become able to reach this gap between syntax and semantics, thus enabling them to start understanding the underlying logical concepts. The Sequent Calculus Trainer therefore adheres to two design principles: it is an easy-to-use and simple assistant for building proof trees in sequent calculus. Moreover, it comes with a compiler-like feedback system, which is known for its benefits in tutorial teaching environments [1].

2.3 Related Tools

There are other high quality interactive proof systems that can be used to train the construction of correct proofs in the sequent calculus. The tool that meets the prerequisites laid out here best is LOGITEXT[1]; others worth mentioning are JAPE [2] and PANDA [4]. None of these treats first-order logic *with* equality, though. This is a major drawback since equality is – possibly together with quantification – one of the most difficult concepts for constructing

[1] `http://logitext.mit.edu`

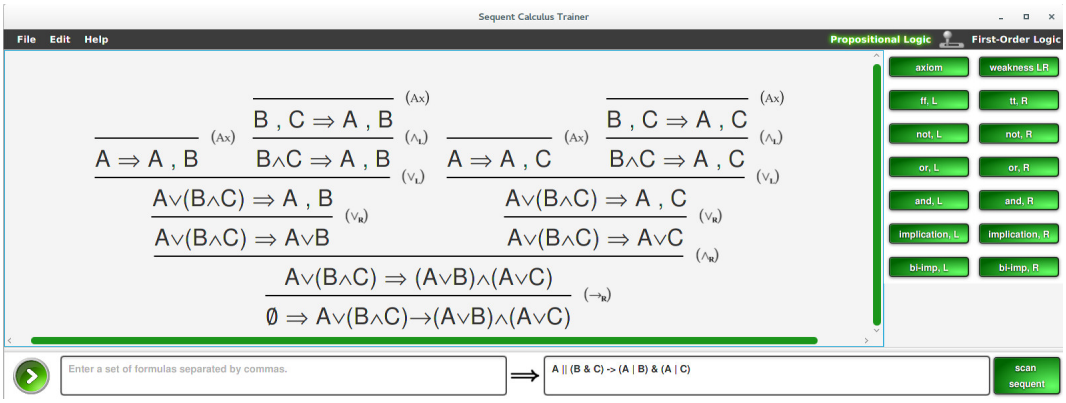


Figure 2 The user-interface.

and finding proofs. Yet equality reasoning is ubiquitous in computer science; hence, it should not be ignored in teaching contexts. Furthermore, JAPE is lacking a feedback system, and PANDA's proof calculus is natural deduction which differs from the sequent calculus.

In a setting where we measure success through the understanding of syntactic concepts, it is essential that the proof calculus used in classroom must be the same as that used by a supporting tool. None of those existing tools is general enough to serve the purposes described above – to act as a tool supporting the learning of proof construction in the sequent calculus for first-order logic with equality. One also should not underestimate the effort that would be needed in order to extend or amend an existing software tool created by others. Hence, having many tools with slightly differing features in this area should be considered advantageous.

3 The Sequent Calculus Trainer

We provide the Sequent Calculus Trainer as an open source application under the BSD-3 license and the source code as well as the binaries are publicly available[2]. Figure 2 shows the graphical user interface which is kept fairly simple. The trainer is shipped with two main views, one for propositional logic and one for first-order logic. They both differ only in the number of applicable rules shown on the right side of the windows, and in the treatment of atomic propositions, which are interpreted as 0-ary predicates in the first-order logic view. Sequents can be input either through a simple text file or in the text fields at the bottom, where the syntax specification for the input is given, too. Furthermore, it is possible to save and load proof trees in an internal format as well as export them in PNG format.

3.1 Key Features

We briefly introduce the two key features of the Sequent Calculus Trainer.

- Nearly every user action leads to a response by the program. Figure 3 exemplarily illustrates such on-screen messages. Each rule button is equipped with a short message, which occurs on mouse-over. These messages usually contain the formal definition of a

[2] `http://www.uni-kassel.de/eecs/fachgebiete/fmv/projects/sequent-calculus-trainer.html`

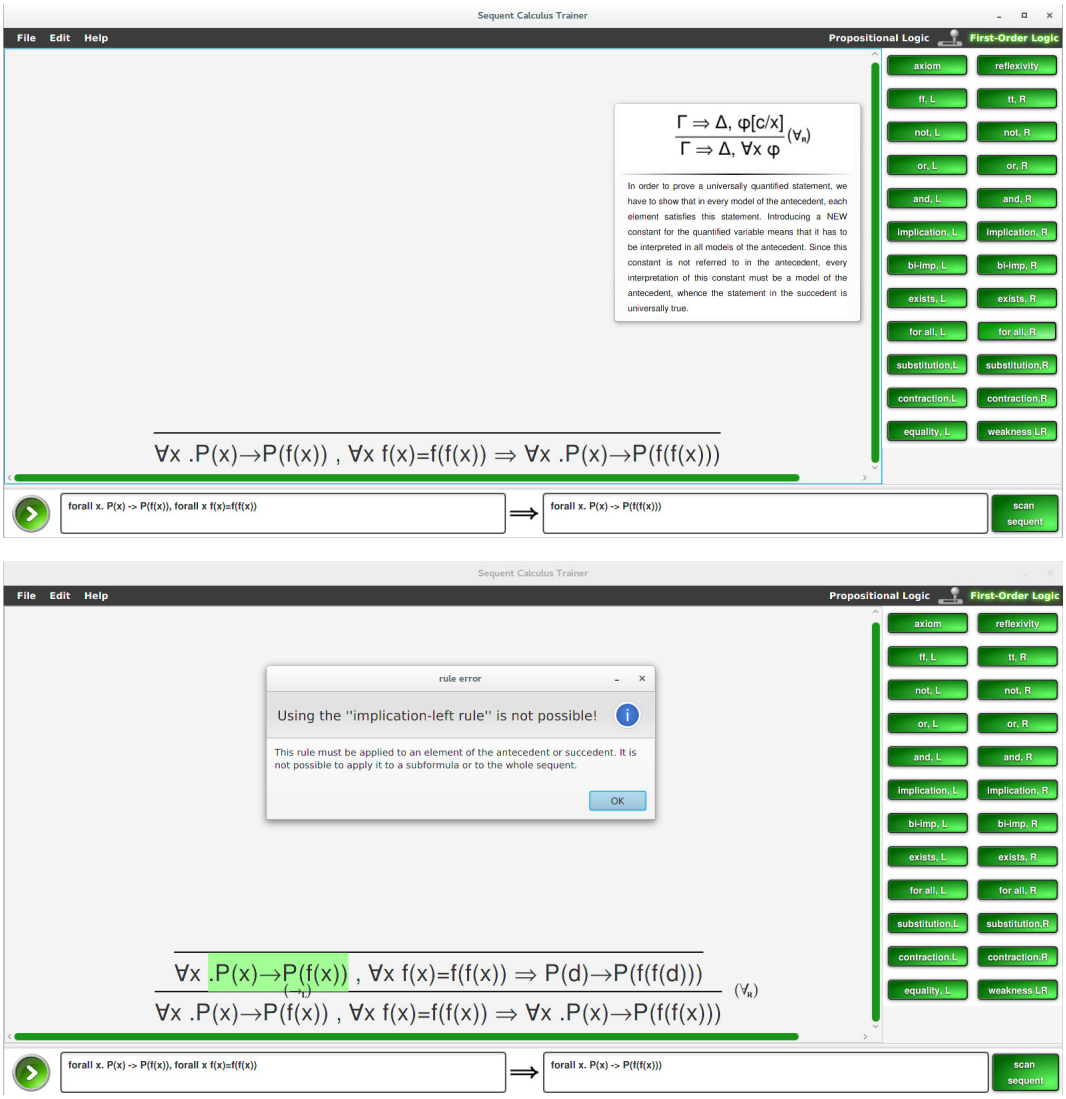


Figure 3 The feedback system.

rule as well as some appropriate high-level explanations of the rule's meaning and, if suitable, why it is a valid logical principle.

If the user has chosen a rule and tries to apply it to a formula by selecting a logical operator, the formula represented by this operator "responds" by telling the user whether or not the rule is applicable there. This happens in two ways: the part of the formula that is in scope of the selected operator or symbol is highlighted. This helps to understand precedence rules and the structure of sequents and formulas. When a wrong operator or symbol is chosen, the user is provided with an error message which includes a hint on the mistake. For instance, the reason why a current leaf in the prooftree is an axiom has to be identified via clicking on the part of the formula that causes the application of an axiom rule.

The second notable feature is the handling of sequents that include equalities. Figure 4 shows how the substitution rule works. After the rule for substitution on the left-hand or

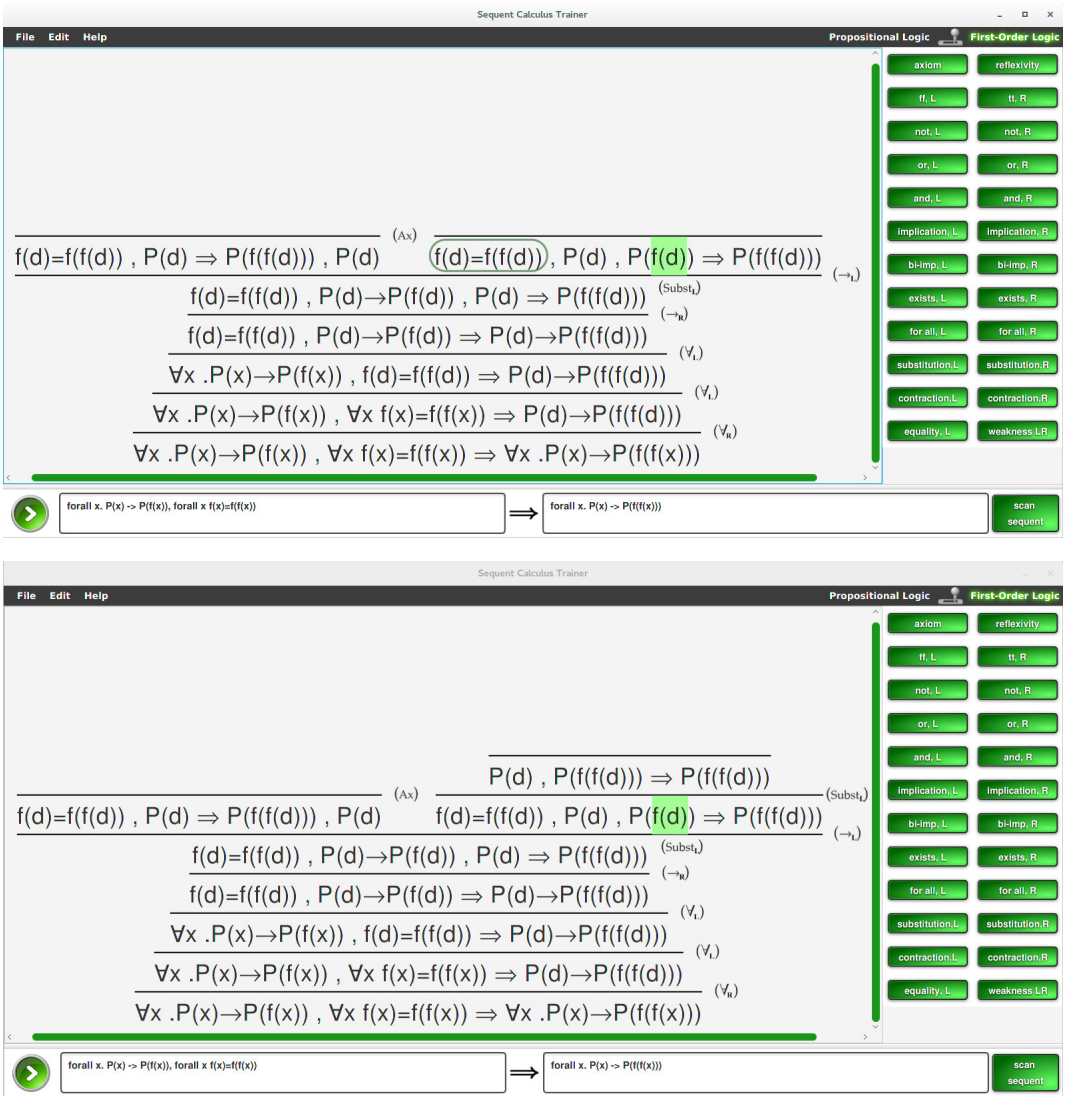


Figure 4 The substitution rule.

right-hand side of the sequent has been chosen, the program expects an atomic formula with an equality predicate to be selected. In the last step, the term that should be substituted needs to be clicked on.

A final point worth mentioning is that the user is able to undo all steps in the proof up to a certain sequent at any time by just applying a different rule to that particular sequent.

3.2 The Backend

The Sequent Calculus Trainer is meant to be used by a broad range of students; hence, it platform independency provided by an implementation in Java is a key design principle. It uses the GUI framework JavaFX, which is integrated in the Java Standard Library since the emerging of Oracle's Java 8. This mainly allows the program to have only one dependency in ControlsFX that is a small extension library for JavaFX, designed to give even more UI

controls and simple dialogs. The result is a clear and comprehensive feedback system which was another key design principle.

The Sequent Calculus Trainer is designed according to the principles provided in the model-view-controller-patterns (MVC). Thus, it consists of three parts; the data model which contains the data and algorithms as a standalone part of the program, the controllers which contain the logic for the GUI, and finally the GUI itself which is build with JavaFX while mainly using the XML based notation for GUI elements called FXML. The program is equipped with the possibility to simply add new languages in form of language sets in simple text files and new country flags for the frontend. Currently German and English are available.

4 Experiences in Teaching the Sequent Calculus

The computer science BSc curriculum at the University of Kassel contains a mandatory 2nd-year course on logic which teaches, amongst others, the sequent calculus for first-order logic with equality. In order to pass the course students need to take a written exam. We report on the results achieved in three successive years. Group 1 took the exam in winter term 2012/13. The Sequent Calculus Trainer was developed afterwards, so it was not available to them at the time of preparation for the exam. Group 2 took the exam in winter term 2013/14, and they were greatly encouraged to solve corresponding exercises using the Sequent Calculus Trainer. Group 3 took the exam in winter term 2014/15. The Sequent Calculus Trainer was made available, but in comparison to group 2 the trainer had been advertised less. The preconditions are comparable in the sense that before the exam, all groups had to pass two graded exercises on sequent calculus, they were provided with the same number of additional voluntary tasks on this topic, and they were allowed to prepare a handwritten note for the exam which contained the sequent calculus rules in many cases.

The three exams under consideration featured a question each, asking for a proof of a given sequent. The three sequents differed, with the ones used for group 2 and 3 being seemingly more difficult to prove with regards to both aspects of constructing and finding a proof.

$$\begin{array}{ll} \text{group 1:} & \forall x \exists y.\ x = v(y) \wedge \forall x\ F(v(x)) \implies \forall x\ F(x) \\ \text{group 2:} & \forall x \forall y.\ E(x,y) \to x = f(y) \implies \forall x \forall y \forall z.\ E(x,z) \wedge E(y,z) \to x = y \\ \text{group 3:} & \forall x \forall z.\ P(x,c) \wedge Q(z, g(x,z)) \implies \exists y \forall x\ P(x,y) \wedge \forall z \exists u\ Q(z,u) \end{array}$$

While the sequents for groups 2 and 3 need some insight into the properties expressed by its formulas, the sequent for group 1 can be proved using almost syntactic considerations only. Surprisingly, out of 70 students in group 1, more than 50% (46 in total) were not able to apply such a simple strategy of applying all possible rules in the correct order. Moreover, when adding the 10 students that did not even try to execute the exercise, we may conclude that 80% of group 1 did not succeed in this exercise because of problems with the correct application of rules. The mistakes most frequently made were *misplacing* of rules and *wrong first-order instantiations* according to the classification listed in Section 2.2. Some examples of typical mistakes are shown in Figure 5.

Group 2 shows a totally different picture. Out of 51 students only 12 already failed in simple rule applications, where 1 student did not execute the exercise, which leads to 25% in total. Thus, nearly 75% of the students where at least able to handle the syntactic formalism resulting in correct rule applications before a deeper understanding would be needed.

The most recent data is taken from group 3. Out of 46 students 19 failed to achieve the mentioned goal and 2 did not execute the exercise which is 46% in total.

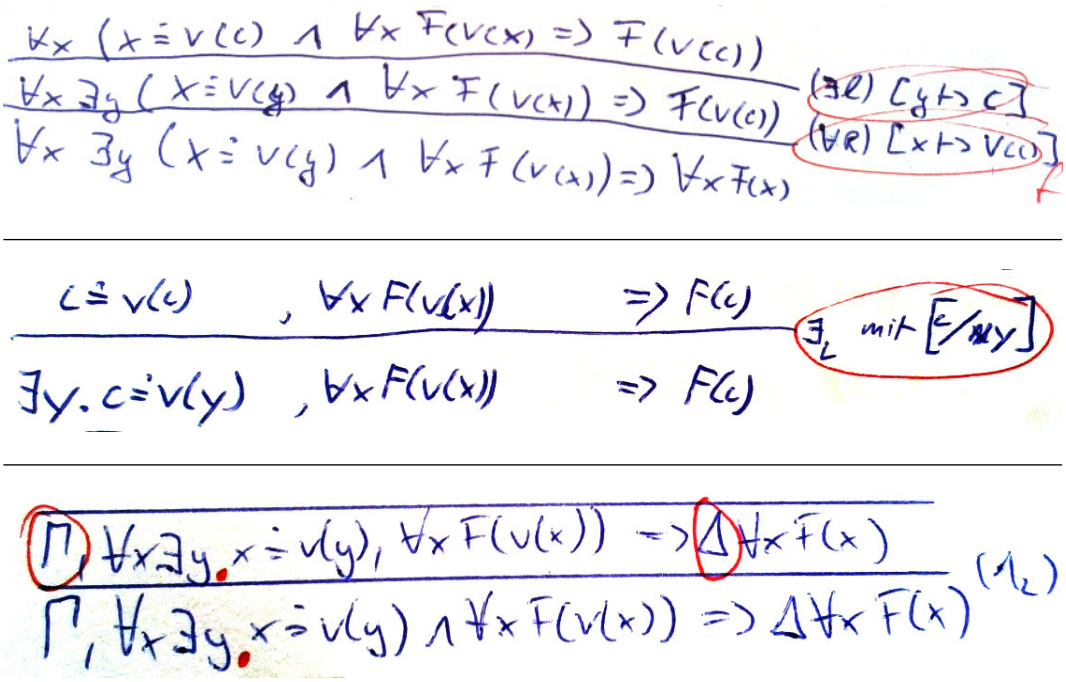


Figure 5 Examples of frequently made mistakes with wrong rule applications.

If we summarize the results of the latter two groups we get a rate of 34 students out of 97 which failed for reasons of wrong application of rules. Thus, nearly 65% of the students where at least able to handle the syntactic formalism resulting in correct rule applications. We believe that this is due to the availability of the Sequent Calculus Trainer.

This conclusion seems to be in contrast to two other – seemingly odd – observations. First of all, group 2 did not achieve significantly more points than group 1. Both exercises were graded with 4 points in total where group 1 reached an average of 1.2 points and group 2 an average of 1.5 points. As mentioned above, from a semantical perspective the sequent for group 2 is harder to proof and, therefore, needs a more involved strategy. This is the main reason for the only slight improvement in the average grade; a syntactically correct but unsuccessful proof attempt is not graded with more than 2 out of 4 points. Hence, the introduction of the Sequent Calculus Trainer into the process of learning how to correctly apply rules resulted in a 25%-gain in points between these two groups. The still low absolute average value achieved by group 2 points out the lack of "understanding" of the underlying theory which is not addressed by the use of the Sequent Calculus Trainer.

Secondly, the results of group 3 seem to worsen in comparison to group 2. Again, the exercise on sequent calculus of group 3 was graded under the same constraints with 4 points in total. The effort needed to prove this sequent is comparable to the effort for the sequent of group 2, as both need a similar strategy of replacing the quantified variables in the correct order. However, the outcome is different. In group 3 more students made mistakes in rule applications although they were also provided with the Sequent Calculus Trainer. They reached an average of 2.7 points. One explanation of this observation can be found when comparing the mistakes made in rule applications. Out of 19 students who tried to solve the exercise and failed in rule applications 7 made the same single mistake of misreading the precedence of the conjunction in the succedent, exemplarily shown in Figure 6. Such "near"

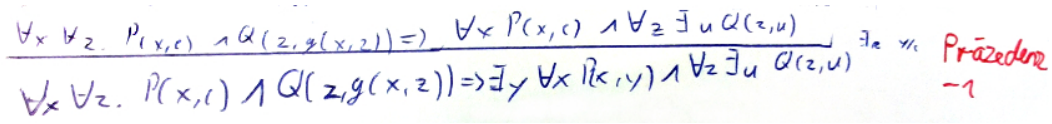


Figure 6 Example of the most frequent mistake in group 3.

solutions were graded with 3 out of 4 points.

5 Discussion and Future Work

The improvement in the exam results of the logic course at the University of Kassel in winter term 2013/14 and 2014/15 compared against those achieved in winter term 2012/13 correlate to the availability and encouragement to actively make use of the Sequent Calculus Trainer. Clearly, the improvement of results in one particular course can be caused by various reasons; it is well known that the teaching staff has the greatest impact on learning outcomes (cf. [7]). In our setting the lectures for all groups were given by the same lecturer and the main part of the exercises was given by the same teaching assistant. The role of the group's composition and the educational background of the students is of course not to be underestimated. However, the significance of the measured effect was too great to be caused solely by the aforementioned reasons. This is strongly emphasised by the fact that the differences in the total outcome of the exams are marginal. Nevertheless, it would obviously be interesting to support this further by a broader empirical evaluation, in particular by considering its effects on students at different universities.

When regarding the effect more deeply, a software for teaching sequent calculus used as an interactive compiler for the language of proof trees perfectly meets the requirements given in Section 2.2 for closing the syntactic gap that may be present in students' minds when introducing a new formal concept. From a different point of view such a piece of software offers a suitable alternative in addressing the rote memory to replace just the right amount of pen and paper exercises, needed to understand the formalism. In addition, it directly forbids mistakes that would be possible to make with pen and paper and would have to be manually marked by human teaching assistants.

Clearly, the goal in teaching a calculus for propositional or first-order logic is not just about the simple manipulation of strings. Instead students need to learn to fluently understand the properties being expressed by logical formulas, to visualise the classes of structures that are being represented by them, etc. In other words, students also need to understand the semantics of logical languages and calculi. The Sequent Calculus Trainer is not meant to address possible deficits in understanding semantics, nor does it do it automatically as the considerations at the end of Section 4 show. We do believe, though, that similar improvement in learning success could be achieved for such semantical aspects by complementing the Sequent Calculus Trainer with a tool that trains the understanding of semantics, for instance using model checking games for first-order logic [6].

Acknowledgements

We would like to thank Angelika Hoffman-Hesse for doing the statistical evaluation of which mistakes were made by how many students as reported above, and Florian Bruse for acting as a linguistic consultant in the development of the Sequent Calculus Trainer. The work has

been financially supported by the *Service Center Lehre* of the University of Kassel under an *Innovation-in-Teaching* grant.

References

1 J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *J. of the Learning Sciences*, 4(2):167–207, 1995.

2 R. Bornat and B. Sufrin. Animating formal proof at the surface: The jape proof calculator. *Comput. J.*, 42(3):177–192, 1999.

3 H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer, Berlin, 2nd edition, 1994.

4 O. Gasquet, F. Schwarzentruber, and M. Strecker. Panda: A proof assistant in natural deduction for all. A Gentzen style proof assistant for undergraduate students. In *Proc. 3rd Int. Congress on Tools for Teaching Logic, TICTTL 2011*, volume 6680 of *LNCS*, pages 85–92. Springer, 2011.

5 G. Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

6 E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

7 J. Hattie. *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*. Taylor & Francis, 2008.

8 M. J. Lage, G. J. Platt, and M. Treglia. Inverting the classroom: A gateway to creating an inclusive learning environment. *J. of Economic Education*, 31(1):pp. 30–43, 2000.

9 Gabriel Robins. Teaching theoretical computer science at the undergraduate level: Experiences, observations, and proposals to improve the status quo. Technical report, Computer Science Department, UCLA, 1988.

10 B. Shneiderman and R. E. Mayer. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *Int. J. of Parallel Programming*, 8(3):219–238, 1979.

11 David R. Surma. Rolling the dice on theory: One department's decision to strengthen the theoretical computing aspect of their curriculum. *J. Comput. Sci. Coll.*, 28(1):74–80, 2012.

Set theory and tableaux for teaching propositional logic

Nino Guallart[*1] **and Ángel Nepomuceno-Fernández**[2]

1 **University of Seville**
Seville, Spain
nguallart@us.es
2 **University of Seville**
Seville, Spain
nepomuce@us.es

Abstract

In this work we suggest the use of a set-theoretical interpretation of semantic tableaux for teaching propositional logic. If the student has previous notions of basic set theory, this approach to semantical tableaux can clarify her the way semantic trees operate, linking the syntactical and semantical sides of the process. Also, it may be useful for the introduction of more advanced topics in logic, like modal logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases semantic tableaux, disjunctive normal form, set theory, propositional logic, propositional model.

1 Introduction

Logic is a discipline that is studied both in sciences as in humanities, so we need to take into account the context in which it is being taught. Our proposal is for teaching logic to students who have completed a course about argumentation theory or informal logic, provided certain basic notions of logic have been studied. However, it may be suitable for students interested in other disciplines such as computer sciences or mathematics.

According to the cross-cutting character of logic, instead of a basic course introduced as usual, we suggest a course on propositional logic in a way that, given a propositional formal language, its semantics will be defined, not only with truth tables but also in terms of set theory. That is to say, after the presentation of semantics by means of truth tables, another presentation of semantics should be given, by means of basic notions of set theory. On the other hand, the method of tableaux introduced in (Beth, 1955) and (Smullyan, 1968) will be taken as a first form of calculus. In both cases graphical representations will be inserted, whether that are known or not (as an auxiliar tool of verification, of course after short explanations when necessary). The corresponding educational methodology allows to tackle easily several relevant topics that can be connected and becomes a general introduction to formal methods with several advantages:

1. Basic set theory, whose rudiments may be known or are being studied simultaneously by students of computer sciences or mathematics, are put in action, which facilitates a rigorous first approach to mathematical logic. For students of humanities, this allows a gradual access to formal treatment of known issues.

* This work has been partially supported by a grant from the V Plan Propio de la Universidad de Sevilla.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 45–54
Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

2. This teaching procedure makes easier the subsequent approach (and understanding) to semantics for first order logics and other topics within this field, like the formation of disjunctive normal form of propositional formulas.
3. Likewise, it facilitates the access to propositional modal logic as a natural continuation of the classical one.
4. Some applications can be understood better with tableaux method. For students of humanities calculi could be more intuitive and ulterior philosophical and metalogical problems could be addressed better. For students of mathematics this approach may be linked to other fields, such as graph theory. For students of computer sciences tableaux can be seen as a way of working in logic next to algorithmical methods.

There are some variations in the way that a semantic tableau can be developed, we will not focus on their differences. For a good starting point on the topic we recommend (D'Agostino, 1999). We will begin with the propositional language with the set theoretic semantics. Tableaux method is described and the corresponding results are settled. Then this method will be put in teaching perspective. To finish, a conluding remarks section is presented and a basic bibliography is incorporated in the paper.

2 Classical propositional logic

Let L be a propositional language defined from a set of atoms or propositional variables $\mathcal{P}$. The syntax must be presented to students by means of a BNF rule:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$$

In order to define the standard semantics, let V be a function from propositions to truth values $V : \mathcal{P} \mapsto \{0, 1\}$. The evaluation of formulas is as usual.

In order to settle the semantics, after presenting the standard one by means of truth tables, an introduction to basic notions of set theory must be made. An atom is true or false with respect to a given situation or state of the world[1], according to the studied truth tables, but we should underlike the interest in the set of situations in which the atom is true, then we can consider a set of states of the world that constitutes the framework to interpret our language L. A propositional model M is now defined as a set $\mathcal{U} \neq \varnothing$, the set of states of the world, and $v_{\mathcal{U}}$ is a function defined from L to $\wp(\mathcal{U})$.

▶ **Definition 1.** $M = (\mathcal{U}, v_{\mathcal{U}})$ (though we shall write v instead of $v_{\mathcal{U}}$ to abbreviate) that accomplished the following cluases:

1. If $p \in \mathcal{P}$, $v(p) \in \wp(\mathcal{U})$ —or, what is the same, $v(p) \subseteq \mathcal{U}$—
2. $v(\neg\varphi) = \mathcal{U} \setminus \{v(\varphi)\} = \overline{v(\varphi)}$
3. $v(\varphi \wedge \chi) = v(\varphi) \cap v(\chi)$
4. $v(\varphi \vee \chi) = v(\varphi) \cup v(\chi)$
5. $v(\varphi \rightarrow \chi) = \overline{v(\varphi)} \cup v(\chi)$

Given a model $M = (\mathcal{U}, v)$ the notion of satisfaction, in a state of the world s, of a formula φ —in symbols $M, s \models \varphi$ is given in the following —

▶ **Definition 2.** In keeping with the grammar of L,

[1] From an intuitive point of view. Despite that they were related, we are not using these terms in a specific sense of situation theory for example.

1. $M, s \models p$ if and only if[2] $s \in v(p)$
2. $M, s \models \neg\varphi$ iff $M, s \not\models \varphi$
3. $M, s \models \varphi \wedge \chi$ iff $M, s \models \varphi$ and $M, s \models \chi$
4. $M, s \models \varphi \vee \chi$ iff $M, s \models \varphi$ or $M, s \models \chi$
5. $M, s \models \varphi \to \chi$ iff $M, s \not\models \varphi$ or $M, s \models \chi$

When a formula is satisfiable in standard sense —there is an assignement of truth values to its atoms such that the formula is true—, then we shall say that *the formula is standard-satisfiable*. Then there exists a situation in wich the formula is true, so a model M is definible, that is to say, it is satisfiable in the sense of the semantics in terms of set theory. The relation between both forms of satiafiability is settled in the following

▶ **Theorem 3.** *A formula $\varphi \in L$ is standard-satisfiable iff it is satisfiable in the sense of the semantics in terms of set theory.*

Proof. By induction over the logical degree of φ. ◀

According to Theorem 1, a wff φ is valid iff for all propositional models M, and all situation s, $M \models \varphi$.

▶ Proposition 1. By applying definition 2, the following results can be proven:

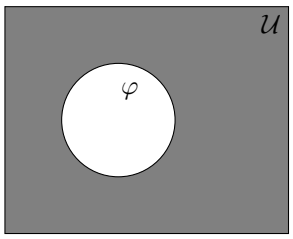
1. If $M, s \models p$, then $v(p) \neq \varnothing$
2. $M, s \models \neg\varphi$ iff $s \in \overline{v(\varphi)}$
3. $M, s \models \varphi \wedge \chi$ iff $s \in v(\varphi) \cap v(\chi)$
4. $M, s \models \varphi \vee \chi$ iff $s \in v(\varphi) \cup v(\chi)$
5. $M, s \models \varphi \to \chi$ iff $s \in \overline{v(\varphi)} \cup v(\chi)$

Specific exercises must be proposed to students to practice theses notions. Example: Is $p \wedge q \to r$ satisfied in a model $M = (\mathcal{U}, v)$ and the state s provided $(v(p) \cap v(q)) \subseteq v(r)$? Let us see both cases:

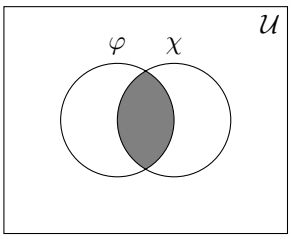
1. $s \in v(r)$, then $s \in (\overline{v(p) \cap v(q)}) \cup v(r)$
2. $s \notin v(r)$, then $s \in \overline{v(r)} \subseteq \overline{v(p) \cap v(q)}$ so that $s \in (\overline{v(p) \cap v(q)}) \cup v(r)$

then, whatever the case may be $M, s \models p \wedge q \to r$.

If we represent subsets of states with Venn diagrams, it is very easy to explain graphically the nature of logical operators. Each circle represents the set of states of the world that make true a certain sentence, φ or χ in this case. In each diagram the whole picture is the universe of states of the world $\mathcal{U}$ and the grey area is the set of states that make true a certain well-formed formula (wff from now on):

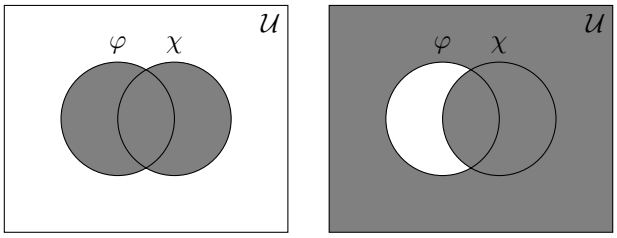


$v(\neg\varphi) = \mathcal{U} \setminus \{v(\varphi)\} = \overline{v(\varphi)}$



$v(\varphi \wedge \chi) = v(\varphi) \cap v(\chi)$

[2] "iff" from now on.



$v(\varphi \vee \chi) = v(\varphi) \cup v(\chi)$ $\quad$ $v(\varphi \to \chi) = \overline{v(\varphi)} \cap v(\chi)$

3 Basic theory for semantic tableaux method

In this section we are going to introduce basic definitions and theorems for the use of semantic tableaux. Since the aim of this paper is pedagogical, not theoretical, this section has a simple purpose, namely to clarify and justify the methods that will be applied as it is shown in the next section. We will focus on adopting semantic tableaux method for propositional classical logic. Given a finite set of formulas Γ, $T(\Gamma)$ denotes the semantic tableau, which is a sequence of sequences of formulas called branches. Each branch is obtained by means of application of rules to formulas that are not literals until the branch is closed (when two complementary literals occur into it) or there is no complex formula without applicating the corresponding rule. A classification of complex formulas:

α	α_1	α_2
$\varphi \wedge \chi$	φ	χ
$\neg(\varphi \vee \chi)$	$\neg\varphi$	$\neg\chi$
$\neg(\varphi \to \chi)$	φ	$\neg\chi$

β	β_1	β_2
$\varphi \vee \chi$	φ	χ
$\neg(\varphi \wedge \chi)$	$\neg\varphi$	$\neg\chi$
$\varphi \to \chi$	$\neg\varphi$	χ

There are three kinds of rules:

$$\frac{\neg\neg\varphi}{\varphi}; \quad \frac{\alpha}{\alpha_1;\alpha_2}; \quad \frac{\beta}{\beta_1 \mid \beta_2}$$

The following theorems establish the way the algorithm for solving tableaux works, giving a finite tableay for a finite set of formulas which is satisfiable if it is open, and the relationship between this method and the formation of disjunctive normal forms.

The fundamental property of the tableaux is given by the following theorem:

▶ **Theorem 4.** *A finite set of formulas Γ is satisfiable iff $T(\Gamma)$ is open.*

Proof. The proof is very easy and can be found in any text related to semantic tableaux. Here we take it for granted, given the practical nature of this work. ◀

Note that for a finite set of formulas Γ, its tableau is equivalent to the tableau of a single formula, the conjunction of them.

As a corollary, Γ is contradictory (not satisfiable) iff $T(\Gamma)$ is closed. On the other hand, let Γ be a finite set of formulas and φ a formula such that the last is logical consequence of the former, that is to say $\Gamma \models \varphi$, then semantic tableaux method can be applied to study that. Of course, $\Gamma \models \varphi$ iff the set $\Gamma \cup \{\neg\varphi\}$ is not satisfiable, which is equivalent to say that $T(\Gamma \cup \{\neg\varphi\})$ is closed.

For a valild wff ϕ, its negation $\neg\phi$ has no possible interpretation so it is contradictory. In order to test whether a wff is valid, we must check its negation; ϕ is valid iff the semantic tableau of $\neg\phi$ is closed.

Semantic tableaux are usually represented graphically as rooted trees. As a tree, a tableau is a graph with no cycles, so for every two nodes there is exactly one path connecting them. We assign the formula we are analyzing to the root node, and their immediate children nodes are the subformulas obtained by any of the aforementioned rules. Since every node can be seen as the root of a subtree, we can have a complete tableau defined recursively. A node with no branches or a leaf is a terminal node, which is the node of a literal.

The following two theorems are crucial for understanding the practical use of semantic trees:

▶ **Theorem 5.** *The semantic tableau of a finite formula is a finite tree.*

Proof. A child node contains a subformula of the node obtained by the rules, so it has to be a smaller formula than the origina its predecessor. The root is a finite formula or a finite set of formulas, so this means that in every branch we will reach to a literal in a finite number of steps, and it corresponds to a leaf. ◀

Our next step is of semantical nature and crucial in our argumentation, since we want to link the use of semantic tableaux to a set-theoretical interpretation of formulas. We will cover this more in-depth in the next section. However, in order to understand the set-theoretical interpretation of the method, we need to know the relationship between the interpretation of literals and the interpretation of the initial formula.

▶ **Theorem 6.** *The interpretation of a formula is a recursive function of the partial interpretations of its subformulas.*

Proof. Informally, the semantic tableau cuts the formula into its constituting subformulas until their literals are obtained, so we can go the opposite direction to prove the theorem; the model of the conjunction os two subformulas is the conjunction of the models of them, and the model of the disjunction of two formulas is the disjunction of their models. A more exhaustive proof will be included into the final work. ◀

▶ **Definition 7.** A wff is in its disjunctive normal form (DNF) if it is the disjunction of a number of conjunctive clauses, which are logical expressions formed by the conjunction of a finite number of atoms:

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} a_{ij}$$

where a_{ij} is an atom, that is, either a literal or the negation of a literal.

Every wff is equivalent to a DNF, and the transformation can be given in a finite number of steps, via a set of rules based on De Morgan's laws, elimination of double negation, associativity and the distributive laws of one operator over the other, having in mind that $p \to q$ is equivalent to $\neg p \vee q$ and to $\neg(p \to q)$ is equivalent to $p \wedge \neg q$. The DNF of a wff is not unique, since there can be equivalent DNF's of a formula [3]. (See (Quine, 1952) for more details about normal forms and the reduction of formulas to them).

[3] A single atom is also a clause which just one atom, and therefore expressions such as $p \vee (q \wedge r)$ are also DNF's. *Complete disjunctive normal forms* are wff's in its disjunctive normal form in which all clauses

▶ Remark. *Semantic tableaux and disjuctive normal forms.* The method of semantic tableaux for a wff is akin to the conversion of this formula to its DNF, since the α and β rules are equivalent to the rules for the formation of the DNF of a wff. There are some methodological differences, for example that the semantic tableaux gives eventually a set of subformulas, not an explicit disjunction of them, but implicitly the semantic tableau can be seen as a method for getting the DNF of a formula via the obtention of a set of satisfiable subformulas. In this way, the teaching of semantic tableaux is also a graphical method for the obtention of DNF's, maybe clearer for the students than other methods based on purely syntactical rules.

We have established that the process of the tableau is finite and, when the original formula is reduced to a set of disjunctive subformulas that are satisfiable under the same interpretation, that set satisfies the wff. Each branch of the tree gives the conjunction of some atoms, that is, of some literals of the wff and, if the branch is open, it has a valid model that satisfies it. If there is a model that satisfies a subformula, it also satisfies partially the original formula. Since this model is a function from the models of its subformulas, the semantic tableau is basically a method for obtaining its whole model; the whole tree is comprised of branches, and the disjunction of the open branches gives the smallest satisfiable model for the wff.

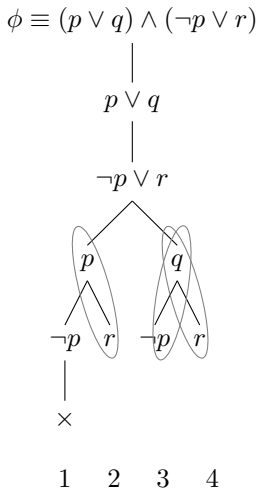
4 Semantic perspective in teaching

Until now, we have seen the theoretical preliminaries of a set-theoretical interpretation of semantic tableaux. Now we are going to show how it works in practical terms, and the usefulness of this method for teaching logic. We are going to sketch the advantages (and some disadvantages) of set theoretical interpretation via semantic tableaux, focusing on a comparison between semantic tableaux and truth tables for determining some properties of formulas, more exactly satisfiability and validity.

4.1 Satisfiability

Semantic tableaux are an easy and quick method for determining the satisfiability of formulas, and their set-theoretical interpretation can be linked to the process in a simple and illustrative way. Let's take a formula in order to check whether it is satisfiable or not; for the purposes of this exposition we will make $\phi \equiv (p \vee q) \wedge (\neg p \vee r)$. From the initial formula, the process of semantic tableaux gives out a set of branches. Now we are going to assign a possible state to every branch, both open or closed. Let's recall that a formula is valid iff it has at least a possible interpretation. Since each branch of a semantic tableau corresponds to an state of a propositional model, the tree of a satisfiable wff has at least one open branch:

contain all the literals in the wff, either positive or negative. A wff have a unique complete disjunctive normal form, but it may be equivalent to several non-complete DNF's. For example, both $\neg p \vee q$ and $(\neg p \wedge q)$ are equivalent to $(\neg p \wedge \neg q) \vee (p \wedge q) \vee (\neg p \wedge q)$.



Observe that the steps in this tree have a set-theoretical interpretation via the rules defined the section 2:

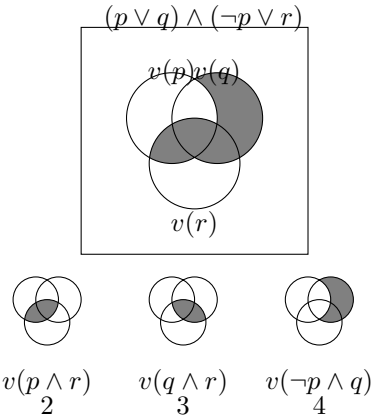
$$v((p \vee q) \wedge (\neg p \vee r)) = v(p \vee q) \cap v(\neg p \vee r) = (v(p) \cup v(q)) \cap (v(\neg p) \cap v(r))) =$$
$$(v(p) \cap \overline{v(p)}) \cup (v(p) \cap v(r)) \cup (\overline{v(p)} \cap v(r)) \cup (v(q) \cap (v(r)) =$$
$$\varnothing \cup v(p \wedge r) \cup v(\neg p \wedge q) \cup v(q \wedge r) = v(p \wedge r) \cup v(\neg p \wedge q) \cup v(q \wedge r)$$

Note that each step in this interpretation corresponds to each line of the semantic tableau, being more clearer on the latter. The set-theoretical interpretation of each branch is the set of states satisfying the formula, which is a subset of the states that satisfy the whole wff; each branch determines a state, and each open branch determines a state that satisfies the formula, so for each state $s_i, i = 2, 3, 4$, we have that $M, s_i \models \phi$. The interpretation of the wff is given by the disjunction of the sets that satisfy the branches.

Since we have four branches, three of them open, we can assign a state to each branch, and satisfiable states are the corresponding ones to open branches, that is, all but 1. Let S ={1,2,3,4} the set of states determined by the tableau, from left to right. Clearly, $v(p) = \{2\}, v(q) = \{3, 4\}, v(r) = \{2, 4\}$, and the interpretation $v(\phi) = \{2, 3, 4\} = S^*$, that is, the union of the states of the open branches. By means of $v(p) \cap v(r) = \{2\}$, $v(q) \cap v(r) = \{3\}$ and $v(\neg p) \cap v(q) = \{4\}$ it can be seen that this model satisfies $\phi \equiv (p \vee q) \wedge (\neg p \vee r)$, and we can take it as a representative of the equivalence class $[S^*]_{\models}$ of all models that satisfy ϕ[4].

Graphically, we can see the same results in the form of Venn diagrams:

[4] Actually, S^* is the representative of the minimum model, since for any model M satisfying this formula (obviously with a cardinality equal or higher to the cardinality of S^*), there is an epimorphism $M \to S^*$ preserving the satisfaction of the wff.



We can also compare this method to the traditional truth tables, showing that the results are the same from two different approaches:

p	q	r	$(p \vee q) \wedge (\neg p \vee r)$	States
1	1	1	1	2,3
1	1	0	0	
1	0	1	1	2
1	0	0	0	
0	1	1	1	3,4
0	1	0	1	3
0	0	1	0	
0	0	0	0	

The truth table offers the complete disjunctive normal form of a wff, since each line that satisfies the wff corresponds to a conjunctive clause, and the whole table can be seen as the disjunction of these (first formula). We can compare it to the disjunctive normal form offered by the semantic tableau:

$(p \vee q) \wedge (\neg p \vee r) \equiv (p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge q \wedge \neg r)$
$(p \vee q) \wedge (\neg p \vee r) \equiv (p \wedge r) \vee (\neg p \wedge q) \vee (q \wedge r)$

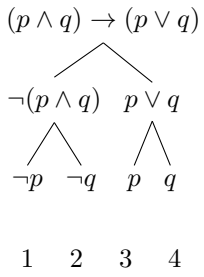
The last column of the table show the equivalence between each truth values for the atoms and their corresponding states. For example, state 2 satisfies the first and the third line, since $\{2\}$ is a model for $(p \wedge q \wedge r)$ and $(p \wedge \neg q \wedge r)$. This comparison also shows that the use of semantic tableaux is usually faster than the construction of truth tables, being this a remarkable advantage of semantic tableaux for checking satisfiability over truth tables in computational terms.

4.2 Validity

If a formula φ is valid, the tableau that begins with φ does not give enough information to prove validity. A valid formula is also satisfiable, so the semantic tree of a valid formula offers an open tree determining a model for it but, when the formula is valid, one or several models are not enough, since all models satisfy the formula. However, we can determine the validity of φ checking the satisfiability of its negation (φ is valid iff the semantic tableau of $\neg\varphi$ is has all of its branches closed), and this is also easily understandable in set-theoretical terms: if $\neg\varphi$ has no propositional model that satisfy it, all propositional models satisfy φ, so

it is a valid formula. Therefore, the direct set-theoretical interpretation of semantic tableaux is not useful for checking the validity of wff.

Let's see an example of the cumbersomeness of the direct approach of this method. If we check a valid formula with this method, for example $(p \wedge q) \rightarrow (p \vee q)$, we can see that we obtain the following tree:



It is obvious that the formula is satisfiable, and that all of the models of the branches are non-empty, $v(p) = \{3\}$, $v(q) = \{4\}$, $v(\neg p) = \{1\}$, $v(\neg p) = \{2\}$, and $v(\varphi) = \{1, 2, 3, 4\}$, but it is not immediate nor evident whether the set of states that satisfy the formula equals to the whole universe of states $\mathcal{U}$, which is the condition for being a valid model. We have to test this with an ulterior method. In this case, it is not difficult to see that

$$\{1\} \cup \{2\} \cup \{3\} \cup \{4\} = v(p) \cup v(\neg p) \cup v(q) \cup v(\neg q) = \mathcal{U}$$

because we have two pair of complementary literals, $v(p) \cup v(\neg p) = \mathcal{U}$ and $v(q) \cup v(\neg q) = \mathcal{U}$. We could have done this directly, ommiting the semantic tableau, so its use is neither useful nor practical. In sum, the semantic tableau of φ only shows if it is satisfiable or not; if we want to check validity using semantic tableaux, we must check the satisfiability of $\neg\varphi$ and proceed as stated.

5 Concluding remarks

This proposal stresses semantics in terms of set theory and the use of semantic tableaux for teaching propositional logic. Both methods together, supplemented with other conceptual tools and showing their complementarieness, can make logic easier to students of any orientation and origin (humanities, computer sciences, etc.). So, for example, it is faster than truth tables, and it can be shown the relationship between a set-theoretical interpretation and a truth table.

As we have said, several advantages of using and combining these techniques can be considered. First, a basic set theory may be known for students of computer sciences or mathematics, but we propose to put it in action to reinforce its notions and appliactions. This basic theory provides a thorough way of working to students of humanities, which can lead gradual access to formal treatment of certain issues and promote the taste for the rigour of thinking. Of course, this facilitates the approach (and understanding) to first order semantics and propositional modal logic in general. Likewise, propisitonal models in terms os set theory of formulas can be obtained systematically by means of tableaux: when a formula is satisfiable, the corresponding tableau allows us to define a minimal universe of states and values for atoms.

Second, tableaux method can be taught as a set of rules of inference based on a semantic compositionality principle, an intutive method of analysis of inferences. On the other

hand, for all kinds of students these methods can develop transversal logical capacities and compentences, which can help them to work other fields (as graph theory, algorithms, language analysis, etc.). In order to normalization of formulas, by means of tableaux disjunctive normal forms can be defined, then the path of learning resolution methos may be open.

To finish, we would like underlying that, though we have no statistical analysis of results, we have some experience in practicing our didactical points of view in Univesity of Seville, particularly with students of humanities, whose understanding of main logical issues has imporved during the last academic years.

References

1 E. W. Beth. *Semantic entailment and formal derivability.* North-Holland, 1955.
2 M. D'Agostino. *Handbook of Tableau Methods.* Kluwer, 1999.
3 L. T. F. Gamut. *Logic, Language and Meaning.* The University of Chicago Press, 1991.
4 S. Hedman. *A First Course in Logic.* Oxford University Press, 2004.
5 W.V. Quine. The problem of simplifying truth functions *The American Mathematical Monthly*, 59(8): 521–531, 1952.
6 R. Smullyan. *First Order-Logic.* Dover, 1968.

Teaching Modal Logic from Linear Algebraic Viewpoints

Ryo Hatano[1], Katsuhiko Sano[1], and Satoshi Tojo[1]

1 **School of Information Science, Japan Advanced Institute of Science and Technology**
1-1, Asahidai, Nomi, Japan
{r-hatano, v-sano, tojo}@jaist.ac.jp

Abstract

This paper proposes a linear algebraic approach to teaching modal logic to students who already have prior knowledge of linear algebra and propositional logic, while we will not assume prior knowledge of first-order logic. Our approach is based on Fitting's linear algebraic reformulation of Kripke semantics of modal logic. A key idea of his reformulation is to represent an accessibility relation R by a square matrix and a valuation $V(p)$ of an atomic variable p by a column vector. Then, we may calculate the truth set of $\Diamond p$ as the multiplication of the square matrix for the accessibility relation and the column vector for p. We discuss how our reformulation is useful to teach modal logic and some restricted form of (nested) quantification to our target students before teaching full first-order logic.

1998 ACM Subject Classification F.4.1 Modal logic

Keywords and phrases Quantification, First-order logic, Modal logic, Linear Algebra

1 Introduction

One of the obstacles to teaching first-order logic (after propositional logic) to students is the notion of quantification and its related notions, i.e., the distinction between $\exists\forall$ and $\forall\exists$, α-equivalence, substitution of variable, etc. It should be also emphasized that bounded and free variables are usually implicit when quantification is used in our natural language. Such features can be well-captured by the syntax of modal logic. In order to make the students to understand these notions on quantification smoothly, we may first teach modal logic to them, since a Kripke model for modal logic is easy to visualize and the students can understand the restricted form of quantification over a visualized model *without* bound variables. One of the points of this paper is: if the students have a prior knowledge of linear algebra (i.e., they are the first or second year undergraduate students), we may teach modal logic more effectively as an intermediate step toward first-order logic.

Our approach of this paper is based on Fitting's linear algebraic reformulation of Kripke semantics of modal logic [2]. A key idea of his reformulation is to represent an accessibility relation R by a square matrix whose component takes Boolean values $\{0, 1\}$ and a valuation $V(p)$ of an atomic variable p by a column vector, provided the cardinality of the domain is finite (see Figure 1).[1] Then, we may calculate the truth set of $\Diamond p$ as the multiplication of the square matrix for the accessibility relation and the column vector for p. One of the merits of this calculation is that it does not require a knowledge of existential quantifier $\exists$

[1] We note that this assumption is often justified because most of the well-known modal logics such as **T**, **S4**, **S5**, etc. enjoy the finite model property, i.e., φ is the theorem of a modal logic Λ iff φ is valid for all *finite* models for the logic Λ.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 55–64

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

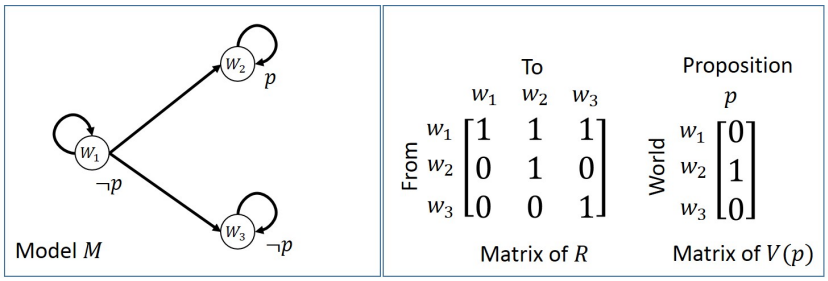


Figure 1 Kripke model and its matrix representation

but just presupposes a simple knowledge of matrix multiplication and Boolean operations on $\{0, 1\}$. This merit allows the students to understand the notion of quantification based on their prior knowledge of linear algebra.

In order to demonstrate our approach, we explain how frame properties such as reflexivity, transitivity, etc. are reformulated in terms of Boolean matrices, and then provide a linear algebraic proof of the implication from reflexivity and Euclidianness to transitivity. Moreover, we also visualize the distinction between $\exists\forall$ and $\forall\exists$ via square matrices, provide a matrix formulation of $\exists y \forall x R(x, y)$ and $\forall x \exists y R(x, y)$ and give a linear algebraic proof of the implication from $\exists y \forall x R(x, y)$ to $\forall x \exists y R(x, y)$. Finally, based on our matrix reformulation of Kripke semantics, we implement a model checker for educational purposes. Our model checker can visualize an input Boolean matrix as a directed graph and check if a given frame property (say, transitivity) is satisfied in the input matrix as a Kripke frame.

We note that Liau [4] also proposes a linear algebraic approach to logic of belief and that Tojo [6] and Hatano et al. [3] extend the linear algebraic approach to capture belief changes of multiple agents with the notion of communication channel.

Here we summarize the structure of this paper. In Section 2, we recall definitions of Boolean matrices and modal logic for preliminaries of remaining section. Section 3 explains a matrix representation of Kripke semantics and its relevant properties from educational viewpoints, and then connects our argument to the concept of quantification in first-order logic. Section 4, we present our implementation based on the concept of the preceding section and show an example for educational purpose. Finally, in Section 5, we summarize our contribution and close with further direction.

2 Preliminaries

2.1 Boolean Matrix

In order to teach modal logic from linear algebraic viewpoints, we recall basic definitions and some important properties of Boolean matrix. The definitions and properties seem almost obvious, however, some operation and properties over Boolean values are different from those of real-valued matrices. For example, the inverse operation of multiplication is not well-defined, and the addition of matrices satisfies idempotence. Those facts will be important when we teach proofs of some properties of modal logic with Boolean matrices. Therefore, we should grasp them first carefully.

Throughout this paper, we use the symbol M, to denote a *Boolean matrix*, i.e., each element of the matrix belongs to the set $\{0, 1\}$. We use the symbol M as a superscript ^{M} with a symbol or expressions (e.g., X^M and $(X + Y)^M$) to denote a matrix representation of them. If the underlying matrix is clear from the context, we omit 'M' from such 'X^M' and

just write 'X'. Then, $M(m \times n)$ means the set of all (Boolean) $m \times n$ matrices, where m and n are the number of rows and columns, respectively. Let M be an $m \times n$ matrix, $1 \leq i \leq m$ and $1 \leq j \leq n$. $M(i,j)$ denotes the element in the i-th row and j-th column entry. Then, $\boldsymbol{E}$, $\mathbf{0}$ and $\mathbf{1}$ denote the *unit square matrix* ($\boldsymbol{E}(i,j) = 1$ if $i = j$; 0 otherwise), *complete matrix* ($\mathbf{1}(i,j) = 1$ for all i and j), and *zero matrix* ($\mathbf{0}(i,j) = 0$ for all i and j), respectively.[2]

Mathematical operations over Boolean matrices are slightly different from real-valued matrices. The Boolean operations of addition '$+$', multiplication '$\cdot$' and complement '$^-$' correspond to the logical operations of '$\vee$', '$\wedge$' and '$\neg$' where $\{0, 1\}$ are replaced by truth values, respectively. [3] Assume such basis, we 'lift' the Boolean operations to the level of matrices. Let $M, M_1, M_2 \in M(m \times n)$. For all i and j,

$$\begin{array}{lcl} \overline{M}(i,j) & := & \overline{M(i,j)}, \\ (M_1 + M_2)(i,j) & := & M_1(i,j) + M_2(i,j). \end{array}$$

Let $M_1 \in M(m \times l)$ and $M_2 \in M(l \times n)$, the multiplication of matrix is:

$$(M_1 M_2)(i,j) = \sum\nolimits_{1 \leq k \leq n} (M_1(i,k) \cdot M_2(k,j)).$$

The transposition ${}^tM$ is defined as: ${}^tM(i,j) = M(j,i)$ for all i and j. Finally, we remark important properties of addition, multiplication and transposition as follows: For any $M, M_1, M_2 \in M(m \times n)$,

$$\begin{array}{lcllcllcl} M & = & M + M, & M & = & \boldsymbol{E}M, & {}^t(M_1 + M_2) & = & {}^tM_1 + {}^tM_2, \\ M & = & {}^t({}^t(M)), & M & = & {}^t(M_1 M_2) = {}^tM_2 {}^tM_1. & & & \end{array}$$

2.2 Modal logic

To introduce Fitting's idea, we recall the definition of well-known modal logic first. A syntax L of modal language is composed of the following vocabulary: a finite set $\mathsf{Prop} = \{p, q, r, \dots\}$ of propositional letters; Boolean connectives $\neg, \vee$; diamond operator $\Diamond$. Then, a set of formulas of L is inductively defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi \mid \Diamond\varphi \quad (p \in \mathsf{Prop}).$$

For $\Diamond\varphi$ can be read as 'it might be the case that φ'. We introduce the dual operator of $\Diamond$ by $\Box\varphi := \neg\Diamond\neg\varphi$ whose reading is 'it must be the case that φ', and also introduce the Boolean connectives $\wedge, \rightarrow$ as usual abbreviations.

Then, we recall also Kripke semantics for the syntax L. A *model* is a tuple $\mathfrak{M} = (W, R, V)$ where W is a non-empty finite set of possible worlds, called *domain*, $R \subseteq W \times W$ is a set of *accessibility relation*, and $V : \mathsf{Prop} \rightarrow \mathcal{P}(W)$ is a *valuation function*. A *frame* is the result of dropping a valuation function from a model, i.e., (W, R) (denoted by $\mathfrak{F}$ etc.).

Let $\mathfrak{M}$ be a model, the *satisfaction relation* $\mathfrak{M}, w \models \varphi$ is defined inductively as follows:

$$\begin{array}{lll} \mathfrak{M}, w \models p & \text{iff} & w \in V(p), \\ \mathfrak{M}, w \models \neg\varphi & \text{iff} & \mathfrak{M}, w \not\models \varphi, \\ \mathfrak{M}, w \models \varphi \vee \psi & \text{iff} & \mathfrak{M}, w \models \varphi \text{ or } \mathfrak{M}, w \models \psi, \\ \mathfrak{M}, w \models \Diamond\varphi & \text{iff} & \mathfrak{M}, v \models \varphi \text{ for some } v \text{ with } wRv. \end{array}$$

[2] Dimensions of those matrices depend on the context.
[3] The inverse operation of the Boolean addition, i.e., subtraction, is not well-defined over Boolean-values. Consequently, subtraction for a Boolean matrix cannot make sense.

A *truth set* $[\![\varphi]\!]_{\mathfrak{M}}$ is defined by $[\![\varphi]\!]_{\mathfrak{M}} = \{\, w \in W \mid \mathfrak{M}, w \models \varphi \,\}$. We say that φ is *valid* on $\mathfrak{M}$ if $\mathfrak{M}, w \models \varphi$ for all worlds $w \in W$.

▶ **Example 1.** We define $\mathfrak{M}$ (see also Figure 1 of Section 1) by: $W = \{\, w_1, w_2, w_3 \,\}$, $R = \{(w_1, w_1), (w_1, w_2), (w_1, w_3), (w_2, w_2), (w_3, w_3)\}$ and $V(p) = \{\, w_2 \,\}$. By definition, $\Diamond p$ is true in w_1 and w_2, $\Box p$ is true in w_2 and $\Box\neg p$ is true in w_3, etc.

Now, we proceed to employ Fitting's idea of reformulation. By definition of Kripke model, we may represent an accessibility relation and a valuation (see the right-hand side of Figure 1) in terms of Boolean matrices. We can observe that the truth set of the formula can be obtained from the calculations of those matrices. For example, the truth values of $\Diamond p$ at each world can be calculated by multiplication of the matrix of the accessibility relation R and the valuation $V(p)$ of Example 1 (see also Figure 1), i.e.,

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The resultant vector meets the result of Example 1. In the next section, we provide an appropriate definition to capture this observation.

3 Linear Algebraic Reformulation of Kripke Semantics

Regarding a possible world as a vertex and an accessibility relation as a directed edge, a frame (W, R) forms a directed graph. If the number of possible worlds is finite, the graph can be written in an adjacency matrix with Boolean values, i.e., *Boolean* matrix.

Here we begin to provide a matrix representation of a frame, then extend it to a model. Let $\mathfrak{F}$ be a frame (W, R), $\#W = m$, and $W = \{\, w_1, w_2, \ldots, w_m \,\}$. R^M is the matrix in $M(m \times m)$ where (i, j)-element is 1 if $(w_i, w_j) \in R$, otherwise 0. In order to obtain our matrix representation of a model, it suffices to consider a valuation function in terms of Boolean matrices. Since $V(p)$ be a subset of W, we regard the valuation $V(p)$ as a column vector defined by $V(p)^M \in M(m \times 1)$ where i's component is 1 if $w_i \in V(p)$, otherwise 0.

Now, it is ready to rewrite our Kripke semantics in terms of Boolean matrix. Let $\mathfrak{M} = (W, R, V)$ be a model. The semantic clauses of each formula φ can be defined by a column vector $\|\varphi\| \in M(m \times 1)$ inductively as follows:

$$\begin{array}{lcl} \|p\| & := & (V(p))^M, \\ \|\neg\varphi\| & := & \overline{\|\varphi\|}, \\ \|\varphi \vee \varphi\| & := & \|\varphi\| + \|\varphi\|, \\ \|\Diamond\varphi\| & := & R^M \|\varphi\|. \end{array}$$

▶ **Example 2.** Let R be a 2×2 matrix and $V(p)$ be a 2×1 matrix ($p \in \mathsf{Prop}$).

$$\|\Box\varphi\| = \|\neg\Diamond\neg\varphi\| = \overline{\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \overline{\begin{bmatrix} x \\ y \end{bmatrix}}} = \begin{bmatrix} (\overline{r_{11}} + x) \cdot (\overline{r_{12}} + y) \\ (\overline{r_{21}} + x) \cdot (\overline{r_{22}} + y) \end{bmatrix}, \text{ where } \|\varphi\| = \begin{bmatrix} x \\ y \end{bmatrix}$$

Now, let us consider the case of $R = W \times W$. By definition, the semantic clauses of the $\Box$ and $\Diamond$ becomes:

$$\mathfrak{M}, w \models \Box\varphi \quad \text{iff} \quad \forall v \in W(\mathfrak{M}, v \models \varphi), \quad \mathfrak{M}, w \models \Diamond\varphi \quad \text{iff} \quad \exists v \in W(\mathfrak{M}, v \models \varphi).$$

These clauses are not restricted by its accessibility relation. If W consists of just two elements, a corresponding accessibility relation becomes $\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$. In this sense, we may regard both clauses of $\Box$ and $\Diamond$ are same as $\forall$ and $\exists$ of first-order logic, respectively.

Let us consider the situation where $\exists x \forall y R(x, y)$, that is, there is some world x from which all the other worlds are accessible. Then, it means that the x-column is filled with 1's. In the similar way, in case $\forall x \exists y R(x, y)$, that is, for each row there must be at least one 1 (see Table 1).

Table 1 Nested quantifications and corresponding matrices (in 3×3)

$\exists y \forall x R(x, y)$	$\forall x \exists y R(x, y)$
$\begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}$.

In order to talk about various frame properties, we now explain that relational union and composition can be defined by matrix addition and multiplication as follows: Given two binary relations $R, S \subseteq W \times W$,

$$(R \cup S)^M = R^M + S^M, \quad (R \circ S)^M = R^M S^M$$

where $R \circ S = \{\, (w, v) \mid (w, u) \in R \text{ and } (u, v) \in S \text{ for some } u \in W \,\}$. Since these operations are from Tarski's relation algebra [5], the reader may wonder if we should teach relation algebra after introducing our linear algebraic approach to modal logic. However, this is not the case. Even if our target students do not have prior knowledge of relational composition and unions, we can introduce these just as the corresponding operations to matrix addition and multiplication. Therefore, our point here is that we may even introduce the notions from Tarski's relation algebra based on Boolean matrices.

Then, we have the following equivalences by set-theoretic properties.

▶ **Proposition 1.** Given any $R, S \subseteq W \times W$, $R \subseteq S$ iff $S = R \cup S$ iff $S^M = R^M + S^M$.

Now we can reformulate well-known frame properties in Table 2 in terms of Boolean matrices, where $\mathbf{1}$ in Table 2 means the column vector of all 1's.

▶ **Proposition 2.** Every frame property in the list of Table 2 can be reformulated in terms of Boolean matrix with elementary matrix calculations.

Table 2 Frame properties in this paper

Name	Condition	Matrix Reformulation	Corresponding Formula
Reflexive	$\forall w(wRw)$	$R = R + \boldsymbol{E}$	$\Box p \to p$
Symmetric	$\forall w, v(wRv$ implies $vRw)$	$R = {}^tR$ (or $R = {}^tR + R$)	$p \to \Box\Diamond p$
Transitive	$\forall w, v, u(wRv$ & vRu imply $wRu)$	$R = RR + R$	$\Box p \to \Box\Box p$
Serial	$\forall w \exists v(wRv)$	$\mathbf{1} = R\mathbf{1}$	$\Box p \to \Diamond p$
Euclidean	$\forall w, v, u(wRv$ & wRu imply $vRu)$	$R = {}^tRR + R$	$\Diamond p \to \Box\Diamond p$

▶ **Proposition 3.** Reflexivity and Euclidianness jointly imply symmetry, i.e., $R = R + \boldsymbol{E}$ and $R = {}^tRR + R$ jointly imply $R = {}^tR$.

Proof. First we observe that if R is reflexive, then the transposition ${}^tR$ is also reflexive i.e., ${}^tR = {}^tR + E$.

$$\begin{array}{rcll} R & = & R + \boldsymbol{E} & \text{(by reflexivity)} \\ & = & ({}^tRR + R) + \boldsymbol{E} & \text{(by Euclidianess)} \\ & = & ({}^tR + \boldsymbol{E})R + \boldsymbol{E} & \\ & = & {}^tRR + \boldsymbol{E} & \text{(by reflexivity of the transposition } {}^tR) \end{array}$$

We have shown $R = {}^tRR + \boldsymbol{E}$. By transposing both sides, ${}^tR = {}^tRR + \boldsymbol{E}$. Since both R and ${}^tR$ are equal to ${}^tRR + \boldsymbol{E}$, we conclude $R = {}^tR$. ◀

We may compare this proof in terms of matrices with the following "ordinary" proof with quantifiers: We show that for any $w, v \in W$, wRv implies vRw. Fix any w, v such that wRv. We show that vRw. By reflexivity, wRw. By Euclidianess, it follows from wRv and wRw that vRw, as desired.

▶ **Proposition 4.** Reflexivity and Euclidianness jointly imply transitivity, i.e., $R = R + \boldsymbol{E}$ and $R = {}^tRR + R$ jointly imply $R = RR + R$.

Proof. $R = {}^tRR + R$ (by Euclidianness) $= RR + R$ (by Proposition 3) ◀

We may also compare this proof in terms of matrices with the following "ordinary" proof with quantifiers: We show that wRv and vRu imply wRu, for any w, v, u. Fix any w, v such that wRv and vRu. By symmetry (Proposition 3), vRw. By Euclidianess, wRu, as desired.

Let us come back to our example of nested quantifiers, i.e., the distinction between $\exists\forall$ and $\forall\exists$. In what follows, we establish "$\exists\forall$ implies $\forall\exists$" in term of matrices. Recall that the property $\forall x \exists y R(x, y)$ of seriality is expressed in terms of Boolean matrix as: $R\mathbf{1} = \mathbf{1}$. The property of $\exists y \forall x R(x, y)$ can be expressed by:

$$({}^t\overline{R})\mathbf{1} \neq \mathbf{1}$$

▶ **Proposition 5.** $({}^t\overline{R})\mathbf{1} \neq \mathbf{1}$ implies $R\mathbf{1} = \mathbf{1}$.

Proof. For simplicity, let R be an 3×3 matrix. Let us write

$$R := \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

We show the contrapositive implication and so assume $R\mathbf{1} \neq \mathbf{1}$. Now, the goal is to show $({}^t\overline{R})\mathbf{1} = \mathbf{1}$. The assumption implies that $[r_{i1}\ r_{i2}\ r_{i3}] = {}^t\mathbf{0}$ for some $1 \leq i \leq 3$. Fix such i. Then, $[\overline{r_{i1}}\ \overline{r_{i2}}\ \overline{r_{i3}}] = \overline{[r_{i1}\ r_{i2}\ r_{i3}]} = {}^t\mathbf{1}$. Since

$${}^t\overline{R} := \begin{bmatrix} \overline{r_{11}} & \overline{r_{21}} & \overline{r_{31}} \\ \overline{r_{12}} & \overline{r_{22}} & \overline{r_{32}} \\ \overline{r_{13}} & \overline{r_{23}} & \overline{r_{33}} \end{bmatrix},$$

$({}^t\overline{R})\mathbf{1} = \mathbf{1}$ holds by $[\overline{r_{i1}}\ \overline{r_{i2}}\ \overline{r_{i3}}] = {}^t\mathbf{1}$. ◀

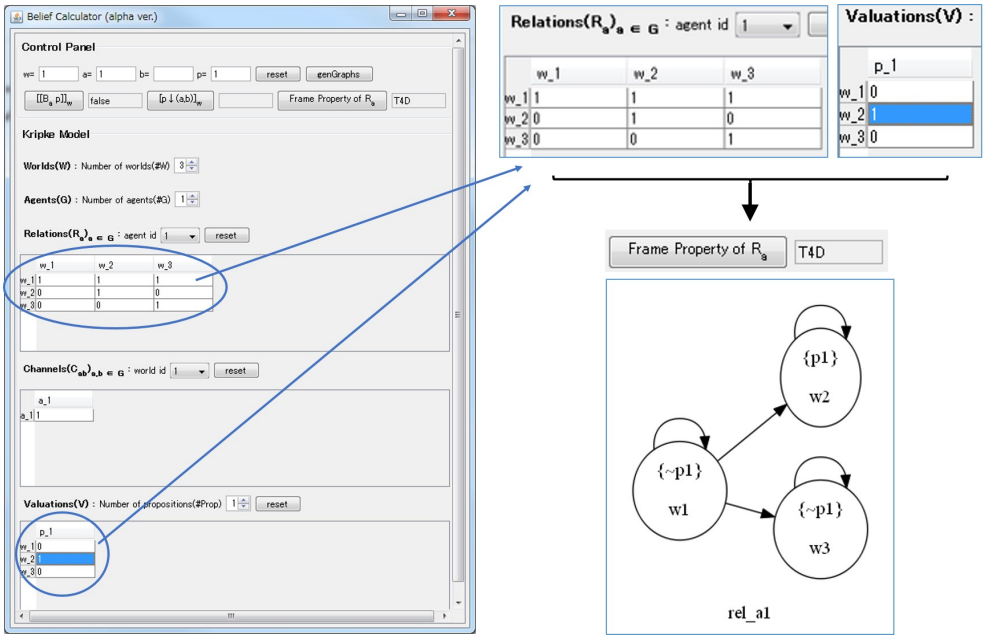


Figure 2 Matrix of Kripke model and its graphical output

4 Implementation

In the preceding section, we have explained a linear algebraic approach to representing the semantics. By these results, we can capture some features of a model and semantics visually, and can also calculate a resultant matrix of the truth value(s) of given formula by hand or any other calculator which allows handling (Boolean) matrices. However, if an objective formula becomes so complicated or the dimension of a matrix is much bigger, some supporting tools must be required. In this section, we introduce our software for such requirement and demonstrate an example of execution from an educational point of view.

A Simple Calculator

We have implemented the preceding semantics with some additional functions in a GUI-based calculator by Java™ 7 and opened for the public.[4] The feature can be summarized as follows: first, we may edit and manage Kripke model by matrix representation. Second, this software can calculate the truth value[5] of a given formula and also can verify frame properties of a given accessibility matrix. Third, we can obtain a visualized output of a given model with the help of Graphviz.[6] We note that this software was originally introduced in [3] to calculate a linear algebraic semantics for dynamic logic of multi-agent communication. In the following example, we focus on a part of the functions which suit our purpose of this paper.

▶ **Example 3.** Figure 2 shows the execution of our implementation. The setting of this example comes from Example 1. A user-defined Kripke model is visualized from given

[4] `http://cirrus.jaist.ac.jp:8080/soft/bc`.
[5] Current version supports only B operator as □ operator.
[6] `http://www.graphviz.org/`

matrices of an accessibility and a valuation via Graphviz. The resultant data are saved to `jpeg` file and immediately appears on the screen. Then, from a matrix of accessibility, we may verify frame properties by `frame property` on the upper right side of the window. We can see the result `T4D`, namely the matrix satisfies the properties of reflexivity, transitivity, and seriality. This was not mentioned in the preceding examples, indeed, we meet a new fact from the given matrix now.

5 Conclusion

We have explained a linear algebraic approach to teaching modal logic to students who have a prior knowledge of linear algebra and propositional logic. Based on Fitting's linear algebraic reformulation of Kripke semantics of modal logic, an accessibility relation R can be regarded as a square matrix with Boolean values, and a valuation $V(p)$ of an atomic variable p also regarded as a column vector. Then, we can calculate the truth set of the formula by the calculations of those matrices. With those results, we have shown some proofs of frame properties of modal logic via matrix calculation as examples for educational purposes.

Our approach is useful to teach modal logic to our target students. Because it does not require a prior knowledge of quantifier of first-order logic. Moreover, it provides a way to understand restricted form of quantification over both a visualized model and its matrix representation. In this sense, we may teach modal logic more effectively as an intermediate step toward the first-order logic. To support this approach, we have introduced our computer system to learn such Kripke semantics.

For further direction, we can use a similar approach to teaching advanced topics. We may expand our mono-modal syntax into multi-modal one to cover, e.g., description logic [1] and dynamic epistemic logic [8], etc. First, as for description logic, a family of *roles* (say "has a child") generates both box-type and diamond-type modal operators, and so the semantics of these operators are also captured by a set of the corresponding adjacency Boolean matrices to the roles. Second, some of dynamic epistemic logics [8] can also be covered by our approach, where multi-modal operators are employed for describing agents' knowledge or beliefs. For example, [3] has presented a linear algebraic semantics of dynamic epistemic logic for multi-agent communication system, with the help of van Benthem and Liu's idea of *relation changer* [7]. A key idea in the notion of relation changer is that a dynamics of beliefs may be described as a *program term* in propositional dynamic logic, i.e., a program term constructed from atomic programs by composition, non-deterministic choice, and test. Since these three program constructors are naturally represented as matrix operations, we may also apply our approach to such dynamic epistemic logic. [7]

References

1 Franz Baader, Deborah L McGuinness, and Daniele Nardi. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, 2003.

2 Melvin Fitting. Bisimulations and boolean vectors. In *Advances in Modal Logic*, pages 97–126. King's College Publications, 2003.

[7] We would like to thank the two anonymous reviewers for their suggestions and comments. The work of the second author was partially supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) Grant Numbers 24700146 and 15K21025.

3 Ryo Hatano, Katsuhiko Sano, and Satoshi Tojo. Linear algebraic semantics for multi-agent communication. In *Proceedings of the 7th International Conference on Agents and Artificial Intelligence*, volume 1, pages 172–181, 2015.

4 Churn-Jung Liau. Matrix representation of belief states: An algebraic semantics for belief logics. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, 12(05):613–633, 2004.

5 Alfred Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

6 Satoshi Tojo. Collective belief revision in linear algebra. In *Computer Science and Information Systems* (*FedCSIS*), *2013 Federated Conference on Computer Science and Information Systems*, pages 175–178. IEEE, 2013.

7 Johan van Benthem and Fenrong Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17(2):157–182, 2007.

8 Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Pieter Kooi. *Dynamic epistemic logic*, volume 337. Springer, 2007.

Making Room for Women in our Tools for Teaching Logic: A Proposal for Promoting Gender-Inclusiveness*

Frederique Janssen-Lauret[1]

1 **Department of Philosophy, University of Campinas**
Rua Cora Coralina 100, CEP 13083-896 Campinas, Sao Paulo, Brazil
`fmjanssenlauret@cle.unicamp.br`

Abstract

Logic is one of the most male-dominated areas within the already hugely male-dominated subject of philosophy. Popular hypotheses for this disparity include a preponderance of confident, mathematically-minded male students in the classroom, the historical association between logic and maleness, and the lack of female role-models for students, though to date none of these have been empirically tested. In this paper I discuss the effects of various attempts to address these potential causes whilst teaching second-year formal and philosophical logic courses at different universities in the UK. I found the most noticeable positive effect came from assigning a good proportion of reading by female authors presenting an original point of view. I go on to suggest some implementations for incorporating more texts by female authors into our arsenal of tools for teaching logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases teaching formal logic, philosophical logic, gender, women in logic, inclusiveness

1 Introduction

Logic is a heavily male-dominated discipline. Men prevail on reading lists, at conferences, and in journals. Although this fact cannot have escaped notice, even those of us who are aware of it and actively attempt to counteract it may be temporarily stunned when presented with the evidence of just how rare it is to see due attention being paid to a woman's original argument in a teaching context in formal or philosophical logic.

Consider the following illustrative example. The University of Cambridge second-year Logic reading list—which covers both formal and philosophical logic—contains, at the time of writing, work by ninety-four distinct men, some of whom are cited many times over; by contrast, only seven women are listed [17]. Two of these women have male co-authors. Four of them are the sole authors of textbooks, historical books, or survey papers; they are presented primarily as commentators on the work of men. Only one of the papers recommended to second-year logic students at Cambridge is a piece of original research in which a female author presents her own point of view: Dorothy Edgington on the paradox of knowability [7].

Men dominate not only the literature, but also the classroom, both as teachers and students. As lecturers and tutors on courses in formal and philosophical logic, we know that these courses generally contain more male students than female students. This appears to be partly due to the general gender disparity in subjects like philosophy and mathematics,

* This work was partially supported by a Capes Postdoctoral Researcher Grant.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 65–74
Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

whose students generally take logic courses, but is also seen to be a result of self-selection by female students away from the area of logic and towards more normative and applied topics. We also frequently see that the male students in courses on formal and philosophical logic are much more vocal than the women. And even though philosophy is generally male-dominated [2], this disparity between male and female students appears to be greater in the area of logic than in other areas of philosophy. Unfortunately no systematic research has been carried out into the reasons for this gender imbalance. But in discussions on the subject, we often hear concerned people point to one or more of the following: the dominance of confident, mathematically-minded male students in the classroom, the historical association between logic and maleness, and the lack of female role-models for students.

In this paper I discuss the results of various strategies which I have used to address these possible causes of gender inequality while teaching formal and philosophical logic to second-year students at the University of St Andrews and at four Cambridge colleges: Clare, Newnham, Selwyn, and Trinity Hall. My ability to make significant changes and investigate issues systematically within these courses was quite limited, because I was involved only in a teaching assistant capacity. But I set out to counteract the potential sources of inequality as best I could with the resources I had available to me. To dispel the impression that men have more of an aptitude for logic or mathematics, I emphasised the importance of working hard over innate ability, encouraged students' efforts in formal work without praising mathematical talent, and drew their attention to the continuity between formal logic and philosophy. All of these, I found, anecdotally had some positive effect in terms of either student feedback or test scores, but the most dramatic positive effect came from consistently assigning a good proportion of original research by female authors as part of the set reading. My proposal is therefore to build teaching materials, including syllabuses, reading lists, and textbooks more specifically around the ideas of women. Where possible a female author should always be presented as the originator of an important view of her own, rather than as a survey author commenting on the work of men.

2 Counteracting Chauvinism and the Association Between Logic and Maleness

When I talk about a pervasive association between logic and maleness as a potential reason that women are put off the academic study of logic, I am not talking about anything as all-encompassing as the idea that the methods of logic are intrinsically linked with oppressive discourse, as proposed, for instance, by Andrea Nye [16]. I mean the idea, or stereotype, which is still commonly held either implicitly or explicitly, that men naturally have more of an aptitude for formal logic and mathematics, or even for rigorous thinking and rationality in general.[1] It is likely that such stereotypes have some effect on female students that results in turning them away from courses in logic, and cause them to be more quiet and diffident compared to their male counterparts when they do take such courses. While teaching formal

[1] Women are not the only marginalised group subjected to the harmful perception that they are less rational than some other, socially dominant group of people. There are racist versions of this stereotype as well, and some currently popular arguments for atheism also paint all religious people with the irrationality brush. Some students, especially Muslim students and others belonging to minority religions, have told me that this has led to their feeling alienated from mainstream philosophy. I am not aware of systematic research on this specific question, but I hope that some of the strategies outlined in this paper may also be effective against these kinds of harmful stereotypes. (As I make clear below, because of the risk of stereotype threat I deliberately avoided saying explicitly that these were efforts to promote gender-inclusiveness specifically.)

logic exercise classes in an intermediate course at the University of St Andrews—comprising some model theory, natural deduction, and Kripke models for modal logics up to S5—I made efforts to challenge this idea. Anecdotally this led to some overall improvement in exam results, and it also occasioned some positive comments on student questionnaires.

2.1 No Good Evidence That Men Have Greater Mathematical Ability

First of all it is helpful to consider the figures which show that the stereotype really is false. There is no clear evidence to corroborate the idea that men have more of an aptitude for mathematics than women do, nor are there recent data that clearly demonstrate that men perform significantly better at mathematical tests than women do. Although in some countries male pupils significantly outperform female pupils, the data available world-wide, for instance those from the Programme for International Student Assessment (PISA), yield very mixed results. Several other countries have female pupils outperforming their male counterparts, or have no gender difference [9, pp. 71-75]. A large global meta-analysis also found only very minor overall gender differences, but great variation between different countries [8]. The causes of this variation are unknown, and there is no clear hypothesis; high performance in mathematics for female pupils does not correlate especially strongly with higher status for women generally [1, p. 411]. Different analyses also reveal different results: although the US is one of the countries where according to PISA, male pupils outperform female pupils on tests, large US-wide meta-analyses have shown no significant differences in school performance in mathematics between male and female pupils [13, 14]. Overall the results appear consistent with the gender similarities hypothesis: that the psychology of men and women is much more similar than different [12], equally so with respect to mathematical ability.

2.2 Avoiding Stereotype Threat

Nevertheless, we know that stereotypes, even when false, are themselves causally efficacious in negatively affecting the test performance of people in underrepresented groups. When teaching intermediate formal logic to second-year students, I wanted to make explicit efforts to counteract the known effects of stereotype threat: the phenomenon of underperformance in disadvantaged groups when they are reminded of their disadvantaged status. This has been shown to apply to women's maths performance [18, 15]. Since formal logic courses are perceived by students as being very mathematical in nature, and are sometimes taken by mathematics students alongside philosophy students (as was the case in the formal logic course I taught), this kind of stereotype threat is likely to affect female students taking courses on formal logic. This implies, amongst other things, that well-intentioned efforts to assure female students explicitly that they are just as capable as the male students have the potential to backfire.

2.3 Emphasising Motivation and Effort over Innate Talent

Another issue that has been identified by social and developmental psychologists as having the potential to hold back female students is that of gendered expectations concerning motivation and feedback. There is some evidence, deriving from extensive studies by Carol Dweck amongst others, that praising students for their efforts, rather than praising them for their innate ability, generally produces more improvement in their performance. But the overall trend is complicated by gendered teaching environments. In Dweck's studies from

the 1970s and 80s, teachers were found to attribute male pupils' failures more often to lack of motivation or situational factors, and girls' failures to intellectual inadequacies [6]. She hypothesised that this might account for gender differences in mathematics achievement, which were much more pronounced in studies from this period than they are now [4].

While the gender gap in school pupils' mathematics achievement has now narrowed, this effect may well still be at work in university-level logic teaching. More recent work by Dweck indicates that adult women are still more likely, in the twenty-first century, to believe mathematics to be a matter of innate ability, and to ascribe their own lack of mathematical achievement to intellectual inadequacy [5]. There are no studies in this tradition that pertain specifically to undergraduate logic courses, but my own experience suggests that the stereotype is persistent. I have certainly personally heard people, including some of my students, say that men are simply better at logic. In my logic classes, I've frequently witnessed female students (and some male students, too) give up at an early stage complaining they don't have 'that kind of brain', attributing their difficulties to lack of innate ability. Male students' difficulties are chalked up to lack of trying, dyslexia, *etcetera*, their successes to mathematical ability. Female students invested in their self-perception as having 'the wrong kind of brain' will explicitly attribute other women's good performance in formal logic to their being atypical women. It may seem as though having female tutors and lecturers would dispel this perception, but as a female logic tutor, and now a female academic working on philosophical logic, I have not found this to be consistently true. I have been told many times, sometimes by female as well as male students, that I must just not be a typical woman to be good at or enjoy logic.[2] I have never had an explicit positive comment from a student about having a female tutor. Specific strategies are needed to make logic-teaching more gender-inclusive.

2.4 My Strategies for Addressing Male-Dominated Classrooms and Harmful Stereotypes

To dispel the impression that logic is a subject that men are naturally more suited to than women, I tried to apply Dweck's theory by emphasising the importance of working hard over innate ability. I wanted to do this in a way that did not explicitly reference the stereotype that logic is 'for men', since this itself carries with it the risk of stereotype threat, which is observed whenever people in disadvantaged groups are explicitly reminded of their disadvantaged status. Instead I attempted to make positive efforts to include all students in completing, demonstrating, and explaining the exercises, and to make them all feel like the course was one they could do well at with some effort, not one that divided the group into the talented ones who would excel at it and the untalented ones who had no hope of doing so.

I stressed frequently to the students that success in logic at the undergraduate level is *not* a matter of talent, or 'having a head for it', or 'having that kind of brain', to use the phrases students often repeat to each other which I view as reinforcing their exaggerated impression of the role played by innate ability. I explained that they would find that the secret to doing well in a logic course is to keep on top of the material and do the exercises every time, without fail. I emphasised that most students were perfectly capable of handling the material, and indicated where to find extra help for those who found it challenging, including help

[2] I am not alone in this; women who are good at any area of the stereotypically masculine subject of philosophy are routinely told that this means they must be masculine somehow [11, p. 212].

for specific learning difficulties like dyslexia or dyscalculia. I wanted to move their thinking away from the rather disablist idea that specific learning difficulties should be classed as 'having the wrong kind of brain', when universities are perfectly capable of providing support and accommodations for these issues. Some students with learning difficulties were provided with accommodations such as extra exam time and dictation software, and did well in the course as a result.

I also explicitly gave the students strategies for reviewing material covered earlier in the course if they were lost or felt baffled by an exercise. In particular, tracing back their steps to last week's exercises, identifying the new component, regaining their confidence in their ability to handle last week's material, and tackling the difficult question with a specific understanding of its relationship to familiar issues and an ability to isolate the newer or more challenging components worked very well for many of the more logic-averse students.

Of course, these solutions are effective only where students are actually made to do the work consistently. I made it mandatory for all students to hand in a full set of proofs every week. This required them to submit their best attempt at solving the problem. I did not allow them to skip any questions or claim that some of the exercises were beyond them. They had to submit at least a few lines of each proof before they could be counted as present for the class, and be given points for attendance. After the initial mild shock all students complied with this policy. I then asked all students in turn to present their proofs on the whiteboard, explain their reasoning to each other, and help each other find the correct answer, stepping in from time to time to moderate and ensure that the discussion was not dominated by the more confident students.

I also made efforts to point out possible applications of the principle at work in a proof, or continuities between logic and philosophy. Some of the more logic-averse students told me that this made them view the course as more interesting and relevant to them.

2.5 Summary of Positive Effects

Although I did not get the chance to investigate the effects of my strategies here in a systematic way, they did appear to lead to improved confidence in previously logic-averse students, some of them male, some female.[3] Class discussion was not in general dominated by confident mathematically-minded men, and this appeared to allow some of the quieter women to speak up and develop their skills and ideas. Some of these women made positive comments to me about feeling more confident or performing better than they had expected—some said this in person, some on their evaluation forms for the course. Overall marks were good and appeared to be slightly better than other instructors' groups, but this difference was not systematically investigated either. It would of course be ideal to see further research into the matter which would produce reliable data. But overall I would recommend all of these strategies to instructors in formal logic courses, since students' responses were positive, their results were good, and my efforts appeared to improve the atmosphere of the exercise classes.

[3] I have since become aware of some promising empirical work that is being done on inclusive methodology in philosophy teaching, though I haven't found anything specific to formal or philosophical logic. For instance, Kristina Gehrman is currently conducting a multi-year study in her introductory philosophy course at the University of Tennessee [10]; her methodology is based on social-role congruence models familiar from science and mathematics [3]. And Eva Cadavid at Centre College is conducting survey research, designed in collaboration with sociologists, concerning the perceptions of philosophy of students in introductory philosophy courses.

3 Introducing Women to Role Models: Assigning Original Texts by Female Authors

Another component to the under-representation of women in formal and philosophical logic is that the paucity of women in the field appears to be self-perpetuating. We often hear people who are concerned about the gender gap in logic speculate that female students are not drawn to the field because they lack female role models. I attempted to remedy the lack of female role models while working as a one-on-one supervisor for several colleges (two co-ed, two women-only) on the University of Cambridge second-year Logic course mentioned above. This is primarily a philosophical logic course with a much smaller—and optional—formal component from the one I taught at St Andrews. I have already pointed out how minuscule the number of women on the official reading list was; in my experience this is not at all unusual, so Cambridge is not noticeably worse in this respect than other universities. The problem of underrepresentation is just very pervasive. My strategy here was assigning some texts from outside the reading lists so that every assignment had at least one piece of set reading by a female author, who whenever possible was presented as the originator of an important view of her own. This strategy was the one that resulted in the most dramatic positive effect that I have seen in my teaching. *All* of my female students, completely unprompted, reacted extremely positively, giving very explicit feedback explaining that it made them feel more included and that it made them enjoy logic more.

3.1 My Strategy to Increase the Proportion of Set Reading by Women

The format of the course included weekly short essay-writing assignments on topics such as reference and descriptions, logical form, theories of meaning, interpretations of quantification, theories of truth, intuitionism, and non-classical logics. I consistently assigned, for each essay, some work by a female logician or philosopher of logic. Works by women comprised at least 25% of the total reading list for each essay. For a few topics it was not possible to find an appropriately accessible piece of original research by a woman, but for the majority of the assignments at least one female author was presented as the originator of an important view of her own—not as a survey author, but as a towering figure in her own right. Because of the overall paucity of female authors in the field, a typical essay question had assigned reading by two or three male authors and one female author. Nevertheless this move particularly enthused the female students. As before, to avoid stereotype threat I did not explicitly draw attention to the genders of the students or the authors; I simply informed them of the essay question and the reading list I had set every week. Even before they were asked to give any feedback on the course, female students told me emphatically that they felt more included and that it made them enjoy logic more.

3.2 Summary of Positive Effects

Female students were very vocal in their appreciation of the original work by female authors, to the extent that I was amazed at the outpourings of enthusiasm. Not only did *all* of the female students, without any kind of prompting from me, comment positively to me about the selection of reading material, but several of them also spoke to their Directors of Studies (Cambridge college subject-specific convenors) who in turn emailed me to pass on even more positive feedback. In previous years I had made extensive use of textbooks and introductory papers by female authors in comparable courses for first- and second-year students, but this never drew a noticeable amount of positive comments from any of the students. Part of the

difference may have been that women were consistently included in reading lists for every assignment. But I believe that the main difference here was that female authors were not presented primarily as commentators on the work of (mostly) men, but as serious thinkers in their own right. Many of the positive comments from female students touched upon this aspect. They were very explicit that they appreciated seeing a female author presented as one of the main experts, as someone whose view they were invited to engage with and take seriously. Several also explicitly said that as a result they liked logic a lot more, and felt that logic was 'for them' as a result, though they had not previously felt that way.

4 Proposals for Gender-Inclusive Tools for Teaching Logic

Given that the most positive results in my efforts to promote gender-inclusive practices in my logic teaching came from including more original texts by female authors among the set reading, my main proposal is to build reading lists, syllabuses, and textbooks around the ideas of women. This proposal divides into various smaller sub-proposals making practical suggestions on how to achieve this.

4.1 Building Syllabuses and Reading Lists Around Original Work by Female Authors

The current trend in teaching philosophical logic at the university level, turning away from using textbooks and towards assigning individual papers which students find online, should make it much easier to create more gender-balanced reading lists. This is more difficult in formal logic courses since they rely heavily on textbooks, very few of which are female-authored. Philosophy of logic textbooks are also mostly written by men, and philosophy of logic anthologies are generally hugely male-dominated too. But it is easy for reading lists to be updated to include more journal papers, book chapters and books by female authors. Another component of the strategy, though, must be to make the female authors' work central to the assignment. They must not be featured as an afterthought, nor as handmaidens who only comment on or expound the works of the great men. They should be presented as making interesting original contributions to the debate.

Some proposals for including more women in syllabuses are very firm that every reading list should be composed of 50% female authors. I think that, sadly, because logic and philosophy of logic have historically been such male-dominated fields, this is not always feasible in this particular area. But it should be realistic to aim for one-quarter or one-third female authors on the majority of topics in this area. It is also heartening that female students appear to feel much more included even with this much female representation. What is more important than fully proportional representation, it seems, from what I've seen, is that women are presented as serious experts in their fields on a par with the male giants.

4.2 Making Essay Questions More Female-Focused

The way essay questions are formulated, or introductory paragraphs on reading lists are written, is also a factor here. The questions and descriptions of the assigned works should present women as originators of serious views as well, and not gravitate only towards the views of the familiar male authors. I frequently used an essay structure where students were asked to compare two opposing views and adjudicate between them, where at least one of these points of view was a female author's.

Examples include:

- 'Compare and contrast Quine's critique of quantified modal logic with Barcan Marcus' defence of it.'
- 'Whose view of conditionals is more successful, Edgington's or Jackson's?'
- 'Does Grover's pro-sentential theory of truth eliminate the need for correspondence to a fact?'
- 'Should we be classical or non-classical logicians? In your answer, refer to Dummett and Haack.'

4.3 Creating Accessible Archives of Women's Writing in Formal and Philosophical Logic

Another very helpful development would be accessible, searchable archives or databases of papers by women for logic lecturers and tutors to refer to. Many instructors who are sympathetic to the goal of assigning more work by women simply do not know where to find such works, or are unsure where to start looking. This leads to them falling back on the familiar male, pale and stale authors.

Some efforts have been made to set up databases of writing by women, but generally the resources for them to be properly maintained are not there, and as a result they are not being added to. For instance, http://women.aap.org.au/papers/areas/logic.html only has one paper on logic or philosophy of logic by a woman. A new database of this sort would be very welcome. Alternatively, the editors of existing compendia of philosophical papers could perhaps be persuaded to add a feature that identifies papers by female authors.

5 Conclusion

In summary, most of my strategies for addressing hypothesised causes for male dominance in the logic classroom had some positive effect. I attempted to move students' thinking away from the stereotype that men have more of an aptitude for logic or mathematics, but without invoking stereotype threat, by emphasising effort over talent, and praising students for their efforts rather than for innate ability. There were some positive questionnaires as a result and several students reported that their results were above expectations (anecdotally). But the most noticeable positive effect came from assigning a good proportion of original research by female authors as set reading, with the majority of female students reporting they felt more included as a result. My proposal is therefore to build syllabuses, textbooks and essay questions more specifically around the ideas of women, e.g. Barcan Marcus on reference, modality, and quantification, Blanchette on models and consequence, Edgington on conditionals and knowability, Grover on truth, Haack on non-classical logic, Weiner on Frege.

Acknowledgements I would like to thank Shannon Dea and the audience at the *Inclusiveness* panel which she organised at the 2015 APA-Central meeting in February 2015. I am grateful to Sophia Connell, too, for her helpful feedback as a Director of Studies of several of the Cambridge colleges where I taught. Thanks are also due to Helen Beebee, Saba Fatima, Kristina Gehrman, and Suzanne Harvey for further discussion.

References

1 Barbara J. Bank. *Gender and Education: An Encyclopedia.* Greenwood, 2007.

2 Helen Beebee and Jennifer Saul. Women in philosophy in the UK (SWIP report), 2011. http://www.swipuk.org/notices/2011-09-08/.

3 Amanda Diekman and Mia Steinberg. Navigating social roles in pursuit of important goals: A communal goal congruity account of STEM pursuits. In *Social and Personality Psychology Compass*, pages 487–501. 2013.

4 Carol S. Dweck. Motivational processes affecting learning. *American Psychologist*, 41(10):1040–1048, 1986.

5 Carol S. Dweck. Is math a gift?: Beliefs that put females at risk. In S.J. Ceci and W. Williams, editors, *Why aren't more women in science? Top researchers debate the evidence*. American Psychological Association, Washington, DC, 2006.

6 Carol S. Dweck, William Davidson, Sharon Nelson, and Bradley Enna. Sex differences in learned helplessness: II. the contingencies of evaluative feedback in the classroom and III. an experimental analysis. *Developmental Psychology*, 14(3):268–278, 1978.

7 Dorothy Edgington. The paradox of knowability. *Mind*, 94:557–68, 1985.

8 Nicole M. Else-Quest, Janet Shibley Hyde, and Marcia C. Linn. Cross-national patterns of gender differences in mathematics: A meta-analysis. *Psychological Bulletin*, 136(1):103–127, 2010.

9 Programme for International Student Assessment. *PISA 2012 Results: What Students Know and Can Do: Student Performance in Mathematics, Reading and Science, Volume 1*. OECD, 2012.

10 Kristina Gehrman. Why 'I Think': Inclusive pedagogy, 2015. https://utk.academia.edu/KristinaGehrman.

11 Sally Haslanger. Changing the ideology and culture of philosophy: Not by reason (alone). *Hypatia*, 23(2):210–223, 2008.

12 Janet Shibley Hyde. The gender similarities hypothesis. *American psychologist*, 60(6):581–592, 2005.

13 Janet Shibley Hyde, Sara M. Lindberg, Marcia C. Linn, Amy B. Ellis, and Caroline C. Williams. Gender similarities characterize math performance. *Science*, 321(5888):494–495, 2008.

14 Sara M. Lindberg, Janet Shibley Hyde, Jennifer L. Petersen, and Marcia C. Linn. New trends in gender and mathematics performance: A meta-analysis. *Psychological Bulletin*, 136(6):1123–1135, 2010.

15 Matthew S. McGlone and Joshua Aronson. Forewarning and forearming stereotype-threatened students. *Communication Education*, 56:119–133, 2007.

16 Andrea Nye. *Words of Power: A Feminist Reading of the History of Logic*. Routledge, London, 1990.

17 University of Cambridge Philosophy Faculty. Reading list and course outline, part IB, Logic 2014-2015. http://www.phil.cam.ac.uk/curr-students/IB/IB-outlines-reading-lists/paper2-logic.

18 Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. Stereotype threat and women's math performance. *Journal of Experimental Social Psychology*, 35:4–28, 1999.

Syntax versus Semantics

Reinhard Kahle[1] and Wilfried Keller[2]

1 **CMA and DM, FCT, Universidade Nova de Lisboa**
P-2829-526 Caparica, Portugal
kahle@mat.uc.pt
2 **Universität des Saarlandes**
66123 Saarbrücken, Germany
wilfried.keller@uni-saarland.de

Abstract

We report on the idea to use colours to distinguish syntax and semantics as an educational tool in logic classes. This distinction gives also reason to reflect on some philosophical issues concerning semantics.

1998 ACM Subject Classification F.4.1, F.4.m

Keywords and phrases syntax, semantics, colour, philosophy, teaching

1 Introduction

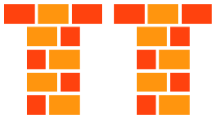
When the first author attended as a first year student a logic course at a philosophy department, he was wondering why the professor was repeating, after introducing the relation $\models$, "the same" again but using the symbol '$\vdash$'. It was, most likely, the negligence of the student, and not the presentation of the professor, which caused this fundamental misunderstanding. In retrospect, there is one question which might make one wonder: how was it possible to miss one of the most fundamental distinctions in modern logic? On a more mature stage, as third year student, the problem repeated itself on a different level: speaking about Gödel's first incompleteness theorem with a teaching assistant, he refused to continue the discussion when the question was raised what its meaning is from the perspective of "proper Mathematics". Again, the distinction of syntax and semantics wasn't clearly seen and the discussion stalled when the participants did not realize that reference of a term like "proper Mathematics" need to be better specified while entering into a discussion of Gödel's theorem.

Based on the experiences above, when starting to teach logic courses by himself, the first author took it as a particular challenge to present the problems concerning the distinction of syntax and semantics in a persuasive way. In this paper, we present the chosen solution: the use of colours to distinguish the syntactical and semantical role of logical text. We also report on some of the educational insights one might win with this approach together with certain philosophical questions which surface (again) in this context.

We assume that the reader is familiar with first-order logic and has, at least, an idea of Gödel's incompleteness theorems. Without formal introduction, we use the standard notations of first-order logic and Peano Arithmetic, using typically Greek letters from the end of the alphabet to denote formulae.

Acknowledgements. The first author was partially supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the projects *The Notion of Mathematical Proof* (PTDC/MHC-FIL/5363/2012) and UID/MAT/00297/2013 (Centro de Matemática e Aplicações). We thank the anonymous referees for helpful comments.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 75–84



Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

2 First-order logic

Let us go *in media res* and recall the definition of the satisfiability relation $\models$ for first-order logic as given by Barwise in [3, p. 21].

▶ **Definition 1.** Let $\mathfrak{M}$ be an L-structure. We define a relation

$$\mathfrak{M} \models \varphi[s],$$

(read: the assignment s satisfies the formula φ in $\mathfrak{M}$) for all assignments s and all formulae φ as follows.

(i) $\mathfrak{M} \models (t_1 = t_2)[s]$ iff $t_1^{\mathfrak{M}}(s) = t_2^{\mathfrak{M}}(s)$,
(ii) $\mathfrak{M} \models R(t_1, \ldots, t_n)[s]$, iff $(t_1^{\mathfrak{M}}(s), \ldots, t_n^{\mathfrak{M}}(s)) \in R^{\mathfrak{M}}$,
(iii) $\mathfrak{M} \models (\neg\varphi)[s]$ iff not $\mathfrak{M} \models \varphi[s]$,
(iv) $\mathfrak{M} \models (\varphi \wedge \psi)[s]$ iff $\mathfrak{M} \models \varphi[s]$ and $\mathfrak{M} \models \psi[s]$,
(v) $\mathfrak{M} \models (\varphi \vee \psi)[s]$ iff $\mathfrak{M} \models \varphi[s]$ or $\mathfrak{M} \models \psi[s]$,
(vi) $\mathfrak{M} \models (\varphi \to \psi)[s]$ iff either not $\mathfrak{M} \models \varphi[s]$ or else $\mathfrak{M} \models \psi[s]$,
(vii) $\mathfrak{M} \models (\exists v.\varphi)[s]$ iff there is an $a \in M$ such that $\mathfrak{M} \models \varphi[s(\frac{a}{v})]$,
(viii) $\mathfrak{M} \models (\forall v.\varphi)[s]$ iff for all $a \in M$, $\mathfrak{M} \models \varphi[s(\frac{a}{v})]$.

Barwise then states: "There is nothing surprising here. It is just making sure that each of our symbols means what we want it to mean."

This is certainly correct, but it might be worth to reflect shortly what this definition actually does.[1] Looking, for instance, to the clauses for conjunction and disjunction, it simply "lifts" the symbols '$\wedge$' and '$\vee$' to the *meta-level* using the natural language expressions "and" and "or". Thus, who understands "and" and "or" is supposed to understand how to obtain the truth value of a formula using '$\wedge$' or '$\vee$'. The *meta-level* comes into play, because the natural language terms are outside of the $\models$ relation, while the logical symbols are inside. Nothing surprising here, as said. We will see later in § 6 that the situation for the negation is actually more subtle than it looks (and this is a first educational pitfall!). But the issue we like to start with is unambiguously expressed by Barwise (in continuation of the citation above):

> There is one possibly confusing point, in (i), caused by our using $=$ for both the real equality (on the right-hand side) and the symbol for equality (on the left). Many authors abhor this confusion of use and mention and use something like $\equiv$ or $\approx$ for the symbol.

It is this confusion[2], which will repeat itself when one comes to the language of arithmetic, we like to address. As Barwise remarks, many authors try to resolve the problem by the use of different symbols for syntactical and semantical equality. This would be a way out, but it comes at the price that the formal language appears artificial, in particular, if this

[1] Here a disclaimer is in order. In the following we don't intend to criticise Barwise's presentation; first, the citation is taken from a handbook article and not from a textbook (and we may add: from an excellent article in an excellent handbook); secondly, mathematically there is no issue here, the definition is surely the correct one; we are just looking for aspects which are of interest in an educational perspective.

[2] And it might be doubted, whether the difficulty at hand is best couched in terms of use and mention. It seems to us rather clear, however, that it is educationally not best dealt with by introducing subtleties of quotation theory.

is extended to arithmetical terms. Let us mention the clumsy notation $\mathbf{k}_n$ for the numeral representing the number n in the classical textbook of Shoenfield [15]. With respect to equality, Sernadas and Sernadas [14], for instance, use $\cong$ as special sign on the syntactical side. Cori and Lascar [5] use, quite generally, a bar over the syntactical symbol to refer to its *semantic* interpretation. However, in general (and with good reasons), the syntactic symbols are chosen to match exactly with their intended meaning; the resulting "overloading" is just a problem when the difference of syntax and semantics is the subject under discussion.

Our solution is to use the same symbols, but different colours to distinguish the use of symbols on the syntactical and semantical side. So, let us choose red for syntax and blue for semantics.[3] After giving a definition of first-order language with all syntactic expressions in red, and the definition of a structure with blue for the elements of the structure, the definition above reads as follows:

▶ **Definition 2.** Let $\mathfrak{M}$ be an L-structure. We define a relation

$$\mathfrak{M} \models \varphi[s],$$

(read: the assignment s satisfies the formula φ in $\mathfrak{M}$) for all assignments s and all formulae φ as follows.[4]

(i) $\mathfrak{M} \models (t_1 = t_2)[s]$ iff ${t_1}^{\mathfrak{M}}(s) = {t_2}^{\mathfrak{M}}(s)$,
(ii) $\mathfrak{M} \models R(t_1, \ldots, t_n)[s]$, iff $({t_1}^{\mathfrak{M}}(s), \ldots, {t_n}^{\mathfrak{M}}(s)) \in R^{\mathfrak{M}}$,
(iii) $\mathfrak{M} \models (\neg\varphi)[s]$ iff not $\mathfrak{M} \models \varphi[s]$,
(iv) $\mathfrak{M} \models (\varphi \wedge \psi)[s]$ iff $\mathfrak{M} \models \varphi[s]$ and $\mathfrak{M} \models \psi[s]$,
(v) $\mathfrak{M} \models (\varphi \vee \psi)[s]$ iff $\mathfrak{M} \models \varphi[s]$ or $\mathfrak{M} \models \psi[s]$,
(vi) $\mathfrak{M} \models (\varphi \to \psi)[s]$ iff either not $\mathfrak{M} \models \varphi[s]$ or else $\mathfrak{M} \models \psi[s]$,
(vii) $\mathfrak{M} \models (\exists v.\varphi)[s]$ iff there is an $a \in M$ such that $\mathfrak{M} \models \varphi[s(^a_v)]$,
(viii) $\mathfrak{M} \models (\forall v.\varphi)[s]$ iff for all $a \in M$, $\mathfrak{M} \models \varphi[s(^a_v)]$.

Now, the difference of $=$ and $=$ is obvious; the "possible confusion" will even catch the student's eye, and this is of particular educational value. In addition, the quantifier clauses allow to distinguish better between the syntactical object variable v and the semantical *element* a—even if a is here, of course, a meta-variable for "real elements" of M.

Introducing now, as counterpart to the semantic consequence relation, a derivation relation $\vdash$ which exclusively, deals with red objects, syntactic and semantic reasoning is already distinguished by colours. It is our teaching experience, that these colours are of great help for the students, providing some kind of orientation in rather technical proofs like these for the compactness and the completeness theorem. As prime examples for the use of colours we may also mention the Skolem paradox, which, like the completeness theorem, can be regarded as a corollary of the compactness theorem.

And so the question of soundness and completeness is the next one to be addressed. Even in a compact course on Gödel's incompleteness theorems these results of Gödel's dissertation

[3] The particular choice of the colours is arbitrary; it is, however, known that one should avoid yellow and green for beamer presentations.
[4] For sure, as one referee pointed put, colouring these expressions does not remove entirely the possibility of misunderstanding: The functional expression ${t_1}^{\mathfrak{M}}$, for instance, takes the syntactic argument t_1 and evaluates it relative to the semantic model $\mathfrak{M}$—so the value of the whole expression of course denotes a blue item. The interpretation function itself does not appear in this notation, and therefore, luckily, we do not have to decide on its colour. However, with interested students we had some interesting discussions about its colour.

should be discussed.[5] Today, they are usually not proven along the lines of Gödel's original work, but rather with the well-known technique introduced by Henkin. This technique is standard for many completeness proofs for quite a couple of logics and, therefore, constitute an integral tool in the toolbox of the mathematical logician: it proceeds by constructing a blue world (validating the negation of a formula, that cannot be proven in the syntactic calculus) out of the red material itself. (The situation is somewhat dual to the more demanding case of (set-theoretic) forcing, where one, so to speak, bestows properties of a language on certain blue sets, thereby emulating the red world with originally blue objects.)

3 Arithmetic

The use of the colours exhibits its real potential only, when we study Arithmetic and want to motivate, for instance, Gödel's incompleteness theorems for Peano Arithmetic. In its original form, Gödel's proof actually entirely refrains from the semantic realm and in some sense requires only "red" reasoning. It, indeed, is a theorem about the red world, but one that operates on a meta-level. It speaks about the axiomatic system, but it does not reason about its meaning, quite like Hilbert's attempted consistency proofs do. And likewise it uses only finitistically acceptable means. In [6, fn. 1, p. 337] the editors explicitly attribute Gödel's avoiding of the concept of truth in his 1931 paper to the "respect for the 'prejudice' of the Hilbert school." Further reflections can be found in [7].

But, of course, the first incompleteness theorem can—and probably should—be motivated by the question whether Peano Arithmetic is complete with respect to the *standard model*. Peano Arithmetic may be defined over the set $\{0, S, +, \cdot\}$ of non-logical symbols. The standard model, as semantic object, can be given in blue as follows: $\mathfrak{N} = \langle \mathbb{N}, 0, S, +, \cdot \rangle$. It goes without saying that the red symbols should be interpreted by their respective blue counterparts. Gödel's first incompleteness theorem tells us, that the red world, as long as a consistent recursive axiomatic system is chosen, will miss some formula *and* its negation[6], it cannot decide all given formulae by proof. However, the blue world so to speak decides formulae: Either a formula or its negation is satisfied (in a model by a valuation); this is a direct consequence of clause (iii) in Def. 1 (see also §6 below). There is no gap in the blue world—but there will always be one in the (sensibly chosen, i.e., recursive and consistent) red one. This then not only entails that no red world (as above) fixes the blue world of Arithmetic as its only model—this could already be concluded from Löwenheim–Skolem—but that not all red statements that are true about the blue world can be proven in the red realm.

So, *how* are the blue objects in $\mathfrak{N} = \langle \mathbb{N}, 0, S, +, \cdot \rangle$ actually given? $\mathbb{N}$ as the set of natural numbers $\{0, 1, 2, \dots\}$ can be taken for granted, and with it, of course, also the object 0. But what *is* $+$? It should be, of course, the set of all triples (a, b, c) such that $a + b = c$. But this doesn't help us, as we just moved the question to another level, writing $+$ in black (this plus sign can, of course, not be blue if we don't want to run into an immediate circle). To avoid

[5] In introductory logic courses in philosophy seminars these results will usually not be proven, but, of course, deserve at least to be mentioned and discussed.

[6] Of course on pain of inconsistency it should *not* prove any formula together with its negation.

it, one could get back to use of dots and say that $+$ is the set:

$$\{(0,0,0),(0,1,1),(0,2,2),(0,3,3),\ldots$$
$$(1,0,1),(1,1,2),(1,2,3),(1,3,4),\ldots$$
$$(2,0,2),(2,1,3),(2,2,4),(2,3,5),\ldots$$
$$\vdots \qquad\qquad \}.$$

In fact, we see only two ways out of this situation: First, we appeal to the common platonism in mathematics and stipulate that $+$ exists in the structure $\mathfrak{N}$ in the way we expect it and with the properties we want it to have. Or, secondly, we treat $+$ as a proper set-theoretic object and construct it within our favorite axiomatic set theory, let's say ZFC.

While the second one is a solution concerning the level of Arithmetic, it simply shifts the problem to set theory. Within ZFC, the defined $+$ would be a red object (relative to ZFC), and we would have to ask now, how the models of ZFC look like—which is, not surprising, a much harder question than to look for structures for Arithmetic, not only in a technical sense, but also in a philosophical one; in particular, it leads us simply back to the situation we are discussing for $+$ just for other, now set-theoretical, objects.

For the first option, the sheer existence of $+$ presupposes a certain form of platonism. This platonism allows to dispose of any further technical questions, but one enters a dangerous philosophical terrain.

We may leave aside here the preferred way out (this is a philosophical question—but one which is naturally asked in this context and can nicely be brought out and talked about with the help of our colours); what we consider as an important lesson for the teaching of logic is, that any discussion of the syntax/semantics distinction in the context of Arithmetic—and, *a fortiori*, in relation to Gödel's Incompleteness Theorems—has to presuppose some "blue objects". And this presupposition is far from being trivial. As a minimal conclusion, we like to state that an unreflected "identification" of, for instance, the "syntactical addition" (symbol) $+$ with the "semantical addition" (function) $+$ prevents a student from any understanding of the problem which is at issue here.

4 Gödel's Incompleteness Theorems

Gödel's Incompleteness Theorems are, without doubt, the most important results in mathematical logic. They are, however, clouded by continuing philosophical misunderstandings,[7] and a pure formal presentation of the results can easily support such prejudices.[8]

We like to address only one problem coming from the fact that many presentations make use of the *Chinese remainder theorem*. Apparently, here proper mathematics is used to prove something about formal systems for mathematics and one may wonder whether there is some kind of vicious circle in the back.[9] This is, of course, not the case; in fact, the use of this theorem is only relevant for a certain technical step, in particular in the case of Peano

[7] For a detailed discussion, in particular in view of Gentzen's later results, we refer to [10].

[8] An extreme case was reported from a philosophical seminar: the participants, after studying carefully—and probably painfully—a formal presentation of Gödel's proof, came to the conclusion that, yes, after all the results seem to be correct *if one has a language with 7 symbols; but, of course, one doesn't know whether it still holds when one would have 8 symbols.*

[9] The first author remembers an elaboration of a Math student of a course presentation of Gödel's theorems which focussed nearly entirely on the use of the Chines remainder theorem; apparently it was the only part of the proof the student was properly understanding based on his Math classes.

Arithmetic, and it is irrelevant for other axiomatization or codings (see [16, §3.2.6]). Hao Wang reported this from interviews with Gödel [17, p. 653]: "He enjoyed much the lectures by Furtw[ä]ngler on number theory and developed an interest in this subject which was, for example, relevant to his application of the Chinese remainder theorem in expressing primitive recursive functions in terms of addition and multiplication." Thus, what is at issue here is the representation of primitive recursive functions in the formal arithmetical theory. Albeit, it is, of course, possible to develop primitive recursion *in* such theories, a proper educational approach *can* "outsource" primitive recursion.

We did that by setting up a new realm for primitive recursion (formally a functional algebra, consisting of function symbols and equalities)—and using a new colour for it, as, for the moment, it can be located outside of the red and blue world. After providing a sufficient stock of primitive recursive functions, one simply proves the representation theorem showing that every such function can be represented in the formal theory under consideration (as Peano Arithmetic, for instance). In the particular case of Peano Arithmetic, this proof may use the Chinese remainder theorem—but it is obvious that it is not relevant if one would provide another proof, in particular for other theories. In addition, it should become clear that Gödel's theorems can be carried out in essentially the same way for all theories allowing for the appropriate representation theorem.

With respect to the Chinese Remainder Theorem and the like the diagnosis is as following: The Chinese Remainder Theorem is—as far as Gödel's first incompleteness theorem is concerned—at the level of meta theory, but not necessarily in the blue world. It deals with the recursion-theoretic realm, concerning codings, and can be outsourced and imported if needed.

The situation changes however, if one wants to proceed to the second Incompleteness Theorem: There one repeats the proof of the first completely inside the red theory. For that case it is of vital importance that one uses codings and theorems that can be reproduced in the object theory itself—this is the sense in which it is "occasionally not just important [in logic] what you prove, but how you prove it" [12, p. 190].

5 "Semantics comes first"?

The dictum "Semantics comes first" is attributed to Tarski.[10] And it has some rationale: in first-order logic as well as in Arithmetic, we presuppose the *meanings* of the logical and arithmetical expressions before we start to manipulate them. It is, in general, seen as a task of the axiomatization to provide calculi which capture such meaning—and the completeness theorems should show us that the axiomatization was successful.

As much as completeness theorems are concerned, they give us—*a posteriori*—the possibility to identify the red with the blue objects. But, to make sense out of a completeness theorem—and to prove it—the difference of the syntactical and semantical expression has to be seen clearly. In this perspective, we could even propose that one could forget of the red and blue colour for, let say, equality, *after* the proof of the completeness theorem for first-order logic.

But there is still an issues to discuss which might call Tarski's dictum in question.

First, the dictum makes sense only, if the semantics is *understood before* the syntax. It is doubtful that semantics could serve its purpose if it is, by itself, not fully understood. For

[10] The well-known computer software package *Tarski's World* [2] illustrates this slogan quite appropriately when the student can literally manipulate the semantical objects.

first-order logic and Arithmetic it should be the case that their semantics is clear (although we mentioned the problem to precisely articulate it in the case of $+$, for instance). And this holds probably also for most of the mathematical theories we are familiar with.

The situation changed drastically, when Computer Science entered the stage. Semantics of programming languages is a rather challenging topic in CS. And here, the syntax is essentially always prior to the semantics. Instead of finding an axiomatization for a given semantics, now one looks for a semantics for a given programming language. But it is not only that such a semantics is no longer "first", it might be the case that proposed semantics are, in a technical sense, more complicated than the programming language in itself (as example, we may refer to [1]). In some cases, one might wonder in which sense such semantics provide any *meaning*.[11]

From the educational perspective, this only means that logic cannot any longer be taught with Tarski's dictum as a rock solid starting point. The differentiation of syntax and semantics becomes even more significant. What it means to put "syntax first" we will discuss in the next section.

6 Intuitionism

When Brouwer conceived his mathematical philosophy of *intuitionism* he had definitely nothing formal in mind. However, based on the "successful" axiomatization of intuitionistic logic by Heyting, today, intuitionism is often presented in an axiomatic context; for propositional logic, for instance, just like a calculus for classical logic, but without *tertium-non-datur*. In this way, it can be recast totally in our red world. An "additional" semantics, may it be informal or set-theoretically, would be alien to intuitionism.[12] One could go a step further and say—instead of that there is no blue world—that the red world should be its own blue world. This is often paraphrased by saying that, from an intuitionistic perspective, the meaning of a formula is given by its proof (better: the set of all its proofs). Today, this idea is reflected in *proof-theoretic semantics*, an approach which "attempts to locate the meaning of propositions and logical connectives not in terms of interpretations . . . but in the role that the proposition or logical connective plays within the system of inference."[13]

In view of our discussion above, the formal framework of intuitionism undermines substantially Tarski's dictum. In particular, when we go back to Definition 1 of a L-structure, one can easily observe that clause (iii) for negation *builds in* classical logic into any structure.

It is worth to reflect a little bit more on this clause; it looks as innocent as any other clause, that for conjunction and disjunction, for instance, "just making sure that each of our symbols means what we want it to mean.", to repeat the citation from Barwise [3, p. 21]. But it is not only the case, that Brouwer would not agree that this is what we want negation to mean; it is the problem that the right-hand side—not $\mathfrak{M} \models \varphi[s]$—can, in general, not practically be verified. It is the full purpose of an axiomatization to give a *positive* approach to negation, i.e., the symbol for negation $\neg$ has to stay always on the right-hand side of the

[11] A colleague expressed this in a sarcastic statement: "They put these brackets $[\![\cdot]\!]$ around something and believe they have achieved something."

[12] The first author remembers a remark of an author to the effect that intuitionism was not succeeding due to the lack of a proper semantics, until this deficit was cured by the introduction of Kripke semantics. In our view, this sounds like claiming that a protestant church is not a church as long as it doesn't have a pope.

[13] `http://en.wikipedia.org/wiki/Proof-theoretic_semantics`, accessed Febr. 20, 2015. For more on proof-theoretic semantics, see [11, 13].

derivation sign $\vdash$, and we must not make use of a "meta-negation" $\nvdash$. That this is possible is far from being trivial—the complexity is visible in the proof of Gödel's completeness theorem—but it makes clear the fundamental difference between syntax (in form of an axiomatization, providing an—executable, albeit not decidable—derivation relation) and semantics, which is, in general, inherently non-constructive.

As much as it comes to Brouwer's criticism of the usual semantic understanding of negation, it is worth to cite Bernays [4, p. 4 (our translation)], who replied to it as follows:

> As one knows, the use of the "tertium-non-datur" in relation to infinite sets, in particular in Arithmetic, was disputed by L. E. J. Brouwer, namely in the form of an opposition to the traditional logical principle of the excluded middle. Against this opposition it is to say that it is just based on a reinterpretation of the negation. Brouwer avoids the usual negation non-A, and takes instead "A is absurd". It is then obvious that the general alternative "Every sentence A is true or absurd" is not justified.

Following Bernays, the controversy about classical and intuitionistic negation could even be boiled down to a question of words. What is of importance for us here, is that the semantic determination of (classical) negation closes the road to understand Brouwer's (intuitionistic) negation. To say it differently: the "standard approach" of teaching logic based on Tarski's dictum (used for Tarskian semantics as given in Def. 1) blocks a proper transition to intuitionistic logic, at least, if the students are "semantically biased". As a matter of fact, intuitionistic logic can—and maybe: has to—be developed purely in the red world.[14] It is not our aim to motivate an approach that exclusively focusses on intuitionism or the like.[15] But it is important that an introduction to classical logic should not come with a semantical bias which blocks the road to non-classical logic from the start.[16] Here the colours may help.

7 Conclusion

It is our experience from courses on mathematical logic—regular university classes, block courses, and summer school courses—that the use of colours gives the students, indeed, a "tool" at hand to see better the fundamental distinction of syntax and semantics.[17] Although the use of colours requires some additional effort in the preparation of the student material, it is a rather inexpensive investment to obtain significant educational added value. At a certain stage—for instance, as mentioned, after the proof of the completeness theorem—, when the student has already internalized the difference of syntax and semantics one can even easily go back to the common identification of the syntactical and semantical expression using them just in black; it provides even a good test template, as one can ask the students in all kind of instances whether a certain expression would have to be coloured red or blue.

[14] As said at the beginning, this concerns intuitionistic logic based on Heyting's axiomatization; Brouwer's original intuitionism, in particular if considered in the mathematical area of Analysis, should not come in a formal livery at all.

[15] It is worth mentioning that *dialogical logic* has a rather interesting status concerning the syntax and semantics distinction, which doesn't seem to be properly explored yet, see Kahle [8, 9].

[16] For instance, substructural logics are best approached via proof theory, because—quite like intuitionism—there lies their motivation; on the other hand their semantics usually turn out to be rather complicated.

[17] We have, of course, not developed an empirical study to "prove" that the use of colours improves the teaching; it is rather an informed impression backed by feedback from our students. And although we do not have statistically valid data comparing the effects of this method with other approaches, the test results of the courses were very positive (but we had few, but rather good and understanding students).

In addition, while the colouring can, of course, not solve any philosophical problem concerning syntax and semantics, it helps to sharpen the sensibility for the underlying problems. In any case, the success of a logic course will not only depend on the use of tools, but on the interaction between teacher, tutor(s) and students. The use of a new dimension—like colour—underlines the importance of the distinction between syntax and semantics, it brings various (philosophical) topics into focus and gives a convenient way of talking about these issues—a way that often is readily accepted by the students.

The message of this paper is not: By all means use colours to distinguish syntax from semantics! But rather: Distinguish syntax and semantics clearly and discuss some of the issues related to that distinction—the use of colour will help you doing so!

References

1 Th. Altenkirch, P. Morris, F. Nordvall Forsberg, and A. Setzer. A categorical semantics for inductive-inductive definitions. In A. Corradini and B. Klin, editors, *Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 70–84. Springer, 2011.

2 D. Barker-Plummer, J. Barwise, and J. Etchemendy. *Tarski's World*, volume 169 of *CSLI Lecture Notes*. CSLI Publications, 2008.

3 J. Barwise. An introduction to first-order logic. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 5–46. North-Holland, 1977.

4 P. Bernays. Bemerkungen zu Lorenzen's Stellungnahme in der Philosophie der Mathematik. In K. Lorenz, editor, *Konstruktionen versus Positionen*, volume 1, pages 3–16. Berlin, 1979.

5 R. Cori and D. Lascar. *Mathematical Logic, Part I*. Oxford University Press, 2000.

6 B. Buldt et al., editor. *Kurt Gödel. Wahrheit und Beweisbarkeit*, volume 2: Kompendium zum Werk. öbv & hpt, 2002.

7 S. Feferman. Kurt Gödel: Conviction and Caution. *Philosophia Naturalis*, 21:546–562, 1984.

8 R. Kahle. Konstuktivismus und Semantik. In J. Mittelstraß, editor, *Der Konstruktivismus im Ausgang der Philosophie von Wilhelm Kamlah und Paul Lorenzen*, pages 197–212. Mentis, 2008.

9 R. Kahle. Dialoge als Semantik. In J. Mittelstraß and Chr. von Bülow, editors, *Dialogische Logik*, pages 43–54. Mentis, 2015.

10 R. Kahle. Gentzen's consistency proof in context. In R. Kahle and M. Rathjen, editors, *Gentzen's Centenary*. Springer, 2015, to appear.

11 R. Kahle and P. Schroeder-Heister, editors. *Proof-theoretic semantics*, volume 148. 2006.

12 W. Rautenberg. *A Concise Introduction to Mathematical Logic*. Springer, 2006.

13 P. Schroeder-Heister. Proof-theoretic semantics. In E.Ñ. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2014 edition, 2014.

14 A. Sernadas and C. Sernadas. *Foundations of Logic and Theory of Computation*, volume 10 of *Texts in Computer Science*. College Publication, 2008.

15 J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, MA, 1967. Reprinted: Association for Symbolic Logic and AK Peters, 2001.

16 C. Smorynski. The incompleteness theorems. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 821–865. North-Holland, 1977.

17 H. Wang. Some facts about Kurt Gödel. *The Journal of Symbolic Logic*, 46:653–659, 1981.

Using Automated Theorem Provers to Teach Knowledge Representation in First-Order Logic

Angelo Kyrilov and David C. Noelle

University of California, Merced
5200 North Lake Road, Merced, CA, 95343, USA
{akyrilov, dnoelle}@ucmerced.edu

Abstract

Undergraduate students of artificial intelligence often struggle with representing knowledge as logical sentences. This is a skill that seems to require extensive practice to obtain, suggesting a teaching strategy that involves the assignment of numerous exercises involving the formulation of some bit of knowledge, communicated using a natural language such as English, as a sentence in some logic. The number of such exercises needed to master this skill is far too large to allow typical artificial intelligence course teaching teams to provide prompt feedback on student efforts. Thus, an automated assessment system for such exercises is needed to ensure that students receive an adequate amount of practice, with the rapid delivery of feedback allowing students to identify errors in their understanding and correct them. This paper describes an automated grading system for knowledge representation exercises using first-order logic. A resolution theorem prover, *Prover9*, is used to check if a student-submitted formula is logically equivalent to a solution provided by the instructor. This system has been used by students enrolled in undergraduate artificial intelligence classes for several years. Use of this teaching tool resulted in a statistically significant improvement on first-order logic knowledge representation questions appearing on the course final examination. This article explains how this system works, provides an analysis of changes in student learning outcomes, and explores potential enhancements of this system, including the possibility of providing rich formative feedback by replacing the resolution theorem prover with a tableaux-based method.

1998 ACM Subject Classification K.3 Computers and Education

Keywords and phrases Automated Assessment, Knowledge Representation, Automated Theorem Provers

1 Introduction

Undergraduate computer science curricula often provide students with opportunities to study artificial intelligence (AI). Courses on AI frequently cover the development of intelligent systems by constructing knowledge bases composed of logical sentences and performing automated reasoning over those sentences. Computer science students regularly have little background in formal logics before attending an AI course, and this makes the learning of logic-based knowledge representation schemes particularly challenging [1]. In the Computer Science and Engineering program at the University of California, Merced, the "Introduction to Artificial Intelligence" class provides a broad survey of AI methods and topics, including the construction of automated reasoning systems using first-order logic to represent knowledge. This is an upper-division semester-long undergraduate course which is taught annually. Historically, students enrolled in this class have found knowledge representation to be a particularly difficult topic. When asked to translate English sentences into first-order logic, using a specified ontology, as part of a written final examination, their performance has been

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 85–92


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

extremely poor. Students only score about 30% of the maximum possible credit, on average, when presented with exam questions of this kind.

These low scores are likely the result of a lack of adequate practice with first-order logic. The broad array of material covered in this survey course limits the amount of lecture time available to illustrate the construction of logical formulae, and high enrollments limit the amount of guidance and feedback each student can expect to receive from the teaching team. While student understanding would certainly benefit from extensive practice on knowledge representation exercises, the grading of such exercises is demanding, as there are often many equally correct ways to express a proposition in first-order logic. Thus, given that students require feedback on practice exercises for them to be useful, the number of exercises that could be assigned has been highly restricted by limited human resources.

In order to address this problem, we built an online repository of exercises involving the translation of English sentences into first-order logic, and we designed and implemented an online software tool to automatically assess student solutions to these exercises. By using this tool, students received instant feedback in the form of "Correct/Incorrect" judgments, and students who submitted incorrect solutions were allowed to revise and resubmit their answers. There was no limit on the number of resubmissions permitted.

This online educational system was used in our "Introduction to Artificial Intelligence" course during the 2012, 2013, and 2014 offerings. We analyzed student performance on final examination knowledge representation questions, and we compared it to the performance of students from previous years, who had no access to our system. We found that students who used our system exhibited significantly improved scores on the first-order logic knowledge representation questions.

The rest of this paper is organized as follows. Section 2 describes the automated grading system in detail. Section 3 provides an assessment of the system in terms of its contribution to student learning outcomes. Section 4 offers ideas for improving the automated grading software by utilizing semantic tableaux methods instead of resolution theorem proving for the production of exercise feedback. Section 5 contains some concluding remarks.

2 System Description

Our goal was to give students much more practice on knowledge representation exercises. We generated a repository of questions in which students were given an English sentence and were asked to translate it into first-order logic. Each question supplied an explicit list of predicates, functions, and constant symbols that students were allowed to use in their answers. A typical example would be:

> Translate the sentence "*All surgeons are doctors*", using the following constants: $Doctor$, $Surgeon$, and predicates: $Occupation(x, y)$.

A correct solution to this exercise is the formula:

$$\forall x\ Occupation(x, Surgeon) \Rightarrow Occupation(x, Doctor)$$

It is important to note that there are usually multiple correct solutions to exercises of this kind. For example, another correct answer to the question, above, is:

$$\neg(\exists x\ Occupation(x, Surgeon) \wedge \neg Occupation(x, Doctor))$$

Thus, student submissions could not be assessed by performing a simple string comparison, or the like, with a correct solution provided by the instructor.

Our system does require the instructor to provide a model answer for each exercise, but it does *not* necessarily label submissions that deviate from this model answer as incorrect. Instead, any submitted formula that is found to be logically equivalent to the model answer is recognized as a correct solution to the exercise. We use the *Prover9* automated theorem prover to check for logical equivalence. If A is the model answer and B is the student solution, the solution is labeled as correct if and only if the formula $A \Leftrightarrow B$ is found to be valid. Prover9 is a resolution based automated theorem prover for first-order logic with equality [4]. Prover9 was selected because it is very easy to use and the syntax of its interface is very similar to what students see in lectures.

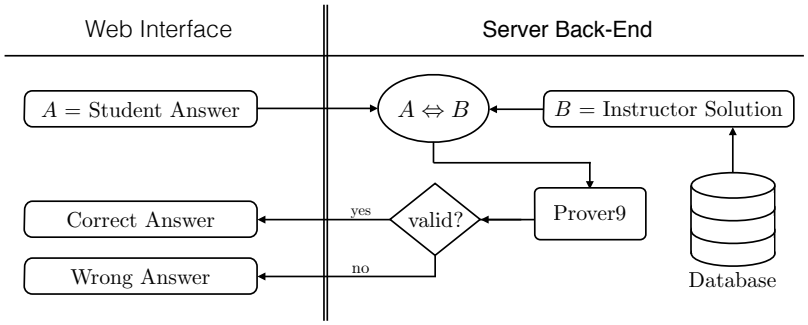


Figure 1 Components of the Automated Grading System

Figure 1 illustrates the automated grading system components, including the web interface and the back-end. When a student submits a solution to an exercise, the model answer is retrieved from the exercise database. Prover9 is used to determine whether the student's solution is logically equivalent to the model answer, and appropriate feedback is immediately sent to the student.

There is a restriction on the amount of time the server is allowed to spend on checking a student's submission. By default, this is set to 5 seconds but it can be adjusted on a per exercise basis. If the time limit is exceeded, an appropriate message is sent to the student informing them that the time limit has been exceeded. While this does not necessarily indicate that the student's answer is incorrect, students are encouraged to revise their solution or talk to an instructor. This takes care of the fact that the prover may run forever due to the undecidability of first-order logic.

Figure 2 shows the user-interface of the system, which appears in a web browser window. In addition to the question listing, which is what students would see, there is also an administrative interface, allowing instructors to create and assign exercises.

3 System Evaluation

The final examination for the "Introduction to Artificial Intelligence" course is a three hour comprehensive written test that covers the full range of AI topics presented during the semester long class. Students complete the exam without access to any textbooks, notes, or other study materials. The final examination contains three questions involving the translation of English sentences into first-order logic, as well as a question asking students to produce a successor-state axiom for a given time-varying predicate [5]. If the extensive practice afforded by our automated grading system is a benefit to student learning, then

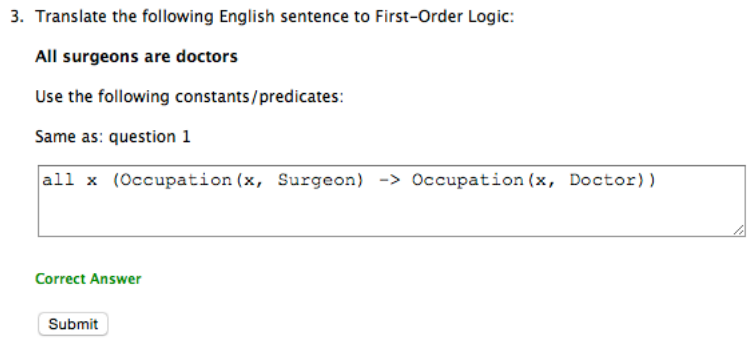


Figure 2 The student user-interface of the automated grading system

we would expect to see higher scores on these particular final examination questions when students made use of our system.

In order to evaluate our system, we collected scores on these four questions over multiple offerings of the "Introduction to Artificial Intelligence" course. Scores collected for offerings in 2007, 2008, 2010, and 2011 were produced by students who had no access to our system, as it had not yet been created. The students from these offerings acted as a control group. The automated grading system was used during offerings in 2012, 2013, and 2014, making the students enrolled during these years members of a test group. There were 113 students in the control group and 169 in the test group. The mean performance of students, as measured by the sum of scores received on all four of the relevant questions (24 points possible), is displayed in Figure 3.

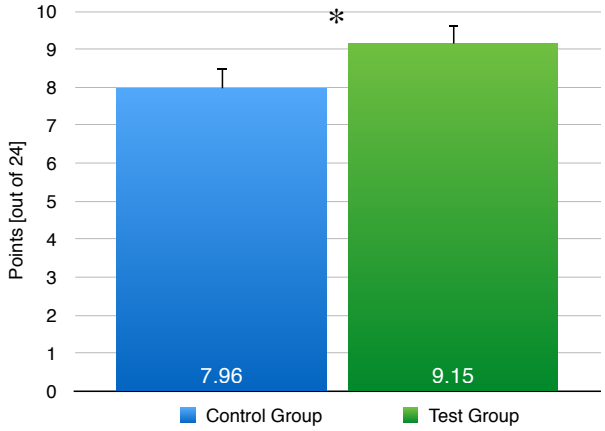


Figure 3 Mean over students of the sum of scores on all of the relevant questions. A maximum of 24 points could be earned. Error bars display one standard error of the mean. The asterisk ($*$) indicates that the difference in mean scores is statistically significant at the $\alpha = 0.10$ level.

We performed a standard analysis of variance (ANOVA) of these data, using group and question as factors. This analysis revealed a marginally significant effect of group membership, with the group making use of our automated grading system receiving higher aggregate scores ($F(1, 280) = 96.5$; $p = 0.066$). We also conducted planned two-tailed t-tests for each of

the four relevant final examination questions, assessing the impact of our automated grading system on student performance on each question type.

The first question involved a simple translation of an English sentence into first-order logic. For example, students might be asked to translate the sentence: "A block can never be on top of another block that is smaller than it." For this final examination question, we found a marginally significant benefit of use of our system ($t(280) = 1.929$; $p = 0.055$).

The second question addressed the representation of uniqueness. An example sentence would be: "There is exactly one block that is smaller than all of the others." Use of our system did not reliably influence performance on this question ($t(280) = 0.304$; $p = 0.761$).

The third question asked students to provide a definition for a predicate. Often, the question demanded the formulation of a recursive definition. For example, students might be asked to provide a definition for a simple blocks-world predicate like $Above(x, y)$ when given a predicate like $On(x, y)$. A sample solution would be:

$$\forall x \forall y \; Above(x, y) \Leftrightarrow On(x, y) \vee (\exists z \; On(x, z) \wedge Above(z, y))$$

Use of our automated grading system produced a reliable increase in scores for this question ($t(280) = 2.077$; $p = 0.039$).

Finally, the fourth question required students to write a successor-state axiom for a given fluent using the situation calculus [5]. Completing practice exercises using our system had no detectable impact on scores for this question ($t(280) = 0.856$; $p = 0.393$). The mean scores for each question are shown in Figure 4.

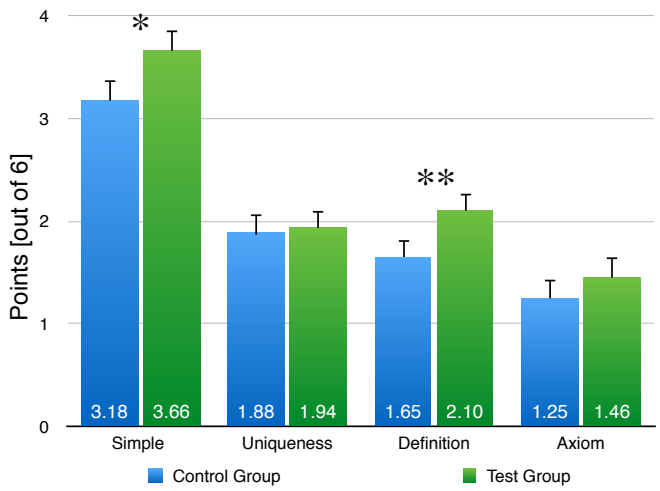


Figure 4 Mean scores for each question type. The maximum possible score for each question was 6 points. Error bars display one standard error of the mean. An asterisk ($*$) indicates that the difference in mean scores is statistically significant at the $\alpha = 0.10$ level, and a double asterisk ($**$) marks significance at the $\alpha = 0.05$ level.

It is worth noting that most of the exercises presented by our online system were similar to the first examination question, described above. A small number of exercises dealt with uniqueness, and there were no definition or successor-state axiom questions in the system. (Examples of definition sentences and successor-state axioms were discussed during class lectures, but the automated grading system offered no additional practice on these kinds of questions.) This observation suggests that practice on the first kind of question actually transferred to definition questions.

4 Future Work

We are currently investigating a specific technical enhancement of our automated grading system to allow for the delivery of more rich feedback than a binary "Correct/Incorrect" assessment. This enhancement involves replacing the resolution theorem prover that is used to check for a logical equivalence between a submission and the instructor's model answer with a tableaux-based prover [2]. The utility of this modification stems from the fact that tableau procedures produce a counter model when the provided sentence is not valid. Such a counter model could provide useful guidance to students to help them determine the source of their misunderstandings. Consider the following example.

Translate the sentence "*Joe does not have a lawyer*", given the following:

Joe	A person named Joe
$Lawyer$	An occupation of being a Lawyer
$Occupation(x, y)$	person x has occupation y
$Customer(x, y)$	person x is a customer of person y

One correct solution to the above exercise, call it A, is:

$$\neg(\exists x\ Occupation(x, Lawyer) \wedge Customer(Joe, x))$$

... and a typical incorrect solution generated by students, call it S, is ...

$$\exists x\ Occupation(x, Lawyer) \wedge \neg Customer(Joe, x)$$

Formula S states that, "There is at least one lawyer that Joe is not a customer of." This sentence has a different meaning than that communicated by the sentence that students were asked to translate. Testing $A \Leftrightarrow S$ for validity using the *ProofTools* [6] tableau prover produces the following counter model:

$$Occupation(B, Lawyer),\ \neg Customer(Joe, B),$$

$$Occupation(C, Lawyer),\ Customer(Joe, C)$$

This counter model is similar to what a human instructor may point out to a student who has submitted S as the solution for the above exercise. ("Can you see how your solution would be true if there was one lawyer, B, who was not hired by Joe, but there was another lawyer, C, who Joe retained? But the original English sentence states that Joe has no lawyer, at all.")

Another example is: *"Joe is an actor but he also has another job."* A correct solution for this exercise is:

$$Occupation(Joe, Actor) \wedge \exists x\ Occupation(Joe, x) \wedge \neg\,(x = Actor)$$

Students would often forget to specify that x has to be different from $Actor$ in this logical sentence. The following counter model is produced by *ProofTools* when such an error is made:

$$Occupation(Joe, Actor),\ Occupation(Joe, C),\ C = Actor$$

Once again, the information provided in the counter model should help a struggling student discover the error that had been made. These examples illustrate the potential for substantial improvement to the automated grading procedure that we have used.

We are also investigating strategies for improving the quality of feedback generated by automated grading systems for computer programming exercises, typically involving writing code in Java or C++ [3]. We have reason to believe that instant feedback of a binary "Correct/Incorrect" nature, for exercises of this kind, has negative effects on students, as it can be demotivating for beginners and can promote cheating. Like the tableaux-based counter model feedback discussed above, we are seeking ways to automatically provide more rich and informative feedback to students completing online exercises.

5 Conclusion

Undergraduate students of artificial intelligence regularly experience difficulties with knowledge representation exercises. This is evident in the low mean scores on relevant final examination questions that we have reported for our AI class. Reasons for this difficulty may include the limited amount of lecture hours devoted to first-order logic knowledge representation examples and students' relative inexperience with the subject matter.

We have developed an automated assessment system for exercises in which students are asked to translate English sentences into first-order logic. Our objective was to give students more practice with knowledge representation, which would lead to improved performance on final examination questions. We deployed the system in our AI class, and it has been in use over the last three instantiations of the course. An analysis of examination scores shows a statistically significant improvement in the performance of students who have used our system.

In attempting to address a major shortcoming of the system, namely that it provides binary feedback, we have started investigating tableaux-based provers to power our automated grading system. This new approach has drawn our interest because tableaux procedures produce counter models for incorrect student submissions, and information from these counter models can be used to provide students with more elaborated and more meaningful feedback. Initial explorations of this idea show promise.

Acknowledgments

The authors would like to thank Valentin Goranko for providing constructive feedback on early drafts of the paper. Thanks are also due to the anonymous reviewers who provided helpful suggestions for improving the paper.

References

1 Bryce, D.: Sceffolding in Teaching Knowledge Representation. Journal of Computing Sciences in Colleges. 28(2), 25–31 (2012)
2 Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer (1996)
3 Kyrilov, A., Noelle, D. C.: Using Case-Based Reasoning to Improve the Quality of Feedback Generated by Automated Grading Systems. In Proceedings of the 8th International Conference on E-Learning, Lisbon, Portugal (2014)
4 McCune, W.: Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/prover9/` (2005 - 2010)
5 Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 3rd Edition, Pearson (2010)
6 Shaw, L.: ProofTools: a symbolic logic proof tree generator. `http://creativeandcritical.net/prooftools` (2010 - 2014)

A pilot study of the use of LogEx, lessons learned

Josje Lodder[1], Bastiaan Heeren[1], and Johan Jeuring[1,2]

1 **Faculty of Management, Science and Technology, Open University of the Netherlands, P.O.Box 2960, 6401 DL Heerlen, The Netherlands**
Josje.Lodder@@ou.nl, Bastiaan.Heeren@@ou.nl

2 **Department of Information and Computing Sciences, Universiteit Utrecht, The Netherlands**
J.T.Jeuring@@uu.nl

Abstract

LogEx is a learning environment that supports students in rewriting propositional logical formulae, using standard equivalences. We organized a pilot study to prepare a large scale evaluation of the learning environment. In this paper we describe this study, together with the outcomes, which teach us valuable lessons for the large scale evaluation.

Keywords and phrases propositional logic, equivalences, e-learning, feedback, evaluation

1 Introduction

Students learning propositional logic practice by solving different kinds of exercises. Many of these exercises are solved stepwise. To support a student solving such exercises an intelligent tutoring system can be very effective [9]. At the Open University of the Netherlands we are developing a learning environment (LE) LogEx[1], which supports students in rewriting propositional logical formulae, using standard equivalences. We intend to evaluate this LE with a large group of students later this year, and to prepare this evaluation we organized a small scale evaluation in December 2014 [6]. Detailed loggings of the learning environment offer the possibility to analyze the way students use our LE. For example, in an earlier study [5] loggings of students working on normalizing propositional logic expressions were used to construct a probability model of the correctness of the use of rules. In the large scale study we want to perform, our main focus is the question whether or not a student learns by using our LE. We will compare different versions of LogEx that have more (or fewer) feedback services. In the pilot study our main questions were: (1) do students learn by using the LE, and (2) what lessons can we learn for a large scale evaluation.

This paper is organized as follows. In the next two sections we describe the LogEx learning environment, and the experiment we performed with it. Section 4 summarizes the results of the assessment tests and loggings. We conclude with lessons learned from the experiment.

2 The LogEx learning environment

LogEx is a learning environment (LE) in which a student practices rewriting propositional logical formulae, using standard equivalences. The LE contains three kinds of exercises: rewriting a formula in DNF, in CNF, and proving the equivalence of two formulae. A student enters her solution stepwise. In exercises on equivalence proofs she has to motivate each step with a rule name; in exercises on rewriting to normal form this motivation is

[1] http://ideas.cs.uu.nl/logex/

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 93–100


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

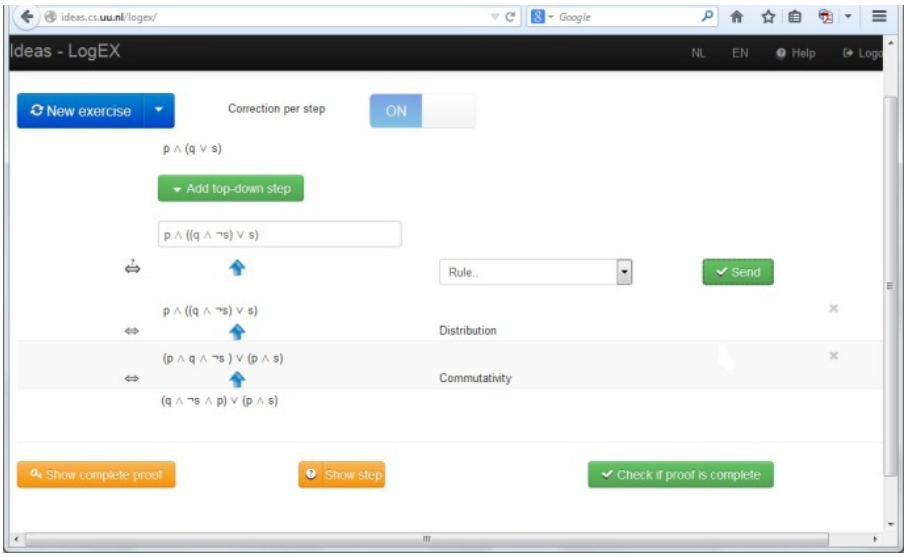


Figure 1 Screenshot of LogEx.

optional. When proving two formulae equivalent, a student can work both forwards and backwards. Figure 1 shows an example of a partial proof consisting of two backward steps. A student can change the direction in which she is working at any moment.

In the version we used in the pilot, correction per step is turned on. This implies that a student receives feedback after each step. Feedback concerns syntax errors, such as missing parentheses, or rule feedback. After a student has entered a formula, the LE will try to recognize the rule that was used. If the LE can detect a rule, it will compare this rule with the rule name provided by the student, and give an error message if the wrong rule name was given. If no rule is detected, the LE checks semantically whether or not the new and old formulae are equivalent. If they are not equivalent, the LE uses a set of common mistakes, also called buggy rules, to try to give informative feedback. For example, if a student rewrites |not (p q) (not2 p / not q) not q| into |(not p not q) (not2 p / not q) not q|, then the LE reports that this step is incorrect, and mentions that when applying DeMorgan's rule, a disjunction is transformed into a conjunction.

A student can ask for a hint (e.g. perform a backward step or apply DeMorgan), a next step, or a complete worked out solution, at any moment. The LE contains solution strategies to calculate this feed forward. A student can choose between exercises of different difficulty levels, or enter her own exercises. Feedback and feed forward are available for all exercises. LogEx integrates improved versions of earlier tools to rewrite formulas in disjunctive normal form [3, 4] and to prove equivalences [2]. The main learning goals for which we want to use the LE are: After practicing with the LE a student

- can recognize applicable rules
- can apply rules correctly
- can rewrite a formula in normal form
- can prove the equivalence of two formulae using standard equivalences
- can demonstrate strategic insight in how to rewrite a formula in normal form or prove an equivalence in an efficient way.

3 The experiment

In the last decade we performed a number of small scale experiments with earlier versions of our LE. Participants in these experiments were students of the Open University of the Netherlands (OUNL). Although we learned quite a lot from these experiments, they had two limitations. First, since OUNL is a distance university, students evaluated the LE at home. This meant that we could not observe students using the LE, and there was little control on the way how students worked with the LE. Second, the students of OUNL are rather heterogeneous.

We performed a new experiment with our LE with a group of first-year students of Utrecht University. This experiment had two goals:

- evaluate the use of the LE: do students achieve the learning goals as mentioned in Section 2 and is the support offered by the LE sufficient to reach these goals.
- prepare for a large scale evaluation of the LE this year: is the information about the LE sufficient to work with it, do students get enough time to practice with the LE, do students get enough time to answer the pre-test and post-test and is the logging adequate.

Before the experiment started, we organized a short introduction to the purpose of the experiment and we explained the main features of the LE. Then students got ten minutes to make a pre-test consisting of three exercises: prove that a formula is a tautology, rewrite a formula into normal form, and prove that two formulae are equivalent. They practiced 75 minutes with the LE, and then made a post-test comparable to the pre-test. We used a special version of the LE with a fixed set of exercises: five on rewriting a formula in DNF, five in CNF, and five on proving an equivalence. We logged all interactions of the students with the LE. During the pre-test and post-test, students could make use of a paper sheet with the list of standard equivalences that were allowed to solve the problems.

Five students participated in the experiment, all male, age 18–22, from the disciplines computer science, information science, and game technology. All students took part in the course Logic in Computer Science, and had already worked on the subjects covered in the LE in this course, except for proving an equivalence using standard equivalences, which is not part of the learning goals of the course.

4 Results

We analyzed the loggings of our LE. Only the first three normal form and proof exercises were completed by almost all students, and we only include these exercises in our results. Figure 2 shows the number of erroneous steps as a fraction of the total number of correct steps performed by the student. The x-axis displays the type and number of the exercise. All but one student completed the exercises in the order presented on this axis. Student 1 completed the exercises on proofs before the exercises on CNF. Moreover, in the first three exercises on CNF this student used only the next step button. He did complete the fourth CNF exercise without help of the LE, but we did not include this exercise in the figures. The figure shows that the students 2, 3, and 5 gradually make less mistakes, and this holds also for the last three exercises of student 4. The relatively high number of mistakes in the exercises dnf2 and pr2 can be explained by the difficulty of these exercises. Dnf2 asked for more complicated applications of the distribution rule. Pr2 had a rather simple solution, but students who did not see this solution got rather long formulae.

We also logged the time that a student needed to complete an exercise. In Figure 3 we present the time per correct step in minutes. At the start students still learn how to use the

LE, and this likely partially accounts for the decrease of time needed to take a step. Most students solved the post-test faster than the pre-test. This indicates that students learn to solve the exercises faster by using the LE. The extra amount of time needed to solve pr0 can be explained by the fact that this type of exercise was new for this group of students.

To evaluate the last learning goal, the development of strategic insight, we compared the number of correct steps of the student solution with the number of steps of the example solution generated by the LE. Here we took also into account how far a solution was simplified. For example, in the first exercise, some students used three steps to reach the normal form |(q / not r) q r|, others used four steps to reach the simplification |q r|. The worked out solution also takes three steps to reach |(q / not r) q r| and four steps to reach |q r|. Hence, in Figure 4 we score both solutions as 1, namely 3/3 resp. 4/4. The outcome suggests that students do not learn to solve the exercises more efficiently. We will discuss this outcome later.

The results of the pre-test and post-test were not very informative. The main positive result is that students completed more exercises in the post-test, see Figure 5. They made a few more mistakes in the post-test, but this might be related to the number of steps they performed. Also, there was no gain in efficiency. Because of the low number of students we could not compare the difficulty of the pre-test and post-test. We think that the post-test was slightly more difficult than the pre-test. The large scale evaluation will be designed in such a way that we can compensate for different difficulties in pre-test and post-test.

5 Lessons learned

In this section we discuss the consequences of our analyses of the loggings and the tests, together with observations we made during the evaluation session.

5.1 Do students reach the learning goals?

The first question we posed in Section 3 was: do students reach the learning goals and does LogEx sufficiently supports them to reach these goals.

Recognizing applicable rules

Students learn to recognize applicable rules, with two exceptions. LogEx admits generalizations of DeMorgan and distribution rules. For example, it is allowed to rewrite |not(p / q / r)| in |not p not q not r| in one step. These generalizations were mentioned in the introduction to the evaluation, but they were not explicitly present in the list of rules. Students did not use these generalized rules. A second rule that was hardly used is absorption. This rule is not needed to rewrite a formula in normal form, but it can simplify the calculations. Only one of the students used this rule by himself, three others only after a hint suggested to use absorption, and one student did not use this rule at all. We have to think of a possibility to make students aware of the usefulness of this rule. A possible solution might be that in case absorption is applicable, but a student chooses another rule, LogEx will point out the possibility to simplify the formula using absorption.

Apply rules correctly

LogEx does provide feedback at the rule level, and we find that this feedback helps to achieve the second learning goal. In general the error messages are sufficient for a student to correct mistakes. However, this is not always the case. In case a student accidentally rewrites a

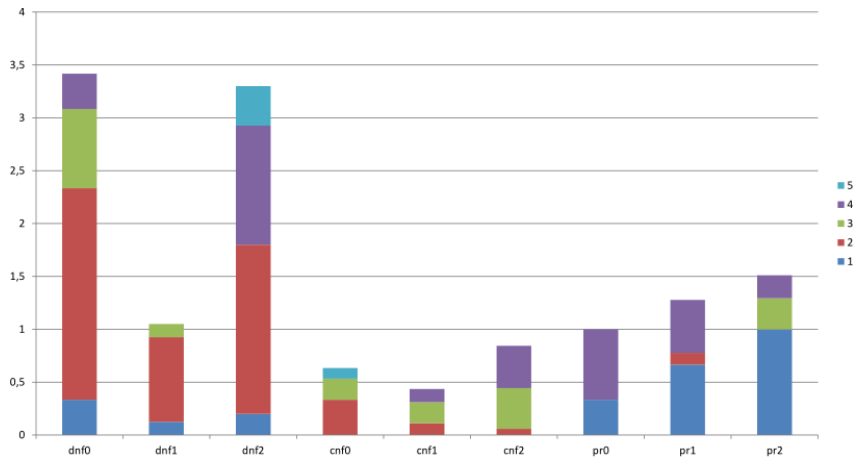


Figure 2 Fraction of errors

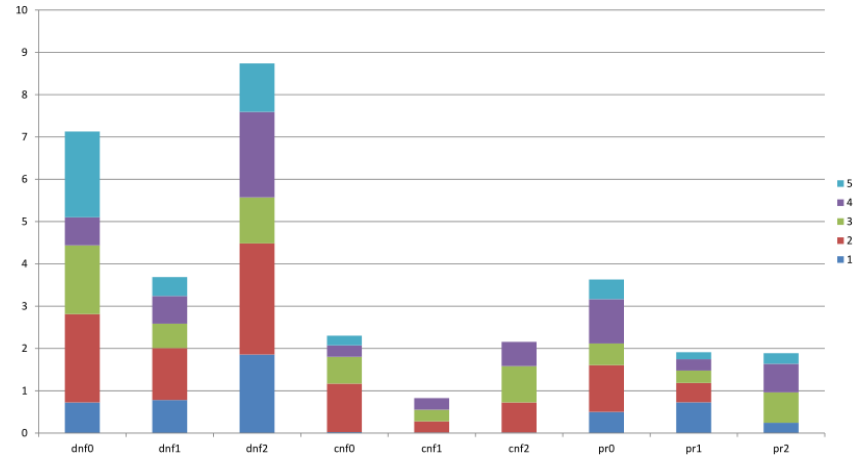


Figure 3 Time per step (in minutes)

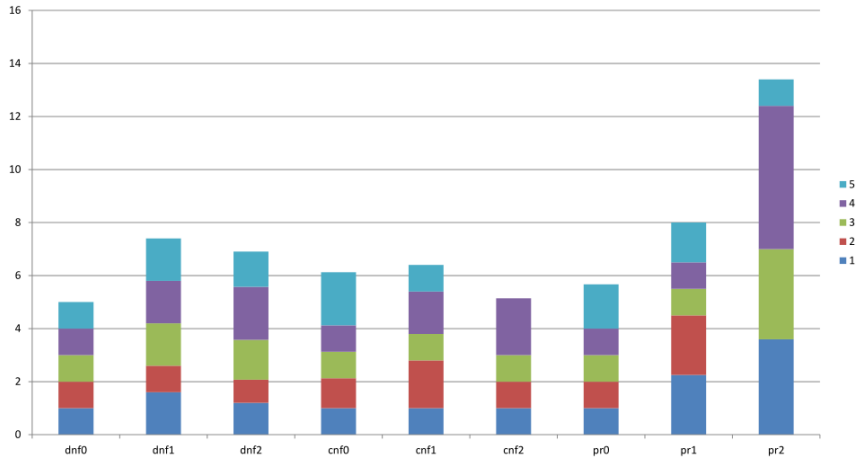


Figure 4 Efficiency measured by the number of performed steps as a fraction of the number of steps in a worked out solution

	pre-test 1			pre-test 2			pre-test 3		
	%comp	#er	eff	%comp	#er	eff	%comp	#er	eff
1	0.29	0	1	0.71	1	1	0	0	0
2	0.29	0	1	0.29	0	1	0	0	0
3	1	1	1	0.43	1	1	0	0	0
4	0.43	0	1	1	0	1	1	1	1
5	1	0	0.43	0.14	0	1	0	0	0

	post-test 1			post-test 2			post-test 3		
	%comp	#er	eff	%comp	#er	eff	%comp	#er	eff
1	1	1	1	1	0	1	0	0	0
2	1	1	1	0.7	1	1.4	0	0	0
3	1	1	1	0.9	0	1	1	0	1
4	0.54	0	1	0.67	1	1	0	0	0
5	1	2	1	0.67	1	1	0.63	2	1.4

Figure 5 Results of pre-test and post-test

%comp number completed steps divided by total number of steps
#er number of errors
eff efficiency: number of steps performed by the student divided by number of steps in example solution

formula into an equivalent formula while making a mistake, no error specific message is given. During the session we were asked several times by a student why his rewriting was incorrect. Finally, in the loggings we found some examples where students could not repair their mistakes directly in such a situation. In a next version we will also provide error specific feedback when the new formula is equivalent to the previous one.

Analysis of the loggings revealed some missing buggy rules, for example rewriting of |(p q) / (not p not q)| into |F|. From the loggings we also learned that error messages for syntax errors were not always helpful: students sometimes need several attempts to correct a syntax error.

Rewrite a formula in normal form and prove the equivalence of two formulae

The loggings and tests indicate that students do learn to rewrite a formula in normal form. Students were able to complete the exercises without too much use of the help button, and most students could finish the exercises on normal forms in the post-test. Since time in the post-test was too short to complete all the exercises, we can only use the loggings to draw conclusions about proving equivalence. The loggings indicate that students also learn to solve this kind of exercises.

Demonstrate strategic insight

The loggings and tests do not show improvement on the last learning goal. A reason might be that students had to answer different kinds of exercises, which needed partially different strategies. A careful analysis of the loggings shows that this is not the only reason. For example, one of the students developed a personal strategy of introducing double negations combined with the use of DeMorgan. In most cases this strategy was not effective, but since he got no feedback on the use of this strategy, he kept using it, also in the post-test. We think that there are at least two reasons why a student does not learn to work more

efficiently. The first reason is that LogEx does not provide feedback on the strategic level, and hence gives no information about a strategy for solving an exercise. This information is given implicitly by hints or next steps, but only one student made use of hints or next steps. The possibility to compare a solution with the complete solution was only used by one student. Help avoidance is one of the known problems with LEs [7, 8, 1]. This might be a second reason that the last goal was not met. Although in general students learn more when they have to ask for help themselves [8], in this case it seems necessary that the system provides help without being asked. LogEx recognizes when a student solution diverges from one of the possible paths of the strategy that we implemented. In a next version we might provide a warning in such a case. Alternatively, LogEx might warn a student if a solution is getting longer than the worked-out solution. A third possibility is to postpone this warning until a student has finished an exercise, but this might cause frustration.

Other remarks concerning the use of the tool

To prevent unreadable formulae and endless derivations, the use of associativity is implicit in LogEx. This means, for instance, that a student does not have to introduce or change parentheses before an application of idempotency in a formula such as |q p p s|. As a consequence, LogEx will consider |p q r| and |(p q) r| to be the same formula. There is no separate rule available to delete parentheses. In the second DNF exercise most students reached the normal form |q (not p q) p|. At this point, the students tried to get rid of the parentheses, but LogEx did not accept this. In a next version we will have to introduce the possibility to delete parentheses.

Some other minor points we learned about the LE concern user friendliness. Overall, students had no problems with the use of LogEx. However, we observed a student copying and pasting a previous formula when he wanted to correct the formula he was editing. He had not noticed that the mini-keyboard in the user interface contains an undo button.

5.2 What lessons can we learn for the large scale evaluation?

The use of a pilot study is an important principle in the design of evaluation studies [6]. Overall, the evaluation went well, but students need more time for the pre-test and post-test. We log all requests and messages between the user and the domain reasoner, but some actions are not yet logged at this moment. For example, LogEx offers the possibility to undo some steps in a proof, but the use of the undo button is not logged. We can only indirectly assume that a student removed part of her proof from the fact that the old formula in a rewriting is not equal to the new formula. Without knowing if and where students use the undo button, it is very hard to draw conclusions about the effectiveness of the student solutions.

To draw conclusions about the learning of the students during the use of the tool, it is necessary that the order in which students make the exercises is fixed.

The instruction about the use of commutativity was not clear. LogEx admits commutative variants of the standard equivalences. For example, the rewriting of |phi / (psi chi)| in |(phi / psi) (phi / chi)| is in the list of standard equivalences, and LogEx also allows the variant where |(psi chi)/ phi| is rewritten in |(psi / phi) (chi / phi)|. However, LogEx considers the rewriting of |(psi chi)/ phi| in |(phi / psi) (phi / chi)| to be a combination of distributivity and commutativity, which cannot be performed in one step. Students did perform these kind of steps without realizing why LogEx did not accept the step.

6 Conclusion

The pilot indicates that with some adaptations, especially in feedback on the strategic level, LogEx can be a helpful LE for students who practice rewriting logical formulae. The large scale evaluation later this year will have to confirm these findings. The pilot was useful for the design of the large scale evaluation, in particular with respect to the timing of the components, the instruction, and the loggings.

References

1 Vincent Aleven, Elmar Stahl, Silke Schworm, Frank Fischer, and Raven Wallace. Help Seeking and Help Design in Interactive Learning Environments. *Review of educational research*, 73(3):277–320, January 2003.

2 Josje Lodder and Bastiaan Heeren. A teaching tool for proving equivalences between logical formulae. In Patrick Blackburn, Hans Ditmarsch, María Manzano, and Fernando Soler-Toscano, editors, *Tools for Teaching Logic*, volume 6680 of *Lecture Notes in Computer Science*, pages 154–161. Springer-Verlag, 2011.

3 Josje Lodder, Johan Jeuring, and Harrie Passier. An interactive tool for manipulating logical formulae. In M. Manzano, B. Pérez Lancho, and A. Gil, editors, *Proceedings of the Second International Congress on Tools for Teaching Logic*, 2006.

4 Josje Lodder, Harrie Passier, and Sylvia Stuurman. Using IDEAS in teaching logic, lessons learned. In *International Conference on Computer Science and Software Engineering*, volume 5, pages 553–556, 2008.

5 Diederik M. Roijers, Johan Jeuring, and Ad Feelders. Probability estimation and a competence model for rule based e-tutoring systems. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, LAK '12, pages 255–258, New York, NY, USA, 2012. ACM.

6 Valerie J. Shute and J. Wesley Regian. Principles for evaluating intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 4(2-3):245–271, 1993.

7 Bram E. Vaessen, Frans J. Prins, and Johan Jeuring. University students' achievement goals and help-seeking strategies in an intelligent tutoring system. *Computers & Education*, 72:196–208, 2014.

8 Kurt VanLehn. The behavior of tutoring systems. *Journal of Artificial Intelligence in Education*, 16(3):227–265, August 2006.

9 Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

Teaching Logic for Computer Science: Are We Teaching the Wrong Narrative?

Johann A. Makowsky *

Faculty of Computer Science
Technion–Israel Institute of Technology, Haifa Israel
janos@cs.technion.ac.il

Abstract

In this paper I discuss what, according to my long experience, every computer scientist should know from logic. We concentrate on issues of modeling, interpretability and levels of abstraction. We discuss what the minimal toolbox of logic tools should look like for a computer scientist who is involved in designing and analyzing reliable systems. We shall conclude that many classical topics dear to logicians are less important than usually presented, and that less-known ideas from logic may be more useful for the working computer scientist.

1998 ACM Subject Classification K 3.2 Computer and Information Science Education

Keywords and phrases Teaching Logic, Computer Science

1 Introduction

Even in teaching mathematics we can at least attempt to teach the students the flavor of freedom and critical thought, and to get them used to the idea of being treated as humans empowered with the ability of understanding. Roger Godement, Cours d'Algèbre, Hermann, Paris 1966

In the last few years we see contradictory developments concerning teaching logic for Computer Science undergraduates. On the one side, the importance of logical tools in hardware and software engineering, artificial intelligence, and database management, including the new trend of handling large data, is widely recognized. The Vienna Summer of Logic of 2014[1] attracted over 2500 researchers, but the traditional central logic conferences (ASL Logic Colloquium, LICS and CSL) played only minor part in terms of attendance. Most attendees were practitioners of logic, benefiting from and contributing to the the *unusual effectiveness of logic in computer science*, [9]. On the other side, leading Computer Science Departments[2] have taken logic courses off their compulsory curriculum of the 3 year program. The argument behind such a decision does not question the importance of logic, but cites the scarcity of available time for teaching the essentials.

The discrepancy between the success of logic in Computer Science and its relegation from the Computer Science curricula may indicate that the standard logic courses are outdated and miss their point. Teachers of logic in Computer Science are often still teaching courses which are a mix of formalizing logical reasoning, meta-mathematics, and the fading reverberations of the famous crisis of the foundations of mathematics. By doing so, they are contributing to the disappearance of their courses from mainstream undergraduate

* President of EACSL (European Association of Computer Science Logic) 2004-2009

[1] Vienna Summer of Logic 2014 http://vsl2014.at/

[2] ETHZ Zurich, is a good example. There was strong support to keep logic in the curriculum, but finally it was dropped, after the time and priorities argument won the vote by a small margin

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 101–110
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

education in Computer Science. We have to rethink which aspects of logic matter for the Computer Science undergraduate programs. I want to argue, that the fascination with completeness and incompleteness theorems, the beauty of the compactness theorem and its applications in infinite model theory, and the focus on first order logic in general, are didactically counterproductive.

I was trained as a mathematician with a PhD in logic (1974, ETHZ Zurich), in model theory, to be precise. I started publishing in Computer Science in 1980, in database theory, logic programming and finite model theory. For the last twenty years I mostly worked on applications of logic to combinatorics [7]. I have been teaching logic for Computer Science and other logic related courses for over thirty years at the Technion's Computer Science Department. I have designed the current version of the compulsory logic course, called *Sets and Logic for Computer Science*[3]. It consists of three hours per week frontal lectures and two hours per week tutorials in a thirteen week semester. It is given twice or even three times a year with close to 400 students enrolling each year. There are various teachers teaching the course, some of them leading researchers who use logic in their work, some of them colleagues who are well educated in logic and happen to like teaching logic, without logic playing a major part in their research. This makes introducing changes in the course more difficult. I tried and still try to shift the emphasis of this course. I lectured about the rational behind this course and the changes I want to introduce at various conferences, formally at LPAR 2007, CiE 2008 [14, 16], and informally at Computer Science in Russia, Moscow 2008. I lectured about it also at ETH Zurich and the Technical University in Vienna. A journal version appeared as [15].

I have received many comments, criticisms, and encouragement[4]. In this paper I want to sharpen my point of view by addressing many of the critical comments I have received in the past.

2 Learning from the Past: Linear Algebra

We first look at the introduction of Linear Algebra into the curriculum of Mathematics and Physics students in the 1950ies. It took more than ten years to become a mainstream policy to teach Linear Algebra in the first year. We shall see in the sequel that logic courses in undergraduate Computer Science fail to achieve similar goals because they focus on the wrong issues.

The arguments for introducing Linear Algebra were twofold. Students should be exposed to Bourbaki-style abstract thinking and they should learn the tools needed later in courses like Numeric Analysis, Differential Equations, Functional Analysis, Mathematical Physics, and Quantum Physics. Linear Algebra seemed ideal for both purposes. On the abstract side, linear functions between vector spaces are distinguished from their representations via matrices over the underlying field. This is a first example where representation of (matrix-syntax) and being a linear function (function-semantics) is distinguished. Solving linear equations is a semantic problem, whereas computing a determinant is a syntactic method. The proposed course would have two goals: normal forms of matrix representations and the foundations of the determinant method. One really proves a *completeness theorem*: A system of linear equations has a unique solution (semantics) iff the determinant representing these

[3] http://webcourse.cs.technion.ac.il/234293/

[4] I would like to thank Nadia Labai for her critical reading of this paper, and to A. Avron, K. Censor-Hillel, M. Kaminski, and two anonymous referees for valuable comments and suggestions.

equations does not vanish (syntax). The introduction of such a course was a big success for two reasons: It provided the student with a theoretical framework to be used over and over again. More importantly however, the concepts lived later on and were applicable in the most diverse situations. I have emphasized here the syntax/semantics dichotomy. The truth, however, is that most teachers of such a course are not aware of this distinction.

For some strange reasons, versions of Linear Algebra or Modern Algebra[5] are still taught almost the same way in the first year of a Computer Science curriculum. The reasons are strange, because it is not only conceivable but entirely normal that students of Computer Science will never encounter determinants in their entire three year program. If a student nevertheless encounters these concepts later on, be it in Numeric Analysis, Coding Theory or Complexity, the teacher feels obliged to develop the concepts from scratch. In other words such a course given in the first year is a total waste of time.

3 Theoretical Orientation vs Practical Knowledge

It is useful to distinguish between *theoretical orientation* and *practical knowledge*. Most electricians do have a lot of practical knowledge which they can apply when installing or repairing wiring and appliances. They may have a vague knowledge of the physics and electrodynamics on which their practical knowledge is based, but they do not have to understand the Maxwell equations. Their theoretical orientation is very limited. Structural engineers must have a very sophisticated practical knowledge of material science and applied mathematics, but again their theoretical orientation concerning the foundations of physics and real and complex analysis remains vague.

Our example of the Linear Algebra course is more balanced in this respect. It tries to convey the level of abstraction needed to understand (rather than use) the tools of matrix calculus, and it does also teach how to use those tools. In Linear Algebra courses for engineers the roles of understanding vs using may be shifted in favor of using.

So what does this mean for teaching Sets and Logic for Computer Science?

4 The Traditional Narrative

When logic and set theory courses were introduced in the Mathematics curriculum at about the same time as linear algebra the main purpose was theoretical orientation. The students should be exposed to the proposed solution of the so called crisis in the foundations of mathematics. The standard textbooks were [8] by P. Halmos for naive set theory and the books [18, 13, 6] by E. Mendelson, R.C. Lyndon and H. Enderton respectively, for mathematical logic. Halmos' book was reprinted in 1974, but now K. Devlin's [3] has taken its place. The books by Mendelson and Enderton are still reprinted and used. Lyndon's text teaches logic, provided the reader already knows abstract algebra. More modern textbooks I like are [5, 4][6].

Naive set theory teaches the student to use set theory for defining ordinals and cardinals and their arithmetic. Sometimes Russel's' paradox is discussed as making a problem, sometimes as a hint that not all concepts can be formalized as sets. The axiom of choice

[5] The original title of van der Waerden landmark monograph [2], where the word "modern" was dropped long ago.

[6] For mathematicians there are also the classics [20, 19, 17]. My true favorite is [1] by S. Adamowics and P. Zbierski.

and well-orderings play a prominent role. Very little of this, however, is relevant for our Computer Science undergraduates. I will briefly sketch what may be relevant for them in the next section.

The logic texts focus on First Order logic FOL. They define first order formulas and their interpretations. They give the semantic notions of logical consequence and logical equivalence. Then they give the syntactic notions of deduction rules and proofs (proof sequences). They may distinguish between natural deduction (Gentzen style) and Hilbert style deductions. They prove the completeness and compactness theorems, the Löwenheim-Skolem-Tarski theorems, and show that neither the real number field nor the arithmetic structure of the natural numbers can be characterized in First Order Logic. They may prove Gödel's incompleteness theorems. Again, this may give some theoretical orientation. But none of this is the raison d'être why our Computer Science undergraduates should learn logic. There is no practical knowledge gained from proving the Completeness and Compactness Theorems in full rigor.

The first proper modern logic monograph was [10] by D. Hilbert and W. Ackermann, which appeared first in 1928, and in English as [11] in 1950. One should emphasize that for Hilbert and Ackermann, logic was Second Order Logic, and what we call First Order Logic is called in their book the *restricted calculus*. They gave an axiomatization of the restricted calculus and asked whether their axiomatization is complete. K. Gödel, reading the book in its first year of publication, showed immediately that the answer was positive. As it follows from Gödel's incompleteness theorem that Second Order Logic has no recursive axiomatization, Second Order Logic was deemed to be too much like set theory, and First Order Logic took center stage. Still, the natural language to describe mathematics is second or even higher order logic[7].

5 For Whom Should We Teach Sets and Logic in Computer Science?

We take our clue from the discussion about Linear Algebra. What we teach in Sets and Logic should be visibly used in the following courses:

- Data Structures and Algorithms
- Formal Languages, Automata Theory, and Computability
- Database Systems
- Graph Algorithms and Complexity
- Formal Methods and Verification, in all their ramifications
- Decision Procedures in Automated Theorem Proving

If the undergraduate curriculum contains in one form or another at least three of the above topics, a course like Sets and Logic should be taught. What remains to be discussed is the choice of topics and their emphasis.

6 The World of Sets

Our undergraduates do not need an introduction into cardinal arithmetic, but they do need an understanding and proficiency in handling statements like

[7] I still was a witness in 1966 of an argument between P. Cohen and P. Bernays on whether Cohen had proved the independence of the Continuum Hypothesis from set theory. Bernays insisted that Hilbert's problem was not about first order provability in Zermelo-Fränkel set theory.

- An ordered pair $\langle a, b\rangle$ is a set with the *basic property* $\langle a, b\rangle = \langle a', b'\rangle$ iff $a = a'$ and $b = b'$.
- There are various realizations of ordered pairs:
  (i) $\langle a, b\rangle_{Ku1} = \{\{a\}, \{a, b\}\}$.
  (ii) $\langle a, b\rangle_{Ku2} = \{a, \{a, b\}\}$.
  (iii) $\langle a, b\rangle_{Wie} = \{\{\{a\}, \emptyset\}, \{\{b\}\}\}$.
- A finite automaton is a quintuple $\langle S, \Sigma, \delta, s, F\rangle$ where S is a finite set (of states), Σ is a finite set of symbols (an alphabet), $\delta : S \times \Sigma \to S$ is a (transition) function, $s \in S$ is a starting state, and $F \subseteq S$ is the subset of final (accepting) states.

Additionally, they do need a repertoire of set constructions[8]. which allows them to construct sets which are realizations of the concepts we use in Computer Science:

- Sets, relations, functions, Cartesian products, infinite unions;
- Finite sets, countable sets, uncountable sets, equipotent sets;
- The set of words $\Sigma^\star$ over an alphabet Σ and the set of natural numbers $\mathbb{N}$.
- Inductively defined sets, such as the well-formed formulas in some formal language, etc.

Some of this material is usually covered in the beginning of high-level textbooks such as [12]. To the usual material, I would add without proof and discussion of the axiom of choice, also the statements:

- The Cartesian product of a non-empty (finite) family of non-empty sets is not empty.
- The union of countably many countable sets is countable.

However, I would advise not to talk about the axiom of choice or the well-ordering principle, and cardinalities of sets. For our purpose it suffices to talk about equipotent sets. Sets are finite, countable, or equipotent to some previously constructed set, such as the powerset of the natural numbers, or, equipotent to it, the set of functions $f : \mathbb{N} \to \{0, 1\}$.

7 Side Effects

One aspect is never covered in the usual introduction to the world of sets: The realization of concepts as sets has **side effects**.

Side effects are properties of the realization which are not intended. In the example of the three realizations of ordered pairs, a is an element only of $\langle a, b\rangle_{Ku2}$, b is not an element of any of them, $\{a\}$ is an element of $\langle a, b\rangle_{Ku1}$ only, and $\{\{b\}\}$ is an element of $\langle a, b\rangle_{Wie}$ only. The further difference between $\langle a, b\rangle_{Ku1}$ and $\langle a, b\rangle_{Ku2}$ is that to prove the basic property for $\langle a, b\rangle_{Ku1}$, extensionality suffices, whereas for $\langle a, b\rangle_{Ku2}$ the axiom of foundations is needed. This observation is needed to explain why $\langle a, b\rangle_{Ku1}$ is preferable over $\langle a, b\rangle_{Ku2}$. It is also preferable over $\langle a, b\rangle_{Wie}$, because $\langle a, b\rangle_{Wie}$ uses the empty set as part of its definition, whereas the other two versions use only curly brackets and the sets a and b.

We can look also at two definitions of natural numbers, both starting with the empty set realizing the element 0. For $\mathbb{N}_{naive}$ we define the successor of n as $\{n\}$. For $\mathbb{N}_{vNeu}$ we define the successor of n as $n \cup \{n\}$. They both satisfy the Peano Postulates. The side effects are that in the first every $n \neq 0$ has exactly one element, whereas in the second every n is equipotent to the set of its predecessors.

The notion of side effects, as properties of a realization or implementation which are not intended, is central to understanding hardware and software systems. It can be easily explained in the world of sets, and should be a *leitmotiv* throughout all the courses taught.

[8] Basically these are the constructions needed to build the cumulative hierarchy

8 Syntax and Semantics

Having learned to master inductive definitions we should now introduce the syntax of logical formalisms. This looks like a further exercise in inductive definitions. We define the formalisms of Quantified Propositional Logic and prove the Unique Readability Theorem. The meaning of a formula is given by the *meaning function* which associates with a formula its meaning: In the case of (Quantified) Propositional Logic this is a Boolean function. We can introduce the notions of logical equivalence and consequence, and prove normal form theorems and quantifier elimination. We can introduce proof sequences and state, but not prove, the completeness theorem for the proof sequences introduced. Proving it requires to much time which we need for other purposes.

As said before, Second Order Logic SOL is the logic which allows us to talk about most important concepts, say in graph theory or formal language theory. Therefore we introduce the syntax of SOL right away. We start with a vocabulary (set of relation and function symbols including constant functions which are constants). The meaning (interpretation) of a vocabulary τ is a τ-structure. The meaning (given by the meaning function) of an SOL(τ)-formula with free first order variables is a *relation in a τ-structure*. If there are no free variables, it is a Boolean value, but it is preferable to treat this as a special case and keep the case with free variables in the center of our attention. If there are free second order variables in the formula, the meaning is given by sets of relations.

9 Read and Write

Before we embark on proving meta-theorems the students should have some minimal proficiency in reading and writing SOL-formulas. The mathematics students learning logic can be assumed to be familiar with mathematical statements from solving equations and from analytic geometry. For them, the notion of the geometric locus of all points satisfying a set of equations might come natural. The Computer Science students have no such background. I am always perplexed to see how difficult it is, even for advanced Computer Science students, to correctly write down an SOL-formula expressing that a graph is connected, or Hamiltonian. The difficulty is real (and mathematical) when trying to formulate that a graph is planar, because we have to use Kuratowski's or Wagner's Theorem. Once the students have written down a formula which supposedly expresses what they had in mind, it remains difficult for the students to read their own formula in order to check its correctness.

Practicing reading and writing skills by drawing graphs and finding FOL formulas which distinguishes the graphs, or internalizing that isomorphic structures cannot be distinguished, is a prerequisite for all further developments. This brings us back to side effects. It is also a side effect that two isomorphic structures are different from the point of view of their definition in the language of sets.

10 Pebble Games Help US Understand Quantification

Students encounter great difficulties in grasping the order of quantification even in FOL. It might be useful to introduce pebble games at an early stage. They are very visual and leave an impression on the students. Reading off the winning strategy from formulas in prenex normal form is very visual and easy to understand and to prove. So if a formula with n quantifiers distinguishes two structures, player I has a winning strategy. The converse, if no formula with n quantifiers distinguishes two structures, then player II has winning strategy, can be stated without proof. The easy part can be used to show that

- One needs at least n existential quantifiers to say that there are at least n elements in the structure over the empty vocabulary.
- One cannot express that there is an even number of elements in the structure over the empty vocabulary.

If we look at a the graph of the gender-free parent relation, there are natural concepts like grandparents, siblings, cousins, and even third-cousins twice removed, which can be expressed on FOL. Other concepts, like being an ancestor, or the requirement that the parent relation be cycle-free, are not expressible in FOL, even if restricted to finite structures. Again, to show the non-definability in FOL one can use the pebble games.

One should also prove that there are $\forall\exists$-formulas in FOL which are not equivalent to any $\exists\forall$-formulas, and vice versa. Again this is not difficult using the properties that universal formulas are preserved under substructures (which can be modified to $\exists\forall$-formulas), and that $\forall\exists$-formulas are preserved under unions of chains.

All this is more relevant for the later courses than proving the completeness or compactness theorem.

11 Definability and Non-definability

Definability is a central concept of Logic. We can define concepts in the language of sets using the membership relation only. These are the set-definable concepts. The realization of these concepts as sets have many properties expressible in the language of sets from which we want to abstract. By defining concepts as τ-structures we fix the level of abstraction, and distinguish between the essentials and the side effects.

Every SOL-definable concept is also set-definable, but the converse is not true. Topological spaces are good examples for mathematicians. They are not τ-structures in the sense we have in mind. For our Computer Science undergraduates one can explain the difference between set-definability and SOL-definability by anticipating the notion of computable languages (sets of words): Every computable language is set-definable, but SOL-definable languages have bounded complexity. FOL-definable concepts are computationally even simpler. Logical formalisms are used to compute definable concepts. We choose the formalisms to be expressive enough for use in a particular application, but to be computationally feasible enough to allow implementation.

12 Conclusions

We have suggested that the meta-mathematical narrative of the traditional logic courses for Computer Science is missing its purpose and contributes to the disappearance of logic from the undergraduate curriculum.

We have emphasized that the world of sets is used for modeling concepts of Computer Science using simple set constructions and inductive definitions. It is also used to prove properties of these concepts. We suggested to introduce, first as an example of long inductive definitions, quantified propositional logic and its meaning, given by Boolean functions. This is used to develop the basic concepts of logic, logical equivalence and consequence. We suggested as a next step to introduce Second Order Logic, and leave First Order Logic as a special case. We put the emphasis on teaching how to read and write properties of the modeled concepts, and to concentrate on the expressive power of Second Order Logic and its fragments. Finally we suggested to concentrate on tools to prove non-definability such as pebble games and preservation properties.

So where are all the beautiful theorems to be taught? In follow up courses when needed. Completeness and incompleteness in a course on decision procedures of logical systems, or in a course on proof theory, compactness in a course on model theory for mathematicians, or in a course on the foundations of logic programming. None of these will be undergraduate courses. In contrast to this, what we suggest to teach is used in

- Data Structures and Algorithms
- Formal Languages, Automata Theory and Computability
- Database Systems
- Graph Algorithms and Complexity
- Formal Methods and Verification, in all their ramifications
- Decision Procedures in Automated Theorem Proving

Explaining Completeness and Compactness can be viewed as part of the *theoretical orientation*. Proving these theorems on the expense of using the meaning function to teach reading and writing SOL and FOL formulas amounts to depriving the students of the practical knowledge they need later on.

13 Postscript

Since the publication of [14, 16] I had many discussions on these topics. I also received feedback on the current paper, from the referees, and otherwise. Many raised objections to my views, however, often these were the result of misunderstandings.

Sets and Logic for Undergraduates

In [14, 16] I discussed how to teach logic to CS-students in general. I had a Logic Course in mind similar to the traditional "Cours d'analyse" of the Grandes Écoles in France still a tradition before 1968, something like Shoenfield's book, [20], but for CS students. I was told by my French colleagues that this was too elitist, that even these "Cours d'analyse" were not anymore what they used to be.

In this paper I ask the question whether one should have compulsory courses covering the basics of modeling artifacts as sets and the basics of logic already in the three year program of computer science. I also assumed that these students were likely to take later courses such as *Introduction to Database*, *Automata and Formal Languages*, and some courses which confront them with the basics of *Programming Languages* and *Hardware and Software Verification*. I assumed further that a one semester (12 weeks) course of three frontal lectures and two tutorial sessions per week was available.

I emphatically defend teaching Sets and Logic to the undergraduates. But facing the threat that by teaching too much material the department colleagues will vote Logic out of the undergraduate program I took great care to balance the topics with a view to other courses in the undergraduate program. I tailored the syllabus of the proposed course fitting my assumptions above. As the three year (or even four year) CS programs tend to be very loaded, I suggested to skip many topics dear to logicians or practitioners of logic. I presented my views on the basis of uniting the basics of modeling artifacts as sets and the basics of logic in one course. There is no need for that, but there are distinct advantages in doing so. Spreading the material over two or more courses, which is frequently done, only dilutes the material, leads to unnecessary repetitions, and creates problems of coordination among the teachers of these courses.

Where is All the Logic Gone?

It was pointed out to me that topics I suggested to skip, such as the proof of the completeness theorem, are needed in courses dealing with *Artificial Intelligence*, especially for description logics. True! Similar objections can be made also in favor of modal logic, temporal logic, proof theory, finite model theory , etc., etc. Well, all these nice and dear topics belong, from my point of view, to specialized graduate programs. The emergence of specialized graduate programs such as Logic in CS, Logic and Computation, and the like, especially in Europe (Vienna, Dresden, Trento) lends strong support to such a view.

Textbooks

My dear readers suggested that my list of books recommended was incomplete. Yes, indeed, it is. However, none of the books they suggested to include was written after 1990, and none of these books went beyond the traditional narrative.

References

1 Z. Adamowicz and P. Zbierski. *Logic of mathematics: a modern course of classical logic*, volume 22. John Wiley & Sons, 2011.
2 B.L.van der Waerden. *Modern Algebra*. Frederick Ungar Publishing Co., New York, 1948.
3 K. Devlin. *The joy of sets: fundamentals of contemporary set theory*. Springer, 1993.
4 H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, 1995.
5 H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic, 2nd edition*. Undergraduate Texts in Mathematics. Springer-Verlag, 1994.
6 H.B. Enderton. *A mathematical introduction to logic*. Academic press, 1972, 2001.
7 M. Grohe and J.A. Makowsky, editors. *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*. American Mathematical Society, 2011. in press.
8 P. Halmos. *Naive Set Theory*. Van Nostrand, Springer, 1960, 1974.
9 J.Y. Halpern, R. Harper, N. Immerman, P.G. Kolaitis, M.Y Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, 7(02):213–236, 2001.
10 D. Hilbert and W. Ackermann. *Grundzüge der theoretischen Logik, 3rd edition*. Springer, 1949.
11 D. Hilbert and W. Ackermann. *Principles of Mathematical Logic*. Chelsea Publishing Company, 1950.
12 John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Automata theory, languages, and computation*. Addison Wesley, International Edition, 2003.
13 R.C. Lyndon. *Notes on logic*, volume 6. van Nostrand Princeton, 1966.
14 J.A. Makowsky. From Hilbert's program to a logic toolbox. In *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings*, page 1, 2007.
15 J.A. Makowsky. From Hilbert's program to a logic tool box. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):225–250, 2008.
16 J.A. Makowsky. From Hilbert's program to a logic toolbox. In *Logic and the theory of Algorithms, Proceedings of the Fourth Conference on Computability in Europe, CiE 2008*, pages 304–323, 2008.
17 Y. Manin. *A Course in Mathematical Logic for Mathematicians*. Springer, 1977, 2010.
18 E. Mendelson. *Introduction to mathematical logic*. CRC press, 1964, 1997, 2009.
19 J.D. Monk. *Mathematical Logic*. Graduate Texts in Mathematics. Springer Verlag, 1976.
20 J. Shoenfield. *Mathematical Logic*. Addison-Wesley Series in Logic. Addison-Wesley, 1967.

The Roles Leon Henkin Played in Mathematics Education

María Manzano

Salamanca University, Department of Philosophy
Campus Unamuno, Edificio FES, 37007 Salamanca
mara@usal.es

Abstract

This paper is divided into two sections. In the first I give reasons for strongly recommending reading some of Henkin's expository papers. In the second I describe Leon Henkin's work as a social activists in the field of mathematics education, as he labored in much of his career to boost the number of women and underrepresented minorities in the upper echelons of mathematics.

Keywords and phrases Leon Henkin, completeness, mathematical induction, history of logic, logic education

1 Some of Henkin's expository papers

Henkin was an extraordinary insightful professor with a talent for exposition, and he devoted considerable effort to writing expository papers. I will mention three of them, while trying to convince you to read them with your students as a source of mutual inspiration.

1.1 Are Logic and Mathematics Identical?

This is the title of a wonderful expository paper [5], which Leon Henkin published in *Science* in 1962, and subtitled: *An old thesis of Russell's is reexamined in the light of subsequent developments in mathematical logic.*

I recommend that you to give this paper to your students, not only because the historical view provided is comprehensive and synthetic but also because it shows the Henkin's characteristic style; namely, the ability to strongly catch your attention from the start.

How does he achieve it? you might wonder. In that particular paper, Henkin tells us that his interest in logic began at the age of 16, when *'(I) came across a little volume of Bertrand Russell entitled* ***Mysticism and Logic****'.* In the introduction Henkin cites Russell's radical thesis, that *'mathematics was nothing but logic'* together with the companion thesis *' that logic is purely tautological'*, and he describes the strong reaction against his thesis by the academic community: *'Aux armes, citoyens du monde mathématique!'*

Henkin then devotes the first section of the paper to explain the two main ideas that could help explain how Russell arrived at his conclusion. The first was the lengthy effort to achieve a *'systematic reduction of all concepts of mathematics to a small number of them'*, and the second was *'the systematic study by mathematical means of the laws of logic which entered into mathematical proofs'.* Henkin relates the work of Frege, Peirce, Boole and Schröder, during the second half of the nineteenth century, with the two efforts mentioned above, and identifies them as the primary *raison d'etre* of *Principia Mathematica.*

In the following section, entitled *From Russell to Gödel*, Henkin explains the introduction of semantic notions by Tarski, as well as the formulation and proof of the completeness theorem for propositional logic by Post and for first-order logic by Gödel. *'This result of Gödel's is among the most basic and useful theorems we have in the whole subject of mathematical logic'.* But, Henkin also explains how, in 1931, the hope of further extension of

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 111–118

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

this kind of completeness was *'dashed by Gödel himself [...] (he) was able to demonstrate that the system of* ***Principia Mathematica****, taken as a whole was incomplete'.* Immediately after, and anticipating what the reader might be thinking, Henkin dispels the hope of finding new axioms to repair the incompleteness phenomenon.

In the section entitled *Consistency and the Decision Problem*, Henkin analyzes these important notions and also explains how *'Gödel was able to show that the questions of consistency and completeness were very closely linked to one another. [...] if a system such as the* ***Principia*** *were truly consistent, then in fact it would not be possible to produce a sound proof of this fact!'.* In the following section, named *Logic after 1936*, Henkin describes how Alonzo Church proved that no decision procedure is available for first-order logic, and he devotes the rest of the paper to set theory, recursive functions, and algebraic logic. Henkin ends the paper with a section where he analyzes *Russell's Thesis in Perspective.*

Henkin was awarded the *Chauvenet Prize* in 1964 for this paper. The prize is described as a *Mathematical Association of America* award to the author of an outstanding expository article on a mathematical topic by a member of the Association.

1.1.1 Bertrand Russell's request

In April 1, 1963, Henkin received a very interesting letter from Bertrand Russell. In it, Russell thanked Henkin for *'your letter of March 26 and for the very interesting paper which you enclosed.'* Right at the beginning Russell declared:

> It is fifty years since I worked seriously at mathematical logic and almost the only work that I have read since that date is Gödel's. I realized, of course, that Gödel's work is of fundamental importance, but I was pussled by it. It made me glad that I was no longer working at mathematical logic. If a given set of axioms leads to a contradiction, it is clear that at least one of the axioms is false. Does this apply to school-boys' arithmetic, and if so, can we believe anything that we were taught in youth? Are we to think that 2+2 is not 4, but 4.001?

He then went on explaining his *'state of mind'* while Whitehead and he were doing the Principia and added: *'Both Whitehead and I were disappointed that the Principia was almost wholly considered in connection with the question whether mathematics is logic.'*

Russell ended the letter with a request: *'If you can spare the time, I should like to know, roughly, how, in your opinion, ordinary mathematics —or, indeed, any deductive system— is affected by Gödel's work.'*

According to Annellis: *'Henkin replied to Russell at length with an explanation of Gödel's incompleteness results, in a letter of July 1963, specifically explaining that Gödel's showed, not the inconsistency, but the incompleteness of the [Principia] system.'*

1.2 On Mathematical Induction

In a personal communication Henkin affirmed that *On mathematical induction* [4], published in 1960, was the favorite among his articles because it had a somewhat panoramic nature and was not directed exclusively to specialists. He wrote: *'*[...] *but my little paper on induction models from 1960, which has always been my favorite among my expository papers'.* In it, the relationship between the induction axiom and recursive definitions is studied in depth.

Why do I so strongly recommend that you ask your students to read this paper? From my point of view, it is the best paper on logic to offer students as a first reading of a "real-life"

article. The paper is especially interesting because Henkin describes something that would never appear in a formal article: his motivation. It seems that Henkin was trying to convince a mathematical colleague about why a given argument about the existence of recursive operations was completely wrong, even though at first sight it might seem convincing.

Before going into the details of the wrong argument that motivated the whole paper, Henkin explains Peano arithmetic.

Peano axiomatized the theory of natural numbers. To do so, he started out from indefinable primitive terms —in particular, those of natural number, zero, and the successor function—- and by means of three axioms he synthesized the main facts. Among those axioms is that of induction, which states that any subset of natural numbers, closed by the successor operation and to which zero belongs, is precisely the set of all natural numbers. Although the axioms for the theory of natural numbers are very important, the most interesting theorems of the theory did not stem from them alone because in most of the theorems, operations of addition, multiplication, etc. are used.

Peano thought that after axiomatizing a theory it did not suffice to organize the facts by means of axiomatic laws; it was also necessary to organize the concepts, using definition laws. In particular, we can define addition recursively by means of:

1. $x + 0 = x$
2. $x + Sy = S(x + y)$, for all x, y of $\mathbb{N}$

However, this definition must be justified by a theorem in which the existence of a unique operation that will satisfy the previous equations must be established.

A poor argument to prove this is as follows. Let us choose any $x \in \mathbb{N}$. We define a subset G of $\mathbb{N}$, by placing all $y \in \mathbb{N}$ elements in G for which $x + y$ is defined by the previous equations. It is not difficult to see that $0 \in G$ and that $\forall(y \in G \rightarrow Sy \in G)$. Now using the induction axiom, we conclude that $G = \mathbb{N}$ and thus that $x + y$ is defined for all $y \in \mathbb{N}$.

Why is this proof wrong? This was the question that Henkin's colleague posed him. Henkin tried to convince him that because the argument was designed to establish the existence of a function f ($+$ in the example), it is incorrect to assume in the course of the argument that we have such a function. Henkin made him see that in the proof only the third axiom was used and that, if correct, the same reasoning could be used not only for models that satisfy all Peano axioms but also for those that satisfy only the induction one. Henkin called these "Induction Models" and proved that in them not all recursive operations are definable. For example, exponentiation fails.

Induction models turn out to have a fairly simple mathematical structure: there are standard ones —that is, isomorphic to natural numbers— but also non-standard ones. The latter also have a simple structure: either they are cycles, in particular $\mathbb{Z}$ modulo n, or they are what Henkin calls "spoons" because they have a handle followed by a cycle. The reason is that the induction axiom is never fulfilled alone, since it requires Peano's first or second axiom. This does not mean that Peano's axioms are redundant, as it is well known that they are formally independent; *i.e.*, each one is independent of the other two.

1.3 Completeness

If you take a look at the list of documents Leon Henkin left us, the first published paper, *The completeness of first order logic* [2], corresponds to his well known result, while the last, *The discovery of my completeness proof* [7], is a extremely interesting as autobiography, thus ending his career with a sort of fascinating loop.

I claim that reading the last paper is a must. Why? As you know, Leon Henkin left us an important collection of papers, some of them so exciting as his proof of the completeness theorem both for the theory of types and for first-order logic. He did so by means of an innovative and highly versatile method, which was later to be used in many other logics, even in those known as non-classical. In his 1996 paper, we learn about the process of discovery, which observed facts he was trying to explain, and why he ended up discovering things that were not originally the target of his enquiries. Thus, in this case we do not have to engage in risky hypotheses or explain his ideas on the mere basis of the later, cold elaboration in scientific articles. It is well known that the *logic of discovery* differs from what is adopted on organizing the final exposition of our research through their different propositions, lemmas, theorems and corollaries.

We also learn that the publication order of his completeness results ([2] and [3]) is the reverse of his discovery of the proofs. The completeness for first-order logic was accomplished when he realized he could modify the proof obtained for type theory in an appropriate way. We consider this to be of great significance, because the effort of abstraction needed for the first proof (that of type theory) provided a broad perspective that allowed him to see beyond some prejudices and to make the decisive changes needed to reach his second proof. In [9] you can find a detailed commentary of Henkin's contribution to the resolution and understanding of the completeness phenomena.

1.3.1 Henkin's expository papers on completeness

In 1967 Henkin published two very relevant expository papers on for the subject we are considering here, *Truth and Provability* and *Completeness*, which were published in *Philosophy of Science Today* [10].

1.3.1.1 *Truth and Provability*

In less than 10 pages, Henkin gives a very intuitive introduction to the concept of truth and its counterpart, that of provability, in the same spirit of Tarski's expository paper *Truth and Proof* [11]. The latter was published in *Scientific American* two years after Henkin's contribution. This not so surprising as Henkin had by then been in Berkeley working with Tarski for about 15 years and the theory of truth was Tarski's contribution.

The main topics Henkin introduces (or at least touches upon) are very relevant. They include the *use/mention* distinction, the desire for *languages with infinite sentences* and the need for a *recursive definition of truth*, the *language/metalanguage* distinction, the need to avoid reflexive paradoxes, the concept of *denotation* for terms, and the interpretation of *quantified formulas*. He also explains what an *axiomatic theory* is and how it works in harmony with a *deductive calculus*. Properties such as *decidability* and *completeness/incompleteness of a theory* are mentioned at the end. I admire the way these concepts are introduced, with such élan, and the chain Henkin establishes, which shows how each concept is needed to support the next.

1.3.1.2 *Completeness*

In this short expository paper Henkin explores the complex landscape of the notions of completeness. He introduces the notion of logical completeness —both weak and strong— as an extension of the notion already introduced of "completeness of an axiomatic theory". This presentation differs notably from the standard way these notions are introduced today where, usually, the completeness of the logic precedes the notion of completeness of a theory

and, often, to avoid misunderstandings, both concepts are separated as much as possible, as if relating them were some sort of terrible mistake or even anathema. Gödel's incompleteness theorem is presented, as well as its negative impact on the search for a complete calculus for higher-order logic. The paper ends by introducting his own completeness result for higher-order logic with general semantics. The utilitarian way Henkin uses to justify his general models as a way of sorting the provable sentences from the unprovable ones in the class of valid sentences (in standard models) is very peculiar.

2 The Roles of Action and Thought in Mathematics Education

Henkin was often described as a social activist, he labored much of his career to boost the number of woman and underrepresented minorities in the upper echelons of mathematics. He was also very aware that we are beings immersed in the crucible of history from which we find it hard to escape, an awareness he brought to the very beginning of his interesting article about the teaching of mathematics [6]:

> Waves of history wash over our nation, stirring up our society and our institutions. Soon we see changes in the way that all of us do things, including our mathematics and our teaching. These changes form themselves into rivulets and streams that merge at various angles with those arising in parts of our society quite different from education, mathematics, or science. Rivers are formed, contributing powerful currents that will produce future waves of history.
> The Great Depression and World War II formed the background of my years of study; the Cold War and the Civil Rights Movement were the backdrop against which I began my career as a research mathematicians, and later began to involve myself with mathematics education.
> (In [6] page 3)

In this paper he gave both a short outline of the variety of educational programs he created and/or participated in, and interesting details about some of them. In particular he discussed the following six:

1. 1957-59, **NSF** Summer Institutes. The National Science Foundation is an independent federal agency created by Congress in 1950. As you can read in their web page, *http://www.nsf.gov/about/*, its aim was *'to promote the progress of science; to advance the national health, prosperity, and welfare; to secure the national defense...'*. Nowadays, NSF is the *'only federal agency whose mission includes support for all fields of fundamental science and engineering, except for medical sciences.'* NSF's Strategic Plan includes Investing in Science, Engineering, and Education for the Nation's Future. In [6] Henkin related this initiative to historical facts: *'The launching of Sputnik demonstrated superiority in space travel, and our country responded in a variety of ways to improve capacity for scientific and technical developments'* In 1957, Henkin involved himself in several NSF's programs, he served as a lecturer of several courses. These programs were designed to improve high school and college mathematics instruction and were directed to mathematics teachers. Henkin explained that the variety of attitudes toward mathematics of the teachers attending the courses was amazing, and that the experience gave him a view of the nature of instruction around the country. The subject of his courses was the *axiomatic foundation of number systems*. One of his aims was to get students to understand "the idea of a proof" because he believed that it could help students in the

effort of finding proofs of their own, in a much better way than the mere understanding of the steps that constitutes a proof.

2. 1959-63, **MAA** Math Films. The **M**athematical **A**ssociation of **A**merica was established one century ago, in 1915. As you can read in their web page, *http://www.maa.org/*, *'Over our first century, MAA has certainly grown, but continues to maintain our leadership in all aspects of the undergraduate program in mathematics'.* Long before internet resources became available, the MAA made movies. As Henkin said: *'Sensing a potential infusion of technology into mathematics instruction, MAA set up a committee to make a few experimental films.* [...] *the committee approached me in 1959-60 with a request to make a filmed lecture on mathematical induction which could be shown at the high-school-senior/college-freshman level. I readily agreed.'* The film was part of the Mathematics Today series, and was shown on public television in New York City and in high schools. In [6] Henkin explained the preparation of the film, both from a technical point of view and from a methodological and pedagogical perspective. He attributed the lack of understanding of the induction principle at the undergraduate level to the current formulation as a mathematical principle, and he proposed to use it as *'a statement about sets of numbers satisfying two simple conditions; formulated in this way, it is a fine vehicle for giving students practice in forming and using sets of numbers to show that all natural numbers possess various properties'*

3. 1961-64 CUPM. The Mathematical Association of America's **C**ommittee on the **U**ndergraduate **P**rogram in **M**athematics (CUPM) is charged with making recommendations to guide mathematics departments in designing curricula for their undergraduate students. In the sixties, the CUPM proposed courses to be taken by elementary teachers. In [6] Henkin said *'Some of my colleagues and I began, for the first time, to have classroom contact with prospective elementary teachers, and that led, in turn, to in-service programs for current teachers. I learned a great deal from teaching teachers-students; I hope they learned at least half as much as I!'*

4. 1964-. Activities To Broaden Opportunity. "The sixties" is the term used to describe the counterculture and revolution movement that took place in several places in the U.S.A. and Europe. Berkeley students were taking energic actions against segregation in southeastern U.S.A. as well as against military actions in Vietnam. In [6] Henkin said *'In the midst of this turmoil I joined in forming two committees at Berkeley which enlarged the opportunity of minority ethnic groups for studying mathematics and related subjects.* [...] *We noted that while there was a substantial black population in Berkeley and the surrounding Bay Area, our own university student body was almost "lily white" and the plan to undertake action through the Senate was initiated'* In 1964, Leon Henkin and Jerzy Neyman, a world-famous Polish-American statistician from Berkeley University, started a program at Berkeley to increase the number of minority students entering college from Bay Area high schools. Henkin told us that the inicitiave came after Neyman participation in *'the MAA's Visiting Mathematician Program in Fall 1963. He lectured in southerns states where, by law, whites and blacks studied in separate colleges. Upon returning to Berkeley he told some of his friend that "first-rate students were being given a third-rate education" '* Henkin and Neyman undertake actions through the Senate, and in 1964 the Senate established a committee with the desired effect. The committee recruited promising students and offered them summer programs to study mathematics and English. If they persisted in the program, they were offered special scholarships.
In the same year, 1964, Henkin heard a talk by a Berkeley High School teacher, Bill Johntz. After that, Henkin was invited to see him in action, while he was teaching

mathematics to elementary students from low-income neighborhoods, and realized that Johntz was able to raise great enthusiasm in the class. Significantly, students enjoyed and actively engaged in the process of learning, and they became integrally involved in their own education. He was using a Socratic group-discovery method modeled after the filmed teaching of David Page, a University of Illinois mathematics professor. The method was working well, and they recruited university mathematics students as well as engineers as teachers, after some training. The program was called Project *SEED* —Special Elementary Education for the Disadvantaged. This program is still alive, as you can see in their web page, *http://projectseed.org/.*

5. 1960-68. Teaching Teachers, Teaching Kids. In this paper Henkin described several conferences on school mathematics as well as several projects and courses he was involved in. The following paragraph caught my eye: *'After I began visiting elementary school classes in connection with CTFO, I came to believe that the emotional response of the teachers to mathematics was of more importance to the learning process of the students than the teacher's ability to relate the algorithms of arithmetic to the axioms of ring theory'.*
6. 1968-70. Open Sesame: The Lawrence Hall Of Science. The Lawrence Hall of Science, a science museum in Berkeley, was created in honor of the 1939 Novel prize winner Ernest Orlando Lawrence. As you can read in their web page,

   http://www.lawrencehallofscience.org/about

   'We have been providing parents, kids, and educators with opportunities to engage with science since 1968.'
   According to Henkin's tale *'In 1968, the newly appointed director, Professor of Physics Alan Portis, decided to transform the museum into a center of science and mathematics education, whose functions would be integrated with graduate research programs directed by interdisciplinary group of faculty.'* To help in his endeavor, he gathered a group of faculty from a variety of science departments interested in science education. *'These faculty members proposed a new, interdisciplinary Ph.D program under the acronym SESAME —Special Excellence in Science and Mathematics Education. Entering students were required to have a masters degree in mathematics or in one of the sciences. Courses and seminars in theories of learning, cognitive science, and experimental design were either identified in various departments, or created'.* Nitsa Movshovitz-Hadar, a student from the Technion in Israel, was admitted in the SESAME program, she wrote her thesis under the direction of Leon Henkin. Nitsa is one of the contributors of the book, *The Life and Work of Leon Henkin: Essays on His Contributions.*

References

1. Anellis, Irving. Review of Bertrand Russell, Towards the "Principles of Mathematics", 1900-02 , edited by Gregory H. Moore, and Bertrand Russell, Foundations of Logic, 1903-05 edited by Alasdair Urquhart with the assistance of Albert C. Lewis. Mod. Log. 8 (2001), no. 3-4, 57–93. http://projecteuclid.org/euclid.rml/1081173771.
2. Henkin, L.: The completeness of the first-order functional calculus. The Journal of Symbolic Logic 14(3), 159–166 (1949)
3. Henkin, L.: Completeness in the theory of types. The Journal of Symbolic Logic 15(2), 81–91 (1950)
4. Henkin, L.: On mathematical induction. The American Mathematical Monthly 67(4), 323–338 (1960)

5 Henkin, L.: Are Logic and Mathematics Identical?. Science 138, 788–794 (1962)
6 Henkin, L.: The Roles of Action and of Thought in Mathematics Education–One Mathematician's Passage. In: Fisher, N. D., Keynes, H. B., Wagreich, Ph. D. (eds.) Changing the Culture: Mathematics Education in the Research Community. CBMS Issues in Mathematics Education, vol. 5, pp. 3-16. American Mathematical Society in cooperation with Mathematical Association of America, Providence (1995)
7 Henkin, L.: The discovery of my completeness proofs. The Bulletin of Symbolic Logic 2(2), 127–158 (1996)
8 Manzano, M. et als (eds.). *The Life and Work of Leon Henkin: Essays on His Contributions*. Studies in Universal Logic. Springer International Publishing. Switzerland (2014)
9 Manzano, M. Henkin on Completeness. In: Manzano, M. et als (eds.). *The Life and Work of Leon Henkin: Essays on His Contributions*. pp. 149-175. Studies in Universal Logic. Springer International Publishing. Switzerland (2014)
10 Morgenbesser, S. (ed.). *Philosophy of Science today*, New York: Basic Books. (1967)
11 Tarski, A. Truth and Proof. Scientific American, June 1969, 63–70, 75–77 (1969)

Fail Better: What formalized math can teach us about learning *

João Marcos

UFRN, Brazil
jmarcos@dimap.ufrn.br

Abstract

Real-life conjectures do not come with instructions saying whether they they should be proven or, instead, refuted. Yet, as we now know, in either case the final argument produced had better be not just convincing but actually verifiable in as much detail as our need for eliminating risk might require. For those who do not happen to have direct access to the realm of mathematical truths, the modern field of formalized mathematics has quite a few lessons to contribute, and one might pay heed to what it has to say, for instance, about: the importance of employing proof strategies; the fine control of automation in unraveling the structure of a certain proof object; reasoning forward from the givens and backward from the goals, in developing proof scripts; knowing when and how definitions and identities apply in a helpful way, and when they do not apply; seeing proofs [and refutations] as dynamical objects, not reflected by the static derivation trees that Proof Theory wants them to be.

I believe that the great challenge for teachers and learners resides currently less on the availability of suitable generic tools than in combining them wisely in view of their preferred education paradigms and introducing them in a way that best fits their specific aims, possibly with the help of intelligent online interactive tutoring systems. As a proof of concept, a computerized proof assistant that makes use of several successful tools already freely available on the market and that takes into account some of the above findings about teaching and learning Logic is hereby introduced. To fully account for our informed intuitions on the subject it would seem that a little bit extra technology would still be inviting, but no major breakthrough is really needed: We are talking about tools that are already within our reach to develop, as the fruits of collaborative effort.

1998 ACM Subject Classification K.3.1 Computer Uses in Education; F.4.1 Mathematical Logic

Keywords and phrases mathematical thinking; computer-assisted instruction; proof assistants; intelligent tutoring systems

1 Ask yourself again: What is a Proof?

Non-obvious arguments have interesting inner structures: *givens* may be assumed to be connected to *goals* by directed loop-free finite paths; each node of such a structure would be annotated with some statement; the finite collection of all edges concurring in a node S_m would itself be annotated by a *justification* according to which the statements $S_1, S_2, \ldots, S_{j_m}$ contained in their other endpoints would suffice for an appropriate inferential link to be recognized; source nodes would be labelled in such a way that some of them could be identified as temporary suppositions, and the others as assumptions that would have been employed as starting points for the corresponding argument. If one puts the corresponding directed acyclic graph upside-down and unravels the paths to avoid a node to be visited

* The author acknowledges partial support by the Marie Curie project PIRSES-GA-2012-318986, funded by EU-FP7, and by CNPq / Brazil.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 119–128

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

more than once, the sink nodes become roots of their own *derivation trees*. While checking that such an argument has been constructed flawlessly according to a number of inference rules set in advance essentially reduces to a rather costless tree traversal task, the very task of constructing the goal-directed argument, starting from the initial given hypotheses is however a potentially costly problem that might require a lot of ingenuity to be solved. However, real-life problems will often demand practitioners to construct precisely this kind of interwoven structures concerning a given conjecture without a hint on whether this conjecture is *true* or *false* — and often the arguments supporting its truth are quite distinct from the arguments supporting its falsity. It should be no wonder that students often have difficulty with mastering the techniques for going about and develop their own arguments, and tutors often have a hard time teaching students how to proceed.

Many tasks related to exploring conjectures by either proving or refuting them may be fully automated, as, for instance, in cases in which complete decision procedures are known to exist. Arguably, however, if students are to be more than button-pushers for imported technological toys that play by themselves, they have more to learn by *implementing* the decision procedures than by actually *using* their tools to evaluate whatever conjecture needs to be checked. At any rate, I believe students can learn many important skills even when working with fully automatable theories, as long as they approach them in a productive way, and I believe tutors have a lot to learn from the many advancements that have been made in the fast-growing field of formalized mathematics. The teaching principles subsequently discussed in the present paper are intended to provide evidence for such beliefs.

2 Lessons from formalized mathematics

Mathematics in the early 19th century was navigating a sea of uncalculated logical gaps and gluts, with few safe harbors in which to dock. Its non-systematic use of axioms and relatively careless use of definitions, found amid a baroque prose that allowed for much ambiguity and imprecision to crop in, resulted in a disgraceful state of affairs in which the mathematical vessel appeared to be going adrift. Disaster seemed imminent, had it not been by the masters of the trade not letting it sink. The forerunners of the now venerable foundational schools strove to provide firm grounds for Mathematics, and Logic had important roles to play in such quest. 'Safe' proof methods and techniques were progressively identified, and stricter standards were imposed for an argument to be considered acceptable. In the following century the very notion of *proof* became a mathematical object in itself, and one whose characterization and properties were to be deeply investigated.

The works of Gentzen and other wise forerunners of Proof Theory have not only shaped the way we now think about proofs, but have heavily influenced the ulterior design of mechanized proof-assistants. As a matter of fact, many lessons in Logic seem to have been made clearer by implementations of such proof-assistants than they had been for several earlier generations of 'pure' mathematical logicians, or early type theorists — as an example, one could argue that the difference in the roles played by free and by bound variables becomes more striking if you think of them as sorted into quantifier rules that introduce *parameters* in contrast to those that introduce *unknowns*. My contention in the present paper is that paying more attention into how novice students interact with such tools for doing computer-assisted math might help understanding other important lessons that would have remained unheard. I propose to discuss some such largely understated lessons in what follows.

Seeing proofs [and refutations] as dynamical objects. You give your student an exam to be solved with pencil-and-paper. She has to write a convincing argument that will lead to either proving or refuting a given conjecture. Having been rightly taught that a mathematical proof is about adequately justifying her conclusions and not about providing an explanation of her thought processes (cf. [13]), the student hands back to you a scratch of the logical structure of her argument, in the form of a derivation tree. How did she arrive to it? You might never know. To the best of your knowledge, she might have received the full argument ready from the dancing hands of Shiva — or maybe from a fellow student.

This time you are the one scratching the skeleton of a proof in the blackboard, in front of your student. Instead of cleanly proceeding from hypotheses to conclusion, or maybe proceeding the other way around, you allow yourself some false starts, backtrack from dead ends, get stuck here and there, and fail in every possible way — having an eye at what can be learned from such repeated failures. You exhibit the full plethora of feelings that accompany the process of inventing/discovering a proof, including the mixture of satisfaction and relief when you manage to fill all the blanks that separate givens and goals. Your student will be happy to follow you attentively, and then take a picture of the final result with her mobile phone. When she arrives at home, though, she tries to retrace your steps, but does not manage to reproduce the same derivation tree for the given conjecture — or to produce any derivation tree at all.

In both situations, what is not being captured by the student is the *dynamical* aspect of the proof object. The complete derivation tree that you find at your student's exam is wholly static, and the media in which it has been registered does not allow you to replay its construction. The picture taken by the student of the proof that resulted from your toil is only the last frame of a movie. It might be the climax of the movie, but, more often than not, it does not allow the interested spectator to recover details of the full story that led to it.

What's worst, in the paper or in the blackboard, an entirely correct proof might have been produced in the wrong way, and you will never get a hint of that just by looking at the final derivation tree that represents such a proof. For instance, consider the usual derivation of the quantifier shift that allows one to conclude that $(\forall x)(\forall y)P(x,y)$ and $(\forall y)(\forall x)P(x,y)$ are equivalent formulas. Of course, to achieve your initial goal what you should do is start by fixing a certain y_1 to represent an arbitrary member of your universe of discourse and set as a new goal the proof of $(\forall x)P(x,y_1)$. Next, you fix an arbitrary x_1 and set as a new goal the proof of $P(x_1,y_1)$. So far you have invoked two applications of a logical rule often called '$\forall$-introduction'. Because you still need to prove $P(x_1,y_1)$ from $(\forall x)(\forall y)P(x,y)$, two applications of '$\forall$-elimination' will now do, if you choose the obvious witnesses for the instantiations. Pick your preferred proof-assistant and write the proof starting from the given, applying $\forall$-elimination (twice) and then try applying $\forall$-introduction. You will fail: the instances that you will produce are not 'arbitrary', but only specific objects of your domain; you should not expect to generalize from them.

However essential is dynamics to a proof, the standard theoretical treatises on Proof Theory still choose to simply ignore it. Rather than defining proofs as sequences of collections of derivations trees each with their own purpose and history, a proof is commonly identified simply with the final derivation tree that magically gather all the former trees into a whole. Most of the movie has been lost in the process, as spectators just look at the final credits in awe. Yet all practitioners know perfectly well that something fundamental is missing.

Playing and replaying proofs. The last example in the previous discussion has already illustrated a common feature of proof dynamics that is taken into account by a practitioner

writing a so-called *proof script* for a given conjecture: reasoning may proceed *forward* from the givens or *backwards* from the goals. For instance, given two families of sets, $\mathcal{F}$ and $\mathcal{G}$, a student might want to prove that $x \in \bigcup \mathcal{F}$ follows from $x \in \bigcup(\mathcal{F} \cap \mathcal{G})$. Now, reasoning backwards from the initial goal, proving $x \in \bigcup \mathcal{F}$ means going through the so-called '$\exists$-introduction' rule, where the new goal would consist in finding a witness $y \in \mathcal{F}$ such that $x \in y$. On the other hand, using the hypothesis $x \in \bigcup(\mathcal{F} \cap \mathcal{G})$ means going through the so-called '$\exists$-elimination' rule, for in this case the procedure, reasoning forward from the given, would consist in considering an arbitrary element z of $\mathcal{F} \cap \mathcal{G}$ such that $x \in z$. The student will easily tie the two ends of the proof, of course, when she finds out she might choose $y = z$.

"If you can't solve a problem, then there is an easier problem you can solve: find it." (cf. [9]). Flexible proof assistants will typically allow you to solve a problem by exploring proofs in both directions, forward and backwards, to try and find that easier problem. Using an analogy from programming language paradigms, proof languages used by proof assistants might come in a *declarative* style that "resembles program source code more than mathematics" or in a *procedural* style in which "one holds a dialogue with the computer" (cf. [15]). For the obvious reasons, research efforts have been consistently devoted to developing a *mathematical vernacular* for proof assistants (cf. [3]) that resembles the formal use of natural language found in mathematical texts.[1] Isn't this a whole lot of work just to end up with something that appears as if it had been done with pencil-and-paper? Well, this is where the analogy breaks down. Note indeed that proofs described as scripts in a proof assistant can be *executed* like any program. That allows one, in principle, to "describe not only what the machine should do, but also why it should be doing it — a formal proof of correctness" (cf. [5]).

More on the power of computer-inspired analogies. Many a cognitive scientist has found a common denominator with defenders of Artificial Intelligence in explaining the difference between the mind and the brain in terms of an analogy to the difference between software and hardware. Imperfect as any analogy, such an explanation will already allow for debates to run, for instance, about the meaning and the roles of consciousness and emotions in the resulting picture. More to the point, a little experience in programming or some experience with proof assistants often help bringing forth some more specific and useful analogies. For instance, a comparison that many students find compelling is the one that equates the discharge of assumptions and the discharge of parameters in certain natural deduction proofs to the destruction of local variables, in imperative programming, when leaving the scope of functions to which they belong, and that equates non-discharged hypotheses to global variables of a program. For another example, the familiarity with functions with type `void`, in programming, not rarely appears to be advantageous to students striving to understand the interpretations of nullary functions or nullary predicates.

Allegedly, a good education in Computer Science turns the apprentice more prone to recognizing that many symbols for variables that appear in mathematical expressions are actually bound to certain operators (such as summations, limits or integrals), or are bound to hidden (universal) quantifiers, and so they might usefully be thought of as standing as placeholders that allow for certain kinds of syntactic reductions. From an implementation viewpoint, the conceptual understanding of the role of *binding* is in fact shared by computer scientists more often with linguists than with mathematicians. On the same track, the

[1] Mathematicians can be quite conservative, in fact, and even proposals to the effect that an effort should be made to employ a more disciplined and structured use of language (cf. [6]) have for the most part appeared to fell on deaf ears.

insightful Curry-Howard correspondence that maps propositions to types, proofs to programs, and proof simplification strategies to program evaluation strategies tends to be better appreciated by those who have been exposed to the compilation routines of modern programming languages or proof assistants.

And here is a final note on the seductive power of analogy. As with any search related to problem solving, one way of thinking about *proof search* is as another example of disciplined activity that benefits from intelligent guidance. Now, if one is still willing to defend that "intellectual activity consists mainly of various kinds of search" (cf. [12]), then one should be more than willing to build tools to let the students experiment with and search for their own proofs, and invest on the partnership between students and their 'intelligent machinery'.

When definitions apply [and when they do not]. There is no space here, nor is it my intention, to discuss the role of definitions in the teaching and learning of mathematics. This has been done with competence in [14], and elsewhere. The value of definitions that make precise certain fundamental concepts should not be underestimated: It would not be (much of) an exaggeration, for instance, to say that the whole of Logic amounts simply to the study of a couple of capital definitions, namely, those of the concepts of *satisfaction*, *proof*, *computation*, and *valid reasoning*.

For all that matters, and putting aside for a moment their cognitive role in reasoning, I may here assume definitions to be dealt with through a simple rewriting: *definiens* is allowed to replace *definiendum*, or vice-versa. Assuming the student can process such rewritings while cleverly avoiding circular reasoning, assuming that the student competently grasps the content of the definitions she is supposed to work with, and also assuming that the student can make such definitions part of her technical repertoire, it still seems to me that insufficient emphasis is put on knowing what to do when a defined term, operation or relation applies as well as when it does not apply in the theory at hand. For instance, if set complement ($\setminus$) is defined in the usual way, what does it mean to reason *from* a hypothesis of the form $c \notin a \setminus b$? (novice students often have difficulty in recognizing in this the conditional assertion 'if $c \in a$ then $c \in b$') For a more important example, what does it mean for a statement *not* to follow from a given set of hypotheses? Many students who claim to understand, respectively, the notions of set complement and logical consequence still have difficulty giving useful answers to the two previous questions.

Much of what has been noted as a problem about dealing with definitions extends to dealing with identities (for which careless rewriting might in fact easily turn out nonterminating). Universal statements concerning the *failure* of certain identities are seldom used in theories that focus exclusively on equational reasoning. For the interested reader, I recall though a nice example of the possible use of inequalities, in combination with automated proof search. Recall that a Robbins algebra is a structure containing a commutative and associative binary symbol $+$ and a unary symbol $'$ satisfying the equation $(\forall x)(\forall y)((x' + y')' + (x' + y)') = x$. A computerized proof that a Robbins algebra that contains an $+$-idempotent element is Boolean was quickly found by supposing, by contradiction, the *denial* of another equation which was known since 1933 to axiomatize the variety of Boolean algebras. This result ended up representing a key step in the proof, entirely uncovered by an automated prover, that *all* Robbins algebras are Boolean, settling a problem that had been open for more than six decades (cf. [7]).

Finding the right balance between automation and our pedagogical purposes. Are computerized proofs legible? And, to better start with, are they *informative*? How much will

the students learn from them? While the first question might largely amount to a puzzle for semiotics, the last question poses a challenge for educators. In the previous paragraph I have just mentioned the 'solution to the Robbins problem'. It should be noted that this solution, which has resisted the attacks of generations of human provers, also proved to be really hard for human checkers, for the substitutions required by the use of the equations involved terms that were long and difficult to parse. In this sense, the first computerized proofs that were automatically found seemed too unstructured and did not contribute much in insight for their intrepid human readers.

So much for fully *automated* proving. As we mentioned before, the languages used by *interactive* proof assistants aim at being flexible, and often contain 'tacticals' or 'methods' that allow one to fine tune the amount of automation one wants to introduce into the most bureaucratic or relatively boring steps of a proof construction. In the case of large steps done with automation, many proof assistants will allow one to inspect the structure of the proof object that has been constructed by them, hidden from the users' eyes. The students directly involved in the very implementation of such tacticals or automated provers will no doubt learn a lot from their service to the community. However, novice students that abuse on proofs produced by a 'blast', followed by a seeming miracle, often fail to see how such proofs relate to the problems they have when confronted with the simplest mathematical conjectures. In that sense, it would seem to me that, for pedagogical purposes, automation had better be introduced in really small doses.

Of course, it all depends on *what* we intend to teach. Take the example of classical propositional logic, and their usual analytic tableaux or sequent systems. Both proof systems are capable of originating simple decision procedures. However, if our purpose involves capitalizing on the conceptual distinction between formal proofs and formal semantics, tableaux tend to dangerously tread on the dividing line. Moreover, when done on pencil-and-paper, the steps taken in the construction of tableaux are often left unjustified, as if the derivation tree by itself would be straightforward to decipher. Arguably, and that will surely depend on the way you teach them, much more flexible systems based on sequents would also seem to pose some difficulties related to the understanding of formal semantics. Many will contend, in fact, that natural deduction is superior as a proof system that allows students to more easily construct a conceptual map in between the way they have been taught to formulate their mathematical reasoning using regimented fragments of natural language and their conversations with the machine. Not by chance, several interactive proof assistants in the market do make use of some variety of natural deduction in their metalanguage. All in all, if I may insist, assuming the chosen purpose of teaching structured mathematical thinking, the motto should be: 'Less automation Is More'.

On the role of proof strategies. Some proof systems are so robust that proofs written in them are bound to terminate (by confirming or by refutating of a given conjecture) no matter how you proceed in developing them. The odd worst-case scenario for usual analytic tableaux for classical propositional logic, for instance, is known to be much more costly than truth-tables, but if you persist long enough in decomposing your hypotheses, you will exhaust any tableau, no longer being able in fact to further extend its branches by the addition of new data. The first-order case, as it should be clear, is quite different in that respect: Already in the examples we presented in the above paragraphs, interchanging the order in which the quantifier rules were applied could easily lead to failed attempts at producing a proof.

Some other proof systems will only work well with a lot of external help. For many decidable modal logics, for instance, some of the best proof systems available need loop-

checkers to put an end to their proof-search procedures (cf. [8]). Some classic-like tableaux for many-valued propositional logics —all such logics being obviously decidable— will only be assured to produce terminating proofs if you abide to certain proof strategies (cf. [1, 2]). Here again there would seem to be a task for proof-theorists: The lesson concerning the importance of employing proof strategies is in fact general enough to suggest that a proof system should *always* be presented associated to certain proof heuristics — even when the choice of such heuristics turns out not to affect the use of the system in actually designing successful derivations.

Designing a proof system that only allows its user a very small number of options for the construction of her proofs is often a poor and unrealistic alternative. Equipping the user with variegated tools-for-thought and proof techniques seems much more advisable. If this generous education process is carried out wisely, then much longer after the student has completely forgotten what she has once proven she might still know how to approach new (or the same old) problems, and go about solving them. The student had better learn how to construct a mathematical argument for a direct proof, an existential one, proofs by *reduction*, by contraposition or by cases, learn the essential forms of induction, learn the basic useful combinatorial principles. And she should learn how to refute a conjecture by a counter-example along with its corresponding step by step justification. What exactly she happens to end up proving, as long as it meets her professed goal, is often less important than the increased expertise acquired by her in employing the appropriate strategies towards accomplishing the task at hand.

3 A case study: the TryLogic

We here briefly assess a tool that embodies some of the above discussed teaching principles. Its design was based on a number of preexisting tools, following the IMS interoperability standards for distributed learning[2], and may be integrated as an external tool to any Virtual Learning Environment[3]. A major interactive theorem prover[4] is used by our tool through the ProofWeb interface[5]. Finally, students are intended to learn how to use the system exclusively through the TryLogic[6] system, an infrastructure consisting of an interactive tutorial, based on a successful online system for teaching programming languages[7]. Students have common access to a number of exercises to practice their skills with natural deduction, and typically they will receive directly in their areas a number of personalized exercises concerning: randomly chosen challenges involving carefully handcrafted conjectures to be proven; computer-generated exercises involving conjectures to be refuted; a number of computer-generated challenges in which they are not told whether the corresponding conjectures should be proven, or else refuted. The latter two groups of propositional-level

[2] See `http://www.imsglobal.org/lti/index.html`.
[3] We happened to use Moodle (`https://moodle.org/`) to that purpose.
[4] We have used a standard formal theory for *natural deduction* implemented in Coq (`https://coq.inria.fr/`), to which I have added primitive rules for bi-implication, and we implemented in Coq also a *refutation theory* for reasoning with the notion of satisfaction in formal semantics.
[5] We have in fact installed our own ProofWeb (`http://proofweb.cs.ru.nl/`) local server, and extended it to correctly display the trees from both our full natural deduction theory and our refutation theory.
[6] Our system is available at `http://lolita.dimap.ufrn.br/trylogic`.
[7] See `http://try.ocamlpro.com/`.

exercises are produced by a tool[8] implemented over the SAT-solver Limboole[9].

In my understanding, the whole process of writing proof scripts with the help of a proof assistant helps the students see proofs as instructions to be executed in order to dynamically explore a problem-solving challenge, even if that does not perfectly match the static proof-theory that they learn from the mathematical textbooks. The difficulty of the task, in which we strictly control the amount of automation that we allow the students to use, makes them ponder about how useful it may be to entertain a strategical approach to a given challenge, and the nature of the task and the easy interaction with and responsiveness of the underlying media encourages them to do some heuristic exploration through trial and error.

Before TryLogic, our group had invested for a few years on the Logicamente (cf. [11]), a growing collection of Learning Objects combined with the respective learning scripts, expositions, tasks and activities on subjects of Logic. The latter suite of tools was largely independent from other existing tools, and was developed *ab nihilo* through the collaborative effort of students that got involved in the project and were evaluated by their contributions to it. My impression about the project Logicamente, as fun as it might have been to work on, is that it was much more successful, from a pedagogical viewpoint, for the students that *implemented* its modules than for other students who *used* these modules later on. I am not trying now to sell TryLogic (and its associated tools) to you, even though I would be delighted if you bought it (and it is free of cost). What I feel to be really important about this new project (which we report upon in more detail in [10]) is that it is entirely based on exploring the teaching principles discussed in the previous section, for whose convenience and effectiveness I do not intend to offer here independent evidence except for the fact that a few dozen students have had very fruitful interactions with TryLogic along the last three years.

Coda

The critical transition from high school to higher education has —and with a good reason— received increased attention from educators worldwide. Two distinct paths might be taken by in individual undergoing post-secondary education, namely: focusing on technical content that adds to *know how*, or investing on a scientific education that presupposes a frame of mind not dispensing with the *know why*. It has been eloquently argued elsewhere that Logic has a secure role in teaching proof and that this would be better promoted by emphasizing general principles leading to deeper understanding, rather than by focusing on concrete narrow problem-solving strategies; more generally, teaching logical reasoning "builds students' confidence in the rationality of the mathematical enterprise and helps allay their fear of failure" (cf. [4]).

All education starts from personal reflections. The present paper has collected my own reflections, at the present date, on the topic of 'tools for teaching logic', and what we can learn from and with such tools. Scientific discussions concerning education, though, cannot end at the level of intuition. We need data.

In the above I have proposed that Proof Theory should redefine proofs as dynamical objects, have argued about the advantages of treating proofs as executable code, and of testing the understanding of definitions by inquiring about the conditions in which they fail to apply, have commented upon the contributions given by certain computer-inspired

[8] The `Conjectures Generator` is available at `http://lolita.dimap.ufrn.br/logicamente-ge/`, with open source code available at `http://github.com/terrematte/logic-propgenerator`.
[9] Available at `http://fmv.jku.at/limboole/`.

analogies, and finally I have briefly discussed the importance of exercising one's mathematical thinking by acquiring a number of skills polished by the intelligent use of proof strategies. We should always aim at achieving maximal pedagogical effect. The best way of measuring this, however, is probably not by bragging about how well the students succeed in the tasks we assign them with our own tools and measure with our own rulers, but by their later performance in other tasks that demand similar competences. Still, data needs to be collected to test whether our teaching principles are sound and far-reaching.

If arguments aim at convincing, the same effect may be achieved *ad baculum*. We had better not teach our students to resort to sticks, though. Many novice students have serious trouble telling the difference betweeen a proof and, say, a broccoli plantation. Talent and experience are needed for recognizing both successful and failed proofs. In the same way that you should not dare teaching formal grammar to a student that is still being alphabetized, I believe that students had better be repeatedly exposed to real-world proofs before you endeavor teaching them formal logic and proof techniques. The particular tool I have described in the previous section was introduced in blended learning, and used concepts of Cognitive Load Theory in its instructional design. I tend to believe, however, that the teaching principles discussed in this paper can —and should— be carried over to corresponding theoretical investigations, as well as to other tools and learning theories.

References

1 Carlos Caleiro and João Marcos. Classic-like analytic tableaux for finite-valued logics. In *Proceedings of* WoLLIC 2009, volume 5514 of *LNAI*, pages 268–280. Springer, 2009.

2 Carlos Caleiro, João Marcos, and Marco Volpe. Bivalent semantics, generalized compositionality and analytic classic-like tableaux for finite-valued logics. To appear in *Theoretical Computer Science*, preprint available at `http://arxiv.org/abs/1408.3775`, 2015.

3 Nicolaas G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In P. Dybjer et al., editor, *Proceedings of the Workshop on Programming Languages*, pages 865–935. Marstrand, Sweden, 1987.

4 Susanna S. Epp The Role of Logic in Teaching Proof *The American Mathematical Monthly*, 110(10): 886–899, 2003.

5 Georges Gonthier. Formal proof — The Four-Color Theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.

6 Leslie Lamport. How to write a 21st century proof. *Journal of Fixed Point Theory and Applications*, 11(1):43–63, 2012.

7 William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

8 Sara Negri. Proof analysis in modal logic. *Journal of Philosophical Logic*, 34:507–544, 2005.

9 George Pólya. *Mathematical Discovery: On understanding, learning, and teaching problem solving*, volume I. Wiley, 1962.

10 Patrick Terrematte and João Marcos. TryLogic tutorial: An approach to learning logic by proving and refuting. 4th International Conference on Tools for Teaching Logic, pages 233–241.

11 Patrick Terrematte, Fabrício Costa, and João Marcos. Logicamente: A Virtual Learning Environment for logic based on Learning Objects. In *TICTTL 2011*, volume 6680 of *Lecture Notes in Artificial Intelligence*, pages 223–230. Springer, Berlin, 2011.

12 Alan M. Turing. Intelligent machinery. Technical report, National Physical Laboratory, 1948. 22 pages.

13 Daniel J. Velleman. *How to Prove It: A structured approach*. Cambridge University Press, 2nd edition edition, 2006.

14 Shlomo Vinner. The role of definitions in the teaching and learning of mathematics. In David Tall, editor, *Advanced Mathematical Thinking*, volume 11 of *Mathematics Education Library*, pages 65–81. Springer, 1991.

15 Freek Wiedijk. Formal proof — Getting started. *Notices of the AMS*, 55(11):1408–1414, 2008.

Easyprove: a tool for teaching precise reasoning

Marek Materzok[1]

1 **Institute of Computer Science**
University of Wrocław, Poland
marek.materzok@cs.uni.wroc.pl

Abstract

Teaching precise mathematical reasoning can be very hard. It is very easy for a student to make a subtle mistake in a proof which invalidates it, but it is often hard for the teacher to pinpoint and explain the problem in the (often chaotically written) student's proof.

We present Easyprove, an interactive proof assistant aimed at first year computer science students and high school students, intended as a supplementary tool for teaching logical reasoning. The system is a Web application with a natural, mouse-oriented user interface.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Mathematical logic, Computer science, Proof assistants, Teaching

1 Introduction

A logic course is an essential part of every computer science curriculum. Computer science students in University of Wrocław take a mandatory course, called "Logic for Computer Scientists", during their first semester. The main goal of the course is to teach the students the ability to perform precise mathematical reasoning – which is a very important skill for theoreticians and practitioners alike. To achieve this goal, the students are given a number of exercises which involve proving simple theorems, mostly in the domain of set theory. The students are expected to provide informal proofs in natural language. The emphasis is placed on logic as a tool for convincing yourself and others of the validity of mathematical statements, and not on any particular logical formalism.

The students consider this course to be very hard. During the first few weeks, they struggle to understand the structure of a mathematical proof. The teachers do their best to guide them, but the time a teacher can give to an individual student is necessarily very limited. Therefore there is a strong need for an educational tool to aid the learning process. Such tool would serve as a playground, allowing the student to experiment freely with writing mathematical proofs. It should preferably both constrain the student, so that only correct proofs are accepted, and suggest him possible ways to continue the proof if he gets stuck.

There is a class of computer programs called proof assistants, which aid in formalizing proofs of mathematical theorems; examples include Coq, HOL Light, Isabelle and Mizar. These tools, while very useful for a specialist, are in our opinion too complex and quirky to be used for teaching logic to freshmen. Some have adapted them for use in teaching logic [5, 8, 4, 6]; we believe these tools are useful only for later stages of student's education. There also already exist a number of tools developed specifically as an aid for teaching logic, such as Jape [2], Yoda [7] and Panda [3], but these tools focus mostly on teaching logic as a formalism (natural deduction, sequent calculus, etc.), not as a method of reasoning, as embodied by pen-and-paper proofs which can be found in mathematical journals.

We developed Easyprove as a supplementary teaching tool to be used in our logic course for freshmen at the University of Wrocław; we believe it might also find use in high schools. We briefly summarize our requirements in the next section.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 129–136


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

2 Design goals

In order to be most useful for teaching mathematical reasoning to freshmen, Easyprove was designed with following goals in mind:

Easy to access. A good teaching tool should be easy to access for the student both during classes and at home. Using contemporary IT technology, the best way to achieve this is to design the tool as a web application. The Javascript engines of today's Web browsers allow to create Web interfaces which are just as usable as classical "desktop" ones. One can also argue that young people are now very familiar with Web-like interfaces thanks to the popularity of smartphones and tablets.
The approach of designing an educational tool as a Web app was successfully applied by many other projects, including Yoda [7], BoxProvr [4] and ProofWeb [5].

Natural syntax. To use a full-featured proof assistant, the user has to learn a specialized syntax used for representing terms, proofs and commands for the system. This can be a big burden for a computer science freshman, who is not yet familiar with any programming language or other formal syntax.
To remedy this problem, Easyprove presents proofs using a notation close to natural language, and the terms are displayed using Unicode mathematical symbols, which are already familiar to the student from mathematics classes. The symbols can be entered using a graphical keyboard, LaTeX-like shorthands, or copied-and-pasted from other sources.

Familiar setting. Many proof assistants are based on complex formalisms, which can be hard to understand to freshmen. For example, HOL and Coq use higher-order logics, and it is typical to encode sets as predicates within the logic. This is often advantageous for an expert user, but for a beginner it is causing many unexpected problems: e.g., simple terms like $\{\{a\}\} \cup \{a\}$, which the student encounters on mathematics classes, give a type error.
In contrast, Easyprove is based on first-order logic and (Zermelo-Fraenkel) set theory. First-year students should already have some basic familiarity with them from mathematics classes.

Discoverable user interface. Proof assistants have in general a very steep learning curve. Some of them present a command prompt to the user, who is required to learn a big number of commands or "tactics" in order to use them (e.g. Coq, HOL Light). Others work in batch mode – the user has to write a proof script, which is then checked for correctness by the tool (e.g. Mizar, Isabelle). There are GUI front-ends available, but these are of little help for learning: they usually list all possible commands in menus, the commands are not well described, and most of them are not applicable in a given context.
In contrast, Easyprove presents the user with applicable actions only, and their effects are completely described. Also, every function of Easyprove is accessible by the graphical interface, which can be used with only a mouse.

Features for classroom use. Easyprove is designed with features important for teachers in mind: creating task lists, managing student accounts, storing students' solutions on the server for review and replay. It also supports internationalization: we have implemented Polish and English front-end language versions.

In the next section we present how Easyprove meets these design goals by presenting example interactions with the system.

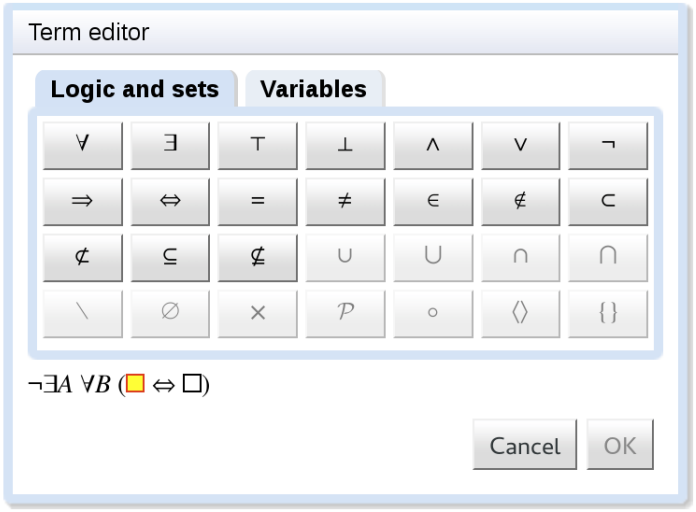


Figure 1 Term editor

3 Using Easyprove

To run Easyprove, one enters its URL in any Web browser with standard CSS and Javascript support. One is then presented with a screen with a list of pre-entered tasks – theorems to prove. One can also create a new task from scratch. To create a task, the user has to give it a name, enter the goal formula, and select the set of assumptions (axioms or lemmas) which will be available in the task.

3.1 Term editor

The goal is entered using a graphical term editor (Figure 1). The editor window has two main parts: on the top is the visual keyboard, which presents available symbols and variable names present in the current context; the currently entered term is displayed below it. In the visual keyboard, each button has a tooltip which displays a short description and presents how a given symbol can be entered using a keyboard.

The editor has two modes of operation: structural and linear. In the structural mode, clicking a button on the visual keyboard fills the currently selected hole in the term, possibly creating new holes for the subterms. When the user starts typing or clicks on a hole, the editor switches to the linear mode. The visual keyboard can now be used to insert a symbol in the current cursor position. Symbols can also be entered by typing a LaTeX-like shorthand, e.g., typing `\forall` causes the symbol ∀ to appear. Pushing the Enter button parses the entered term and switches the editor back to visual mode. If a mistake is made, clicking on a subterm returns it to linear mode for editing.

3.2 Proof editor

The main part of Easyprove is the proof editor. The proof editor screen (Figure 2) consists of the main block, the sidebar and the toolbar. The main block presents the proof in a form resembling a pen-and-paper mathematical proof. The sidebar lists the current goals and the assumptions applicable for the currently selected goal, which is highlighted. There can be

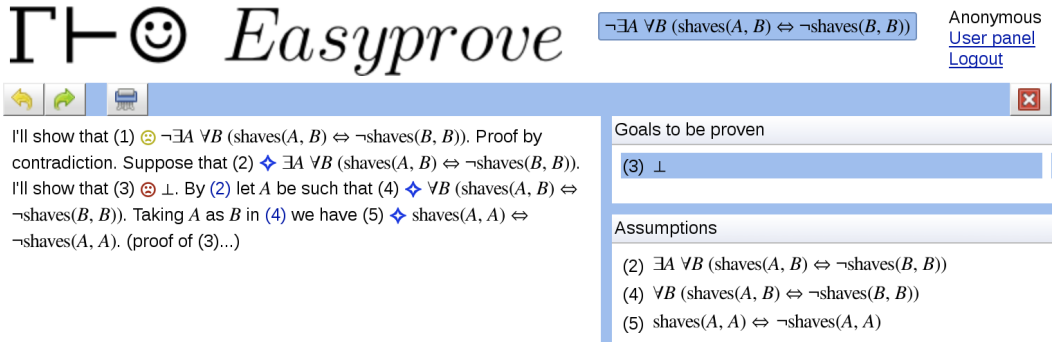


Figure 2 Proof editor

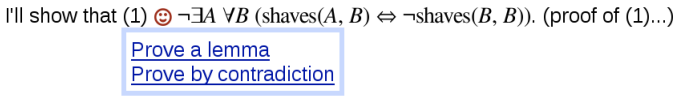
many goals active, one for each branch of the proof created by case analysis or unproved lemmas; the user is free to switch between them at any time. The toolbar contains the undo/redo buttons and a "delete last step in this proof branch" button.

The formulas occurring in the proof text are of one of two kinds: they can be either a goal or an assumption. The currently applicable assumptions are marked with a blue star ✧; the ones from other proof branches are marked with a gray star. Unproved goals are marked with a red sad face ☹; when the goal is not active, but its proof branch is not yet proved completely, the face turns yellow ☹. A proved goal is distinguished by a green happy face ☺. Every formula in the proof text is numbered, so that it can be referred to using this number both in the proof text and in the sidebar.

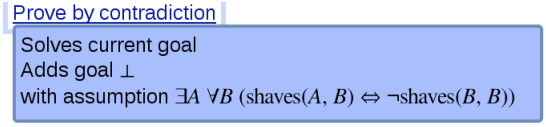
Initially the proof has only one goal – the theorem one wants to prove – and no assumptions (the assumptions selected when creating a task are implicit and are not shown in the proof). To illustrate how one writes a proof in Easyprove, we use the classical barber paradox as an example. In particular, we will prove that the existence of a barber which shaves every person that does not shave himself is paradoxical. This can be stated in first order logic as:[1]

$$\neg\exists A \forall B (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$$

Interaction with the proof editor is inspired by proof by pointing [1]. Clicking on a subterm of the goal (the subterm currently pointed at is highlighted) causes a menu window to appear, which lists proof steps applicable to the selected subterm. In our example, clicking on the whole term opens the window presented below:



The proof step titled "prove a lemma" is always available, and allows to introduce lemmas. The next one, "prove by contradiction", is also always available, and is the one we want to use here. When the user hovers the mouse over a proposed proof step, a tooltip appears:



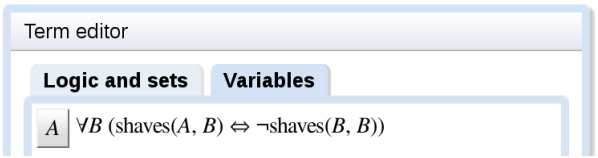
[1] This sentence can be directly copied from this article's PDF file and pasted into the term editor.

The tooltip briefly summarizes how the proof will be extended if the user activates the selected proof step. In this case, the negated goal will be added as an assumption, and the new goal will be $\bot$ – a contradiction.

Clicking on the new assumption will show a proof step named "take this", which corresponds to existential quantifier elimination. Selecting this step will add an assumption about the variable A – which symbolizes the paradoxical barber:

> I'll show that (1) ☹ $\neg\exists A\ \forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. Proof by contradiction. Suppose that (2) ✦ $\exists A\ \forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. I'll show that (3) ☹ $\bot$. By (2) let A be such that (4) ✦ $\forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. (proof of (3)...)

The new assumption (4) is an universal sentence, so a proof step named "specialize" is available for it. Choosing it invokes the term editor, which allows to enter the term used to specialize the universal sentence. Here we intend to examine the case when the person B is the barber A himself, so we choose A from the "Variables" tab (the sentence shown next to the button is the first assumption mentioning this variable):



We get the obviously contradictory sentence $\text{shaves}(A, A) \Leftrightarrow \neg\text{shaves}(A, A)$ (the proof state at this point is shown in Figure 2). Contradiction can be derived from it either by using a built-in lemma about equivalence, or by case analysis using the law of excluded middle; we show below the completed proof using the first possibility:

> I'll show that (1) ☺ $\neg\exists A\ \forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. Proof by contradiction. Suppose that (2) ✧ $\exists A\ \forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. I'll show that (3) ☺ $\bot$. By (2) let A be such that (4) ✧ $\forall B\ (\text{shaves}(A, B) \Leftrightarrow \neg\text{shaves}(B, B))$. Taking A as B in (4) we have (5) ✧ $\text{shaves}(A, A) \Leftrightarrow \neg\text{shaves}(A, A)$. By trivial falsity of equivalence and (5) we have (6) ✧ $\bot$. Contradiction from (6) proves (3).

3.3 Proving theorems in set theory

Easyprove was specifically designed for writing proofs of simple theorems in set theory. To illustrate its features, we now briefly present a proof that for every set A, the union of all its subsets is equal to A; i.e. that $\bigcup(\mathcal{P}(A)) = A$.

Set equality is usually proven using the extensionality principle, or equivalently by proving two inclusions. Both principles are available as lemmas in Easyprove, here we choose the second one:

> I'll show that (1) ☹ $\bigcup\mathcal{P}(A) = A$. By two inclusions it is sufficient to show that (2) ☹ $\bigcup\mathcal{P}(A) \subseteq A \wedge A \subseteq \bigcup\mathcal{P}(A)$. (proof of (2)...)

We get a goal which is a conjunction, so we use a proof step named "prove conjuncts" to prove the conjuncts separately. We can then apply the definition of set inclusion to both goals (the option to do so is present in the drop-down menus for the goals):

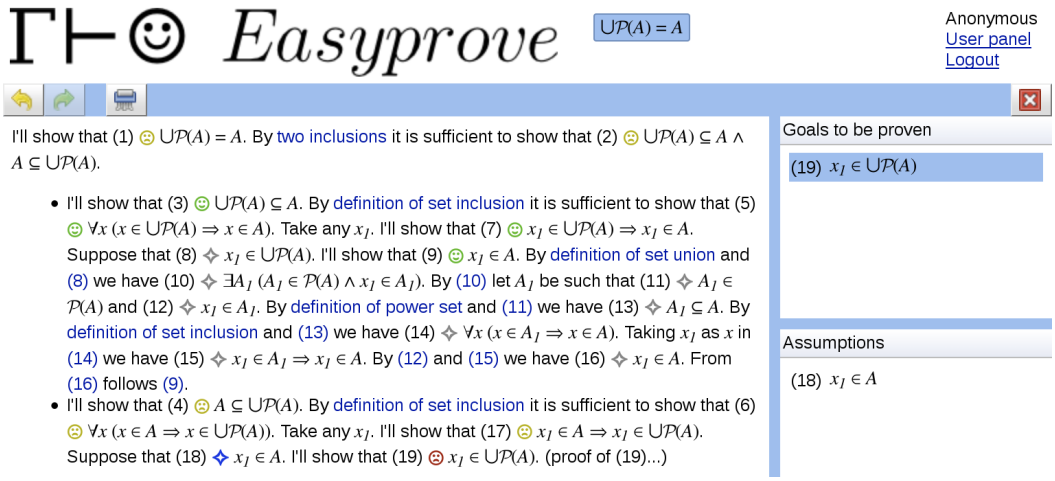


Figure 3 Proof editor – proof in set theory

I'll show that (1) $\cup\mathcal{P}(A) = A$. By two inclusions it is sufficient to show that (2) $\cup\mathcal{P}(A) \subseteq A \wedge A \subseteq \cup\mathcal{P}(A)$.

- I'll show that (3) $\cup\mathcal{P}(A) \subseteq A$. By definition of set inclusion it is sufficient to show that (5) $\forall x\,(x \in \cup\mathcal{P}(A) \Rightarrow x \in A)$. (proof of (5)...)
- I'll show that (4) $A \subseteq \cup\mathcal{P}(A)$. By definition of set inclusion it is sufficient to show that (6) $\forall x\,(x \in A \Rightarrow x \in \cup\mathcal{P}(A))$. (proof of (6)...)

The proof can easily be completed in this way, by repeatedly applying proof steps from logic and using set-theoretic definitions and lemmas. A partially completed proof of the theorem in Easyprove is shown in Figure 3.

4 Implementation

Easyprove is mostly implemented using the programming language Java. The Google Web Toolkit[2] is used for the user interface, for communicating with the Easyprove server, and for compiling Java code to Javascript, which can be executed in a browser. Most of the Easyprove code, including the proof engine, runs on the client side, which reduces the load placed on the server and allows easy scaling for large number of users. The Easyprove server, which manages the task list and user accounts, is implemented in Scala[3], and runs on any Java Servlet container (e.g. Tomcat, Jetty).

The proofs are represented internally as trees, with nodes corresponding to each proof step used by the user. The nodes may be one of two kinds: they may represent forward reasoning steps, which add new assumptions, but do not change the goal; or they can change the current goal or add new goals. Either way, discarding assumptions (weakening) is prohibited – an introduced assumption is valid for its entire proof branch. This corresponds to the way assumptions are usually treated in pen-and-paper proofs. The reasoning rules implemented in the Easyprove system are sound and complete with respect to the classical first-order logic.

[2] http://www.gwtproject.org
[3] http://www.scala-lang.org

The new formulas introduced by proof steps (assumptions or goals) are automatically simplified according to the following rules:

- Expressions with binary logical operators
- Truth $\top$ and falsity $\bot$ symbols are propagated (e.g. $\phi \vee \top$ becomes $\top$) or eliminated ($\phi \wedge \top$ becomes ϕ) if possible; $\vee$ and $\wedge$ are reassociated to the left;
- Negations are pushed inwards using the de Morgan's laws, double negations are eliminated.

The justification is that a mathematician writing a proof is not concerned with the particular shape of the formulas he is working with, but with their meaning.

Assumptions applicable to a clicked subterm on some goal or assumption (see Section 3.3) are found using pattern matching. For example, the definition of set inclusion is stored in Easyprove as:

$$\forall A \forall B (A \subseteq B \Leftrightarrow \forall x (x \in A \Rightarrow x \in B))$$

The system finds this assumption to be applicable to the goal $\bigcup(\mathcal{P}(A)) \subseteq A$ by matching it to the pattern $A \subseteq B$, where A and B are considered pattern variables. To make the search for matching assumptions efficient when the number of them is large, the patterns are stored in a prefix tree (trie) structure, where the keys are pre-order representations of the patterns – e.g. $A \subseteq B$ is stored in the prefix tree as "$\subseteq, A, B$".

5 Conclusions and future work

We have implemented Easyprove, an easy to use proof assistant, as an aid for teaching logical reasoning. It is targeted for first year computer science students and high school students. The system is currently in prototype stage and requires further work in order to be fully usable. The goals for future development are:

Other axiom systems. Easyprove can be in principle extended to handle axiomatic systems other than set theory – for example, Hilbert's axioms for Euclidean geometry. Such an extension would enhance the educational utility of the tool.

Better proof editor. The proof editor does not currently allow easy modification of the proof: the only way to restructure an already entered part of the proof is to undo the most recent proof step in a proof branch. A user interface for restructuring a proof would aid active experimentation, and therefore improve the learning experience.

More use of natural language. Easyprove currently presents formulas in symbolic form. Adding an option to present them in natural language instead (e.g. instead of $\forall x$ shaves(x, y) write "for all x, x shaves y") would help the student make the connection between what he sees on screen and actual pen-and-paper mathematical proofs.

Proof automation. The main priority in Easyprove's design was handling short proofs of simple theorems well, because these are most instructive for teaching logical reasoning. Because of this decision, writing longer proofs in Easyprove is tedious. It remains to be explored how one can include automation in Easyprove without hurting its didactic character.

Mobile interface. Easyprove was designed with a traditional keyboard and mouse in mind. Given the recent popularity of touch screen devices, a redesign of the user interface to accommodate them would certainly allow Easyprove to reach a wider audience.

Nevertheless, the system has proven to be usable in its current state. It was not used in the classroom yet, but it was presented to selected students and their response was positive.

The readers are invited to try out Easyprove. The current stable version of the system is available online at `http://easyprove.ii.uni.wroc.pl/`.

References

1 Yves Bertot, Gilles Kahn, and Laurent Théry. Proof by pointing. In Masami Hagiya and John C. Mitchell, editors, *Proceedings of the International Conference on Theoretical Aspects of Computer Software (TACS'94)*, volume 789 of *Lecture Notes in Computer Science*, pages 141–160. Springer, 1994.

2 Richard Bornat and Bernard Sufrin. Jape: A calculator for animating proof-on-paper. In William McCune, editor, *Automated Deduction—CADE-14*, volume 1249 of *Lecture Notes in Computer Science*, pages 412–415. Springer Berlin Heidelberg, 1997.

3 Olivier Gasquet, François Schwarzentruber, and Martin Strecker. A Proof Assistant in Natural Deduction for All. A Gentzen style proof assistant for undergraduate students. In P. Blackburn, H. van Ditmarsch, M. Manzano, and F. Soler-Toscano, editors, *Proceedings of the Third International Congress on Tools for Teaching Logic (TICTTL'11)*, volume 6680 of *Lecture Notes in Computer Science*, Salamanca, Spain, June 2011.

4 Jonas Halvorsen. Web based GUI for natural deduction proofs in Isabelle. Master's thesis, University of Edinburgh, 2007.

5 Maxim Hendriks, Cezary Kaliszyk, Femke van Raamsdonk, and Freek Wiedijk. Teaching logic using a state-of-the-art proof assistant. *Acta Didactica Napocensia*, 3(2):35–48, June 2010.

6 H. James Hoover and Piotr Rudnicki. Teaching freshman logic with Mizar-MSE. In *Workshop on Teaching Logic and Reasoning in an Illogical World*, Piscataway, New Jersey, July 1996.

7 Benjamín Machín and Luis Sierra. Yoda: a simple tool for natural deduction. In P. Blackburn, H. van Ditmarsch, M. Manzano, and F. Soler-Toscano, editors, *Proceedings of the Third International Congress on Tools for Teaching Logic (TICTTL'11)*, volume 6680 of *Lecture Notes in Computer Science*, Salamanca, Spain, June 2011.

8 Jakub Sakowicz and Jacek Chrząszcz. Papuq: a Coq assistant. In H. Geuvers and P. Courtieu, editors, *Proceedings of PATE'07 International Workshop on Proof Assistants and Types in Education (PATE'07)*, pages 79–96, Paris, France, June 2007.

Presentation of Classical Propositional Tableaux on Program Design Premises*

Juan Michelini and Álvaro Tasistro

Universidad ORT Uruguay
michelini@ort.edu.uy, tasistro@ort.edu.uy

Abstract

We propose a presentation of classical propositional tableaux elaborated by application of methods that are noteworthy in program design, namely program derivation with separation of concerns. We start by deriving from a straightforward specification an algorithm given as a set of recursive equations for computing all models of a finite set of formulae. Thereafter we discuss the employment of data structures, mainly with regard to an easily traceable manual execution of the algorithm. This leads to the kinds of trees given usually as constituting the tableaux. The whole development strives at avoiding gaps, both of logical and motivational nature.

1 Introduction

We teach a course *Logic for Computing* in a Software Engineering programme of studies. Prior to this, students have received courses in Calculus, Algebra and introductory Programming in Java, plus a course called *Foundations of Computing*, which introduces polymorphic, higher-order functions and inductive types with the fundamental methods of induction and recursion in their various forms. *Foundations of Computing* makes emphasis on a mathematical approach to Programming, specifically on correctness proofs. *Logic for Computing*, in turn, concerns itself essentially with the notion of *formal* proof.

It follows from the foregoing that we should be very much interested in making explicit *methods* of proof. By this we mean both general strategies for developing and fully understanding solutions to problems, as well as manners of presenting the corresponding proofs which convey natural, concise and complete justifications of their design. Now, as it turns out, we have observed that some methods that have arisen within what could be called the science of Programming can be employed for obtaining or conveniently presenting mathematical results. This is to our mind a fact to be most welcome, for it exposes a unity of method between Programming and Mathematics that cannot but bring about positive outcomes for both sides, at least in as much the learning and teaching aspects are concerned.

In this paper we present an example of the latter, concerning the presentation of the method of *tableaux*. This is a proof procedure for both propositional and predicate logic dating back to [1] and [2], and whose ultimate variant (termed *analytic* tableaux) has been introduced in [3]. Specifically, what we do is: (1) We derive the method as a set of equations —to be used as rewriting rules— from a straightforward specification, namely the one demanding the computation of the set of all models of the given set of formulas. (2) We discuss the design of data structures for actually effecting and keeping trace of the execution of the method, which leads to the sorts of trees that are called "the tableaux" in textbooks. The first part yields a compact proof of the correctness of the method, much simpler than the ones in textbooks. The second part introduces the convenient and classical notation and establishes its correctness relating it to the set of equations originally given by a simple

* This work was partially supported by ANII–Agencia Nacional de Investigación e Innovación, Uruguay.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 137–146

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

inductive argument. As a whole, the process is one in which we repeatedly employ simple techniques of *program derivation* and *separation of concerns* to obtain a presentation and justification both modular and simpler of the method of tableaux.

The rest of the paper consists of a general background section whose contents is assumed to be taught priorly to the study of tableaux. In section 3 we present the derivation of the equational algorithm calculating the set of all models of given set of formula. In section 4 we discuss the data structures for tracing the execution of the algorithm, leading to the usual presentations of tableaux, after which we finish up with a general discussion. The presentation is to be read basically as a concise course handout, with some explicit considerations of logical or didactic nature.

2 Background

Syntax. It is enough to consider the set of connectives $\{\neg\ ,\ \wedge\}$. Then the set of formulæ is defined as usual, starting out from a denumerable set $\mathcal{V}$ of propositional letters $\mathtt{p}$:

$\alpha, \beta ::= \ \mathtt{p} \mid \neg\alpha \mid \alpha \wedge \beta.$

We use *signed* formulae $\sigma ::= \ S\alpha$ where $S ::= \ \mathsf{F} \mid \mathsf{T}$, as the forms of assertion or judgement.[1]

Semantics. Interpretations belong in $\mathcal{I} = \mathcal{V} \to \mathsf{Bool}$. The *semantic value* of each formula is defined as follows — let A be the set of formulae and (!) and (&&) denote respectively Boolean negation and conjunction:

$$
\begin{array}{l}
[\![\_]\!]- :: A \to \mathcal{I} \to \mathsf{Bool} \\
[\![\mathtt{p}]\!]^i \quad = \quad i\,\mathtt{p} \\
[\![\neg\alpha]\!]^i \quad = \quad !\,[\![\alpha]\!]^i \\
[\![\alpha \wedge \beta]\!]^i \quad = \quad [\![\alpha]\!]^i \ \&\&\ [\![\beta]\!]^i \ .
\end{array}
$$

Using the former we now define *truth* of an assertion (signed formula) in an interpretation. Call $\hat{S}$ the boolean value corresponding to sign S. Then $i \models S\alpha \equiv [\![\alpha]\!]^i = \hat{S}$, which reads: i is a *model* of $S\alpha$, and also: i *satisfies* $S\,\alpha$ or $S\,\alpha$ is *valid in* i. We shall consider *finite* sets Γ of signed formulae and define models thereof (i.e. $i \models \Gamma$) as the interpretations satisfying every formula of Γ.

Truth in an interpretation. It is generally interesting to develop a method for checking truth of signed formulae in an interpretation. If we start with the propositional letters, we get:

$i \models S\mathtt{p} \equiv$ (model of signed formula)
$[\![\mathtt{p}]\!]^i = \hat{S} \equiv$ (semantic function)
$\mathtt{i}\,\mathtt{p} = \hat{S}.$

For the other cases we wish to obtain (structurally) recursive equations. As to negation, writing $\overline{S}$ the sign opposite to S, we obtain:

$i \models S(\neg\alpha) \equiv$ (model of signed formula)
$[\![\neg\alpha]\!]^i = \hat{S} \equiv$ (semantic function)
$!\,[\![\alpha]\!]^i = \hat{S} \equiv$ (negating both sides to isolate $[\![\alpha]\!]^{\mathtt{i}}$)
$[\![\alpha]\!]^i = !\,\hat{S} \equiv$ (opposite sign)
$[\![\alpha]\!]^i = \hat{\overline{S}} \equiv$ (model of signed formula)
$i \models \overline{S}\,\alpha.$

Finally, turning to conjunction:

$i \models S(\alpha \wedge \beta) \equiv$ (model of signed formula)

[1] The use of signed formulae avoids privileging one boolean value over the other and consequently simplifies definitions.

$[\![\alpha \wedge \beta]\!]^i = \hat{S} \equiv$ (semantic function)
$[\![\alpha]\!]^i$ && $[\![\beta]\!]^i = \hat{S}$,
where we seem to get stuck. Indeed, to rewrite the left hand side requires to consider the definition of (&&) and this is not uniform with respect to truth and falsity. Therefore we are led to try instead distinguishing the cases of S:
Case T:
$i \models \mathsf{T}(\alpha \wedge \beta) \equiv$ (calculation above)
$[\![\alpha]\!]^i$ && $[\![\beta]\!]^i = \mathit{True} \equiv$ (definition of &&)
$[\![\alpha]\!]^i = \mathit{True} \mathbin{\mathsf{\wedge}}^2 [\![\beta]\!]^i = \mathit{True} \equiv$ (satisfaction)
$i \models \mathsf{T}\,\alpha \mathbin{\mathsf{\wedge}} i \models \mathsf{T}\,\beta$.
Case F:
$i \models \mathsf{F}(\alpha \wedge \beta) \equiv$ (calculation above)
$[\![\alpha]\!]^i$ && $[\![\beta]\!]^i = \mathit{False} \equiv$ (property of &&)
$[\![\alpha]\!]^i = \mathit{False} \mathbin{\mathsf{\vee}} [\![\beta]\!]^i = \mathit{False} \equiv$ (satisfaction)
$i \models \mathsf{F}\,\alpha \mathbin{\mathsf{\vee}} i \models \mathsf{F}\,\beta$.
Ultimately, we arrive at the following characterisation of the satisfaction relation:

Signed letter :	$i \models S\mathtt{p}$	$\equiv$	$i\,\mathtt{p} = \hat{S}$
Signed negation :	$i \models S\,(\neg\alpha)$	$\equiv$	$i \models \overline{S}\,\alpha$
True conjunction :	$i \models \mathsf{T}(\alpha \wedge \beta)$	$\equiv$	$i \models \mathsf{T}\alpha \mathbin{\mathsf{\wedge}} i \models \mathsf{T}\beta$
False conjunction :	$i \models \mathsf{F}(\alpha \wedge \beta)$	$\equiv$	$i \models \mathsf{F}\alpha \mathbin{\mathsf{\vee}} i \models \mathsf{F}\beta$.

3 The Set of All Models

We now set ourselves the problem of computing *all models* of any given finite set Γ of signed formulae. This is accomplished by the function
$\mathcal{M} :: \mathcal{P}_{\mathsf{fin}}(\Sigma) \to \mathcal{P}(\mathcal{I})$
$\mathcal{M}(\Gamma) = \{i \in \mathcal{I} \mid i \models \Gamma\}$,
where Σ is the set of signed formulae, $\mathcal{P}$ is the power set operator yielding the set of subsets of given set, and $\mathcal{P}_{\mathsf{fin}}$ does the latter for the finite subsets[3]. Now this straightforward definition presents the inconvenience that, as a method of computation, it obliges to construct all the interpretations and check each of them against the formulae in Γ. We are rather in the search of a syntactic procedure, i.e. one that applied exclusively to the formulae in Γ ends up arriving at the desired set of models. Let us then examine Γ.

First of all, Γ could be empty, which is indeed a plainly uninteresting case. Indeed, every interpretation trivially satisfies the empty set of formulae and so the result in such case is $\mathcal{I}$. So let us assume $\Gamma \neq \emptyset$. If this is the case, then we can pick any one of the formulae σ in Γ and write the latter in the form $\Delta\,|\,\sigma$, which means that $\Gamma = \Delta \cup \sigma$ and $\sigma \notin \Delta$. Given the former, we can now write:
$\mathcal{M}(\Gamma) =$ (split Γ)
$\mathcal{M}\,(\Delta\,|\,\sigma) =$ (definition of $\mathcal{M}$)
$\{i \in \mathcal{I} \mid i \models \Delta\,|\,\sigma\} =$ (satisfaction of a set of formulae)

[2] We distinguish $\wedge$ from $\mathsf{\wedge}$, the first being the conjunction in the object language, the second being the conjunction in the meta-language. Likewise with $\vee$ and $\mathsf{\vee}$.
[3] Our use of "computing" seems at first generous indeed, since we are setting ourselves to generating in general infinite sets of infinite objects. Consider for that matter the case $\Gamma = \{\mathtt{p}\}$. Then any interpretation assigning *True* to $\mathtt{p}$ is a member of the answer set. We shall see later how to settle this issue in detail —the general idea is to give a finite sufficient characterisation of infinite sets of interpretations.

$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \sigma\}$.

The only source of information in the latter expression is the analysis of the form of σ, and so we are led to an examination of cases, i.e. to considering:

$\mathcal{M}(\Delta \mid S\,\mathtt{p}) = \{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models S\,\mathtt{p}\}$,
$\mathcal{M}(\Delta \mid S(\neg\alpha)) = \{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models S(\neg\alpha)\}$,
$\mathcal{M}(\Delta \mid S(\alpha \wedge \beta)) = \{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models S(\alpha \wedge \beta)\}$.

We can make profit of this analysis by using the results obtained at the end of the preceding section for checking the truth of signed formulae in a given interpretation. As it happens, the first case is a bit discouranging, for the satisfiability condition $i \models S\,\mathtt{p}$ takes us to consider the value of $\mathtt{p}$ in the given interpretation, i.e. a semantic rather than a syntactic move. But it pays off to insist. Negation gives the following:

$\mathcal{M}(\Delta \mid S(\neg\alpha)) =$ (inferred above)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models S(\neg\alpha)\} =$ (satisfaction of signed negation)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \overline{S}\,\alpha\} =$ (satisfaction of a set of formulae)
$\{i \in \mathcal{I} \mid i \models \Delta, \overline{S}\,\alpha\} =$ (definition of $\mathcal{M}$)
$\mathcal{M}(\Delta, \overline{S}\,\alpha)$,

where we have used $(,)$ instead of $(\cup)$ for set union. Notice that it is indeed this operation and not the formerly used split $(|)$ which is to be employed in this case, for we do not now know whether the formula $\overline{S}\,\alpha$ belongs or not to Δ. The equation thus obtained, namely

$\mathcal{M}(\Delta \mid S(\neg\alpha)) = \mathcal{M}(\Delta, \overline{S}\,\alpha)$,

is very convenient, for it rewrites the desired set of all models into an expression in which the overall complexity of the formulae has been strictly decreased. The same works for conjunction, whose results with respect to satisfiability can be used by distinguishing the two cases of the sign affecting it:

$\mathcal{M}(\Delta \mid \mathsf{T}(\alpha \wedge \beta)) =$ (inferred above)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{T}(\alpha \wedge \beta)\} =$ (truth of conjunction)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{T}\,\alpha \mathbin{\mathsf{A}} i \models \mathsf{T}\,\beta\} =$ (satisfaction of a set of formulae)
$\{i \in \mathcal{I} \mid i \models \Delta, \mathsf{T}\,\alpha, \mathsf{T}\,\beta\} =$ (definition of $\mathcal{M}$)
$\mathcal{M}(\Delta, \mathsf{T}\,\alpha, \mathsf{T}\,\beta)$.

On the other hand:

$\mathcal{M}(\Delta \mid \mathsf{F}(\alpha \wedge \beta)) =$ (inferred above)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{F}(\alpha \wedge \beta)\} =$ (falsity of conjunction)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} (i \models \mathsf{F}\,\alpha \mathbin{\mathsf{V}} i \models \mathsf{F}\,\beta)\} =$ (distributing A over V)
$\{i \in \mathcal{I} \mid (i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{F}\,\alpha) \mathbin{\mathsf{V}} (i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{F}\,\alpha)\} =$ (trading V for $\cup$ out of set comprehension)
$\{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{F}\,\alpha\} \cup \{i \in \mathcal{I} \mid i \models \Delta \mathbin{\mathsf{A}} i \models \mathsf{F}\,\beta\} =$ (satisfaction of a set of formulae)
$\{i \in \mathcal{I} \mid i \models \Delta, \mathsf{F}\,\alpha\} \cup \{i \in \mathcal{I} \mid i \models \Delta, \mathsf{F}\,\beta\} =$ (definition of $\mathcal{M}$)
$\mathcal{M}(\Delta, \mathsf{F}\,\alpha) \cup \mathcal{M}(\Delta, \mathsf{F}\,\beta)$.

As a result we have so far obtained:

$\mathcal{M}(\Delta \mid S(\neg\alpha)) = \mathcal{M}(\Delta, \overline{S}\,\alpha)$
$\mathcal{M}(\Delta \mid \mathsf{T}(\alpha \wedge \beta)) = \mathcal{M}(\Delta, \mathsf{T}\,\alpha, \mathsf{T}\,\beta)$
$\mathcal{M}(\Delta \mid \mathsf{F}(\alpha \wedge \beta)) = \mathcal{M}(\Delta, \mathsf{F}\,\alpha) \cup \mathcal{M}(\Delta, \mathsf{F}\,\beta)$,

where the case of a signed letter, i.e. a *literal*, could not be included. Now taking a look at the preceding equations for $\mathcal{M}$, we readily realize that the missing case is actually that of a set not containing any composite formulae, i.e. that of a set of literals. Such is the base case of our recursion, since this proceeds by decreasing the size of the formulae of the set being treated —and not the size of the set itself. Therefore it is natural to wonder whether the solution of such base case could actually be just immediate. This is indeed the case,

because there is a straightforward manner of converting a set Γ of literals into the set of all its models. There are two cases:

- ▷ Γ contains pairs of opposite literals. Then it is inconsistent and the set of its models is $\emptyset$.
- ▷ Otherwise the models of Γ are the interpretations that coincide with Γ at the letters mentioned in it.

Formally, call $\hat{\Gamma}$ the set of all models of the set Γ of literals. It is defined as follows:

$$\hat{\Gamma} = \begin{cases} \emptyset & \text{if } \{S\,\mathtt{p}, \overline{S}\,\mathtt{p}\} \subseteq \Gamma \text{ for some } \mathtt{p}, \\ \{i \in \mathcal{I} \mid (\forall S\,\mathtt{p} \in \Gamma)\ i\,\mathtt{p} = \hat{S}\} & \text{otherwise.} \end{cases}$$

Notice that the alternative is decidable and that in the second case the result is sufficiently characterised by the set Γ of literals and so we get a finite representation of it. We can then put together equations for actually computing $\mathcal{M}$:

$\mathcal{M}\,(\Delta \,|\, \mathsf{T}(\alpha \wedge \beta)) = \mathcal{M}\,(\Delta, \mathsf{T}\,\alpha, \mathsf{T}\,\beta)$
$\mathcal{M}\,(\Delta \,|\, \mathsf{F}(\alpha \wedge \beta)) = \mathcal{M}\,(\Delta, \mathsf{F}\,\alpha) \cup \mathcal{M}\,(\Delta, \mathsf{F}\,\beta)$
$\mathcal{M}\,(\Delta \,|\, S\,(\neg\alpha)) = \mathcal{M}\,(\Delta, \overline{S}\,\alpha)$
$\mathcal{M}(\Gamma) = \hat{\Gamma}$ if Γ is a set of literals.

We claim that $\mathcal{M}$ captures the essence of the method of tableaux, and the derivation carried out above gives actually a quite simple proof of its correctness. Nevertheless, its actual execution needs to employ some kind of data structure to record the successive transformations leading to the final result. That is what we turn now to examining.

4 Data Structures for the Tableaux

List of lists. If we ignored the second equation above we would be in presence of a tail-recursive algorithm, i.e. one whose execution could consist merely in successively rewriting the finite set of formulae at hand. We would then do simply with a list of formulae from which we would choose the next formula to be transformed. Now, consideration of the second equation does not in principle introduce any dramatic modification of this situation: it is enough that each application of the equation produces a split of the list from which the formula $\mathsf{F}(\alpha \wedge \beta)$ is taken into two lists, each of it containing exactly one of the two formulae $\mathsf{F}\alpha$ and $\mathsf{F}\beta$ in place of the original one, without any further change.

We illustrate this by means of an example. Suppose we consider $\Gamma = [\mathsf{T}(\mathtt{p} \wedge \neg\mathtt{q}), \mathsf{F}(\mathtt{p} \wedge \mathtt{q})]$, which we already make into a *list* of formulae (as indicated by the use of the $[\ldots]$ notation). Then we may choose say the second formula to proceed, which leads us to employ $\mathcal{M}$'s second equation splitting the original list into two, yielding $[\mathsf{T}(\mathtt{p} \wedge \neg\mathtt{q}), \mathsf{F}\mathtt{p}], [\mathsf{T}(\mathtt{p} \wedge \neg\mathtt{q}), \mathsf{F}\mathtt{q}]$. In the next step we may choose any one of the two occurrences of the only composite (i.e. non-literal) formula. Say we take the left one. The equation to employ is $\mathcal{M}$'s first, yielding $[\mathsf{T}\mathtt{p}, \mathsf{T}(\neg\mathtt{q}), \mathsf{F}\mathtt{p}], [\mathsf{T}(\mathtt{p} \wedge \neg\mathtt{q}), \mathsf{F}\mathtt{q}]$. Of course we may do the same with the right occurrence of the formula just considered, arriving at $[\mathsf{T}\mathtt{p}, \mathsf{T}(\neg\mathtt{q}), \mathsf{F}\mathtt{p}], [\mathsf{T}\mathtt{p}, \mathsf{T}(\neg\mathtt{q}), \mathsf{F}\mathtt{q}]$. We now are left only with the two occurrences of $\mathsf{T}(\neg\mathtt{q})$ to treat, which we must do in two steps using the third equation of $\mathcal{M}$. We write the final result at once: $[\mathsf{T}\mathtt{p}, \mathsf{F}\mathtt{q}, \mathsf{F}\mathtt{p}], [\mathsf{T}\mathtt{p}, \mathsf{F}\mathtt{q}, \mathsf{F}\mathtt{q}]$. Clearly the first set of literals is unsatisfiable which, by the way, we could have noticed some steps earlier, thereby obtaining a less expensive development. The second set is just $\{\mathsf{T}\mathtt{p}, \mathsf{F}\mathtt{q}\}$ and characterizes all the models of the original Γ.

There seem to be three inconveniences as to this execution. The first is that we have treated one and the same formula twice, and that on two different occasions. One readily realizes that the issue is avoidable if the use of the branching equation corresponding to a false conjunction is always subsequent to the use of every other (non-branching) equation formerly applicable. The second inconvenience is that we have rewritten many a formula that was without change. And, finally, the execution is awkwardly traceable —we have

namely indicated the successive steps taken by means of narrative text interspersed in the successive rewritings. The latter is of importance when we consider executions by hand — then a more formal and easily checkable notation would be most welcome by both students and teachers. We shall consider these two remaining issues in the next two subsections, beginning with the latter about an easily traceable notation.

Tree of lists. The straightforward manner of making executions like the former traceable and easily verifiable is just to record the application of each rule, including mention to the formula used. We should therefore begin by naming the equations of the algorithm, say $\mathsf{T}\wedge$, $\mathsf{F}\wedge$, $\neg$ and I in the order in which they are written above. The procedure leads to the deployment of a tree structure whose nodes are lists of formulae as in the preceding section, and whose internal nodes (not leaves) are decorated by labels as explained presently:

0. To begin with, we have only one item, namely the original list of formula. This is of course a tree with only one terminal node (leaf).

1. At each step we choose a composite formula within a leaf (call this leaf $\mathcal{L}$) and apply the corresponding rule as already explained. As a result one or two new lists of formulae are obtained, which are linked to $\mathcal{L}$, becoming successors of $\mathcal{L}$ in the tree. At the same time we label $\mathcal{L}$ with the name of the equation and the formula used.

The leaves of these trees coincide with the lists of formulae obtained by the procedure explained in the preceding paragraph —we have only added a tree structure on top of them for tracing their computation. Therefore, the set of models of the root of the tree obtains as the union of the sets of models of the leaves. Formally, this much becomes clear after the consideration that the union of the sets of models of the leaves, and therefore the invariant just mentioned, are indeed preserved by each application of one equation as described above. Therefore the correctness of the computation procedure using these trees follows by straightforward mathematical induction. The right formulation and proof of this result is left as exercise.

Notice that the preceding description amounts to inductively defining these trees as a family $\mathcal{T}(\Gamma)$ indexed by the finite sets Γ in a manner such that the constructors stand in correspondence with the equations as named above, in the following manner: to internal nodes, constructors $\mathsf{T}\wedge$, $\mathsf{F}\wedge$ and $\neg$ are associated, corresponding to the equation used in each case. The leaves are the as yet untreated nodes or those already formed by literals only. In either case we associate to the leaf the constructor I. Unfortunately, we must skip a detailed explanation for reasons of space.

Tree of formulae. The repetition of possibly large lists of formulae along the trees as introduced in the preceding section can be avoided, e.g. by employing the procedure described in [3]. We describe these less expensive trees as follows. The general idea is to write at each node of the tree different from the root only the formulae originated by the use (decomposition) of another formula. The root of the tree will contain the originally given set (list) of formulae. With this information it is possible to compute the full trees of the preceding paragraph provided the used formulae are recorded at each step, i.e. at each node. Therefore, the correctness of the present method with improved trees will follow from the correctness of the prior method. Specifically, we define the improved trees as follows:

1. Each node will have associated an explicit set E and an implicit set Γ of formulae. E is to be written down explicitly, whereas Γ is to be computed when necessary.

2. For the root of the tree, both E and Γ coincide with the originally given set of formulae.

3. For the other nodes, E will consist of one or two formulae.

4. All internal (i.e. non-leaf) nodes will also have associated one formula, to be called the one *used* at the node.

We now indicate how to extend the tree down from a leaf:

1. A formula in Γ is chosen and written down at the node as its used formula.
2. Then one proceeds according to the form of the chosen formula:
   a. In case it is of the form $\mathsf{T}\,(\alpha \wedge \beta)$ then the tree is extended with *one* child node. For this new node, $E = \{\mathsf{T}\,\alpha, \mathsf{T}\,\beta\}$.
   b. In case it is of the form $\mathsf{F}\,(\alpha \wedge \beta)$ then the tree is extended with *two* children nodes. One of them will have $E = \{\mathsf{F}\,\alpha\}$, whereas the other one will have $E = \{\mathsf{F}\,\beta\}$.
   c. Finally, in case the chosen formula is of the form $S\,(\neg\alpha)$ then the tree is extended with *one* child node having $E = \{\overline{S}\,\alpha\}$.

   For every case of newly created node, the set Γ is computed as follows: If Γ_0 is the implicit set of the parent node, then $\Gamma = (\Gamma_0 - \sigma) \cup E$, where $-$ denotes deletion of a member in a set.

Now, to each improved tree t with a non-leaf root which has associated explicit set E and implicit set Γ of formulae, as well as used formula σ, a full tree of type $\mathcal{T}(\Gamma)$ can be associated, whose constructor is the one corresponding to the form of σ, i.e. $\mathsf{T}\wedge$, $\mathsf{F}\wedge$ or $\neg$, and its children trees are the ones (recursively) corresponding to the children trees of t. If otherwise t is just a leaf, then its corresponding full tree is $\mathsf{I}(\Gamma)$, where Γ is the implicit set of formulae of the leaf in question. This correspondence gives already a method for using the improved trees in order to compute all the models of any given set of formulae. Nevertheless, the following result makes such process easier: The implicit set at each leaf is the union of the explicit sets at the branch ending up at the leaf in question, minus those formulae that have been used on that branch. Thereby one can determine when a branch is *completed*, which happens when the implicit set at the corresponding leaf is a set Γ of literals. Further, then $\hat{\Gamma}$ is the corresponding set of models, and one can then compute the set of models of the whole tree (i.e. of the originally given set of formulae) by taking the union of the sets at each leaf, just as with the full trees.

5 Conclusions

We have put forward a presentation of classical propositional tableaux elaborated by application of some principles that are noteworthy in program design. Foremost among those principles is the one of *separation of concerns*: We have namely started by deriving from a straightforward specification an algorithm given as a set of recursive equations for computing all models of a finite set of formulae. The correctness of the algorithm is brought about hand-in-hand with its derivation by means of a basic inductive argument whose cases are each solved by calculational reasoning yielding identities between sets of interpretations that need not the usual "ping-pong" (or direct-and-converse) argument.

Thereafter we discussed the employment of data structures, mainly with regard to a manual execution of the algorithm. A requirement of natural traceability and verification led us to the trees of sets or lists of formulae presented in [1, 4], the correctness of which is immediate after their derivation as traces of the employment of the original equations. A further improvement avoids repetition of unmodified formulae giving rise to the trees presented in [3], whose correctness is in turn guaranteed by showing that they carry the same information as the former trees.

Smullyan's classical presentation [3] introduces instead the method as a proof procedure for establishing unsatisfiability of (finite) sets of (signed) formulae. The tableaux are given directly in the form of our improved trees of formulae. The proof of correctness is then as usual composed by two arguments, one of soundness and one of completeness, to the

effect that unsatisfiable sets give rise to *closed* tableaux, i.e. one in which every branch contains a contradiction and thus has no model. The proof of soundness is by a quite direct tree induction, whereas the proof of completeness involves showing that an open *completed* branch, i.e. one in which every formula has been fully decomposed, is a Hintikka set. Besides, Hintikka's lemma is proven, to the effect that every Hintikka set has a model.

In our experience, the use of the method as in the classical presentation leads students to the realisation that they either prove the given set of formulae inconsistent or can compute every counter-example (i.e. a sufficient characterisation thereof). Subsequently they tend to ask why we cannot establish such fact as a meta-theoretical result. Our presentation does precisely that —and the correctness of the method as a proof procedure follows as immediate corollary. The idea of computing all models of the given set of formula has led us to give an abstract formulation of the procedure. We then treat as a separate matter the question of the concrete trace of the manual execution of the method. As we have been able to check, this treatment provides the students with improved command over the method, i.e. they exercise a more sound domain over what they are doing and also over the various possible notations or manners of justification they can give thereof.

It could be argued that Smullyan's presentation and proof is scalable to infinite sets of formulae and first-order-logic, and therefore ask about such feature regarding our presentation. Concerning infinite sets of formulae, the first thing to say is that the validity of our equations is certainly not affected. Nevertheless, they cannot of course be interpreted anymore as an algorithm. Even if we assume as usual a principle of omniscience concerning the infinite sets, the method of choice of the formulae to be succesively decomposed by application of the equations is essential for getting the right result. But, as is the case also with the classical presentation, there exist method of orderly choice that guarantee (under the ominiscience principle) the computation of all models and thus the correctness of the method. Generalisation to first-order logic, on the other hand, requires to abandon the idea of "computing all models", replacing it by e.g. "determining whether the set of formulae is or not (un)satisfiable".

We conclude that our presentation may contribute in a better way to the achievement of *profficiency with understanding*, which is our main learning objective. Also it emphasizes design methodology, which we strive to do along and across the whole of the program of studies. It also could be argued that the method is tailored to just students of Computing Science or Software Engineering. We however believe that it can be taught also without much difficulty to Mathematics or Philosophy students and that the advantages we claim to obtain can also be appreciated in such cases. This, however, is yet to be checked out.

Finally, we should like to think of this work as one interpretation and case of the disclosing of the "doing" of Mathematics as advocated by Dijkstra [5]. We have tried to avoid all gaps of both mathematical and motivational nature. To our mind, this case is yet another sample of the unity of structure and method that mathematics and programming share[4]. Exploiting such unity should be fruitful for improving understanding and thus better helping learning.

References

1 Hintikka, K. J. J. *Form and content in quantification theory.* Acta Philosophica Fennica, 8, 7-55, 1955.

2 Beth, E. W. *The Foundations of Mathematics.* North Holland 1959.

[4] And that at a deeper level shows up in the propositions-as-types principle.

3 Smullyan R. M. *First-Order Logic.* Dover, 1994. An unabridged, corrected republication of the work first published by Springer-Verlag, New York, 1968.
4 Ben-Ari, M. *Mathematical Logic for Computer Science.* Springer, 2012.
5 Dijkstra, E. W. *EWD 1059.* http://www.cs.utexas.edu/users/EWD/index10xx.html

RAESON: **A Tool for Reasoning Tasks Driven by Interactive Visualization of Logical Structure**

Ştefan Minică

stefan.minica@gmail.com

Abstract

The paper presents a software tool for analysis and interactive engagement in various logical reasoning tasks. A first feature of the program consists in providing an interface for working with logic-specific repositories of formal knowledge. A second feature provides the means to intuitively visualize and interactively generate the underlying logical structure that propels customary logical reasoning tasks. Starting from this we argue that both aspects have didactic potential and can be integrated in teaching activities to provide an engaging learning experience.

1998 ACM Subject Classification I.2 ARTIFICIAL INTELLIGENCE, I.2.4 Knowledge Representation Formalisms and Methods: Modal Logic, Predicate Logic, K.3 COMPUTERS AND EDUCATION, K.3.1 Computer Uses in Education: Collaborative learning, Computer-assisted instruction, Computer-managed instruction

Keywords and phrases Logic Teaching Software, Reasoning Tools, Interactive Visualization

1 Introduction

The tradition of using diagrammatic representations of formal structure in logical reasoning stretches back to times when sand or papyrus were the media for visualizations. The tradition of using interactive software tools in logical reasoning is a recent refinement of an ancient interest. This stretches back to the age of compact disk distributed software alongside paper printed books, with [1] as a paradigmatic example, and plenty others, many of which have been presented at this venue [3]. Using didactic software online for logic education and e-learning is an equally respectable tradition going back to the age when java applets roamed on the web, with [2] as a representative example, and plenty others, many of which have been also presented at this venue [4]. Latest trends in this evolution are dissemination of logic courseware freely available online, with [5] as a pertinent example, alongside didactic software tools built on a technology stack up to date with current web standards, with [15] as a relevant illustration. This is the general trend for many other fields across the curriculum [13]. When compared to what exists in other fields, many reasoning and modeling techniques emerging in current branches of logic remain in many ways still underrepresented, e.g. modal logics, interrogative logics, epistemic logics, etc. However, consult [16–21] as a short list of notable exceptions containing robust software tools for reasoning and modeling techniques emerging in nonclassical logics that have been recently integrated into this tradition, frequently accompanied by web workers or didactically tailored online demos. Inside this general landscape, the paper will revisit classical challenges for teaching logic and reasoning skills and implement possibly new software tools for interactive reasoning tasks driven by automated visualization of logical structure emerging in logics that are often underrepresented in or overlooked by existing approaches.

Overview The paper is structured as follows: in the first section we introduce the topic and describe the main architecture of the program and some design options. Section 2 describes the use of the repositories of formal knowledge underlying the program. Section 3

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 147–156

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

describes the reasoning tasks and interactive visualization tools provided by the software. Sections 4 and 5 explore teaching use case scenarios and provide evidence for didactic relevance. Section 6 draws conclusions and hints towards possible directions for further work.

The Overall Architecture The rest of this section contains information about implementation and development aspects. Readers only interested in using the program will probably want to skip to Section 2 where the program's features are described.

The Backend Repositories of Formal Knowledge The program consists of two layers: remote repositories of formal knowledge are stored and managed on the server side and the client side consists of the user interface to interact with and visualize the repositories' formal content. The backend application layer is a REST-like API built using NodeJS and Express to provide the usual CRUD operations on repository items. The storage layer consists of several document structured databases built using MongoDB and Mongoose to define repository items [10]. The databases contain items from the standard domains in a logic curriculum: propositional logic, modal logic, predicate logic as well as quantified modal logic. Database entries are JSON documents with each item containing a basic ASCII representation of its content, as received from an input device [11], together with a set of additional representation formats (MathJax-LaTeX, D3-SVG, Unicode, etc.) used in visualizations, interactive reasoning tasks and natural language information. The left and middle panels in Figure 1 illustrate intuitively the list representation and the item features for the formulae library database. Section 2 contains further information about how this data is managed via CRUD operations on the REST-like API. In total, the databases contain over five hundred items from four logic domains. Moreover, besides the predefined, ready to be used items, and perhaps more importantly, the backend application layer also serves as a tool for students or instructors to add items of their own interest to the database and use the reasoning and visualization tools in an approach to learning or teaching that fits their needs. The right panel in Figure 1 shows the formula editor GUI, details on how the GUI maps to CRUD actions are in Section 2. The server contains mainly databases of formulae libraries and models corpora in the four logic domains mentioned and a repository of interactive reasoning tasks (dialogic/semantic games, etc.) using both formulae and models.

The Frontend User Interface The client side layer of the program uses HTML5 and CSS3 for structure respectively presentation and JavaScript for handling application logic as well as for implementing logic specific functionality. The framework used to build the frontend application layer is Backbone enhanced with Marionette modules, together with their standard dependencies JQuery and Underscore [8]. For the graphical user interface we use Bootstrap and for rendering formulae in a browser MathJax. For displaying logical structure we use SVG and for interacting with reasoning tasks visualizations we use D3 [9], particularly tree and force graph layouts. Figure 1 presents the main user interface components and Section 2 describes standard use scenarios for the GUI. Figure 2 illustrates some visualizations of logical structure and Section 3 describes how these can be interactively generated, transformed and explored. The logic specific application component consists of several modules containing functionality handling formal aspects concerning the syntax and semantics of the logics and dealing with their associated reasoning tasks. These include a term unification package and an expression parser [12], a module for translating syntax into D3 tree layout, a tableau based theorem prover [6] with a corresponding module to translate the analytic tableau proof trace into a D3 force graph layout with a hierarchical tree structure, and a (counter)model generator [7] with a corresponding module for translating the (closed) tableau branches into (counter)example relational structures and further into a D3 force graph layout.

Further information about implementation details and the technology stack used are

included in the electronic version of this paper and in the documentation available on [22]. The program was developed and tested using primarily Chromium in an Ubuntu desktop environment and will run optimally in WebKit-based browsers and using a desktop format factor. A live demo of the program is openly available online at [22].

2 Managing Repositories of Formal Knowledge

A first feature of the program consists in providing an interface for accessing and working with the following logic-specific repositories of formal knowledge:

1. *Formulae Libraries* provide an interface for accessing databases containing syntactic items from various logical languages. Currently, these include: Propositional Logic, Modal Logic, Predicate Logic, Quantified Modal Logic.
2. *Models Corpora* provide an interface for accessing databases containing semantic items associated with various logics. These are, depending on domain: truth assignments, relational structures, first-order models, QML models.
3. *Games Repositories* provide an interface for accessing databases containing interactive reasoning tasks. These are not entries for separate entities but merely documents associating, formulae and/or models with logic modules.

To facilitate access to repository items and reasoning tasks, the main interface layout has three component regions containing distinct information panels.

The first panel, in the left in Figure 1, is a collection view, its role is to visualize the repository content in an items list format. These can be either a formulae library or a corpus of models from those enumerated above. The second panel, shown in the middle of Figure 1, is an item view, its role is to display item features. These features are specific to each repository type and logic domain and link to various reasoning tasks, additional natural language information or further symbolic representations for a particular repository item selected from the items list panel. The third panel, in the right of Figure 1, is also an item view. It will display, depending on the chosen feature, the specific visualization content and/or an interface for interaction. This can be, fore example, a SVG canvas, a editor form, a table, etc. The specific details of each interactive reasoning task are described in Section 3. The subviews are interlinked and in a usual workflow will be navigated from left to right by first choosing an item of interest from a collection then a specific feature and finally working with its interactive visualization content in the working details panel. The main navigation menu also provides links to instances of the three layout regions. More detailed descriptions of possible workflows applied to concrete examples are presented in Section 4.

Working with the Formula and Model Editors The first level of feedback guided interaction with logical structure takes place starting from syntactic aspects and is mediated by the formula and model editors. The right side panel in Figure 1 illustrates the editors' user interface. This includes examples of well formed entities for each logic domain and detailed information about the formation rules. Feedback is provided as the user types and parser errors are displayed whenever an incorrect string is entered. A modular parsing expression grammar is used for input validation. This extends naturally from logical fragments to more expressive logics, from object language to the specification metalanguage, and from formulae to model syntaxes. Section 7 contains an illustration of the the grammar modules. A concrete editor use case example is given below in Section 4. Further details are included in the electronic version of this paper and in the program documentation available on [22].

Working with Repository Item Features A second level of feedback guided interaction with logical structure is mediated by reasoning tasks visualizations, accessed through

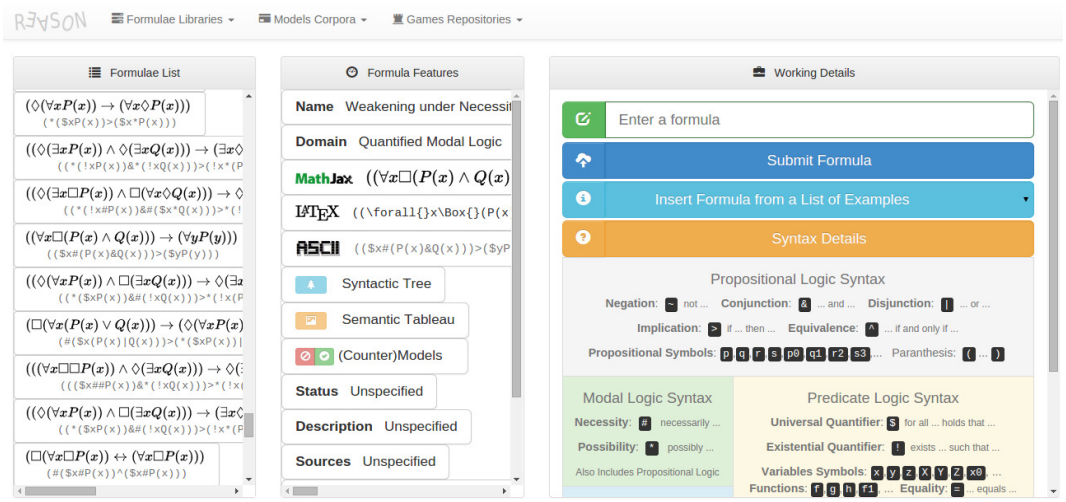


Figure 1 Graphical user interface components: library view, formula features and formula editor

specific item features. The main feature categories are:

1. *Reasoning Tasks* is the main category, it contains various entries such as: visualizing syntactic construction trees, building semantic tableaus, generating countermodels and models, etc. Each task is described in detail in Section 3.
2. *Item Information* contains various categories of information about the chosen items in natural language, such as name, description, sources, domain, etc.
3. *Symbolic Representation Formats* used for reference, linking, translation, etc.

Clicking on a formula feature will display its corresponding working details in the right hand side panel of the main layout presented in Figure 1. While the first two panels are used for organization and navigation of formal repositories, the working details panel is used to host and manage most of the interactive aspects by coupling interface actions with a feedback loop allowing the user to generate and/or explore the formal structure underlying and propelling various logical reasoning tasks.

3 Interactively Visualizing Reasoning Tasks

The main interactive feature of the program is visualization of logical structure. This takes place for several kinds of analysis and reasoning tasks specific to each logic domain. In this section we describe how to use the main reasoning tasks.

Syntax Construction Trees A first analysis task is the interactive generation of syntactic construction trees. This uses the syntactic formation rules of each logic domain to generate a tree layout of the formula and all its subformulae, all the way down to its atomic components. The left side of Figure 2 illustrates a syntactic tree layout generated in this fashion for a first order logic formula. The interface distinguishes between nodes containing fully extended components and the nodes containing further analyzable components. The fully extended tree branches and the nodes containing atomic syntactic elements are marked with a blue color while the not yet fully analyzed nodes are colored in red, the user can further extend these nodes by clicking on their content. In the illustration the second branch is not yet fully decomposed, this is marked in red and can be interactively extended further by clicking on the terminal subformula. Concrete teaching context are included in Sections 4, 5.

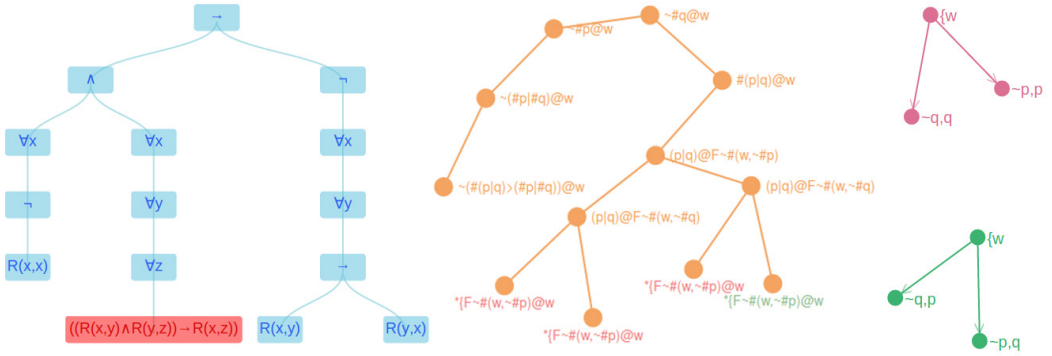


Figure 2 Examples of interactively visualizing logical structure in reasoning tasks: syntactic construction tree (left), semantic tableau proof trace (middle), generated counterexamples/models (right)

Semantic Tableaus A second analysis task is the interactive generation of semantic tableau proofs. This uses the semantic decomposition rules specific to each logic domain to generate a force tree layout by analyzing the head formula in a branch and so on in all resulting tableau branches, all the way down to exhausted branches. The middle region of Figure 2 illustrates a semantic tableau generated in this way for an example formula from the modal logic domain. The interface distinguishes between nodes representing fully expanded branches and the nodes standing for branches that can be further expanded i.e. contain formulae that can be further analyzed. The layout also distinguishes between closed, i.e. containing contradictory formulae, and open branches, i.e. fully expanded but not containing a formula and its negation. Nodes are labeled by the head position formula in the branch, this is the formula logically analyzed and decomposed in that expansion step. The user can interactively display the full content of a tableau branch by hovering over any node of the tableau. Branches which are not fully expanded yet can be extended by clicking on a terminal node. This triggers the process of logical decomposition on the head formula in the branch and adds the resulting branch(es) to the tableau. Clicking on an expanded node reverts the tableau expansion step and removes the outgoing branch(es). For modal logics in addition to decomposition rules for formulas in the logical object language the tableau construction process also uses a specification metalanguage and corresponding decomposition rules for hybrid logic with nominals. These decomposition rules are based on applying metanominals to possible worlds as Skolem terms constructed from formulae containing modal operators and using relational properties of accessible successors during tableau expansion. The theoretical background behind this technique has been described in full detail in [6], and the logical grammar used for parsing hybrid logic formulae in the specification metalanguage is also included for reference in Section 7. Concrete teaching context are included in Sections 4, 5.

Countermodel Generator A third reasoning task consists in generating countermodels for a given formula. This starts from the proof trace generated by a semantic tableau and from each open branch in the terminal nodes of the tree layout extracts a corresponding model. Hybrid logic formulae with nominals are used for expressing relational components of a model inside the specification metalanguage. These are then translated in a force graph layout representing a relational structure which is displayed for visualization and can be further interactively inspected. The right hand side of Figure 2 presents two graphs generated in this fashion. The interface distinguishes counterexamples falsifying the formula, which are colored in red, from models satisfying the formula, which are colored in green. The satisfying

models are obtained in a similar way starting from a complementary formula. Each node of the graph is labeled with a relevant extensional component, for modal logic this is the world's propositional valuation. The user ca obtain further information about the graph by hovering over the nodes. This will reveal the metanominal corresponding to each possible world. This nominal is a Skolem term generated during tableau construction and contains further data about the stage in the proof trace it was introduced and the formula and other metaterms it was generated from. Concrete teaching contexts are included in Sections 4, 5.

Several additional reasoning tasks are described and/or exemplified in Sections 4 and 5, in the extended version of this paper and the documentation available on [22]. These reasoning tasks are either specific to a particular logic domain, such as Truth Tables for Propositional Logic, or only relevant for corpora of models, such as Model Inspection etc.

4 The Didactic Touch

In this section we will present a series of concrete use case scenarios: in order to illustrate the way in which the program can be useful for didactic purposes.

Use Case Scenario 1: Exploring Formal Knowledge Repositories This use case will step through a typical repository navigation. Step (1): From the Main Navigation Menu select Formulae Libraries > Modal Logic, alternatively, you can click on the Modal link from the Go to a Formula Library green div in the Items List panel in the left of the main layout. The left side panel will display the content of the Modal Logic Library. Step (2): Search in the list the formula named Distribution Axiom (K), alternatively, you can hit ctrl+f and type the formula's ascii content `(#(p>q)>(#p>#q))` or its name, as a shortcut to finding the formula faster (using native browser search). When you click on the formula, the middle panel will display the formula's features. Step (3): Inspect the list of available features for the formula listed in the middle panel and find either the Name or the Domain item. When you click on the chosen feature the right hand side panel will display additional information about the item of interest. In this case this will be additional information in natural language about the formula. Item features related with reasoning tasks are explored in the next use case.

Use Case Scenario 2: Interactively Visualizing Reasoning Tasks This use case will present step by step a possible scenario of exploring reasoning tasks. Repeat steps (1) to (3) from the previous use case, but this time use `(#(p|q)>(#p|#q))` to find formula named Distribution Axiom (F), note the (F) at the end which distinguishes it from the previous example. Step (4): Click on the Syntactic Tree feature and explore the construction tree by clicking on the nodes of the tree. Note that the syntactic construction of the formula is the same with the one displayed in the previous example (both formulae generate the same tree structure). Step (5): Click on the Semantic Tableau feature and fully expand all the tableau branches by clicking on the nodes of the tree layout. By hovering over the nodes in the tableau you can see the entire structure of the branch containing all the extended formulae up to that point. Continue until all the formulae are logically decomposed and every branch of the tableau is exhausted. Note that the semantic tableau constructed in this case is different from the one for the previous formula. Find the green colored open branch that makes the difference and inspect its content, try to imagine a counterexample starting from it. Step (6): Click on the Model Generator feature and scroll the panel of models generated in the Working Details panel until you find one that is colored in red. The nodes in the graph represent possible worlds and are labeled by their propositional valuation. Hovering over possible worlds in the models will reveal their construction nominal, compare it with the open branch found in Step (5).

Use Case Scenario 3: Update Formal Knowledge Repositories In the previous use case scenario you have found a counterexample to a formula labeled as an axiom, now you are going to fix this. If needed, go through steps (1-6) in the previous use case to obtain the open tableau and the countermodel illustrated in Figure 2 (middle, respectively, right). Step (7): Click on the Copy to Sandbox Library feature. The Sandbox Library repository will be displayed in the Items List panel. Find the formula and click the Edit Formula Information feature. The rightmost panel will display a form with editable entries. Step (8): Change formula's name from Distribution Axiom (F) to Distribution Formula (fails for disjunction). Optionally, add a description and sources, then click the Submit Changes button. Step (9): Find the modified formula in the Sandbox Library, click the Edit Ascii Content feature. The Formula Editor will open with the formula's ASCII content available for editing. Step (10): Change all disjunctions in the formula to conjunctions, as indicated in the available instructions panels this is done by replacing all occurrences of | with &. Feedback will be provided as you type, when the new formula is parsed correctly, click on the Submit Formula button. Eventually, repeat Step (8): renaming the formula Distribution over Conjunction and adding the description "This fails for disjunction".

Use Case Scenario 4: Semantic Tableaus Expansion Heuristics Step (1): Use the semantic tableau feature to find a counterexample model falsifying the modal formula $(\Box(\neg p \vee \Box q) \leftrightarrow (\Box\neg p \vee \Box q))$, Theorem (S5) 3 in the Modal Library, using no more than seven expansion steps. Step (2): Can you do this with less than seven clicks? If yes, how many? If no, what syntactic property of the formula can be used to prove this for the general case.

Several other use cases are possible including features such as: interactive visualization and inspection of models, linking formulae with structures through model checking tasks, playing dialogic or semantic games, etc. More use cases are described and explored in the electronic version of this paper and in the program documentation available on [22].

5 Classroom Settings Using the Tool

In this section we describe several possible scenarios of using the program in a classroom setting and the didactic experience that has been accumulated. The program can be used both during live classroom activities and also remotely by students. The live teaching classroom settings can integrate the tool alongside traditional teaching methods also during theory exposition but mostly for solving exercises. The following list describes typical exercise templates (concrete examples can be randomly selected from repositories):

Exercise Model One Given: A list of several formulae in symbolic notation (MathJax-LaTeX) and a matching list of formulae represented as syntax construction trees (D3-SVG). Required: Establish the correct correspondence between the items in the two given lists.

Exercise Model Two Given: A list of structures in extensional notation (MathJax-LaTeX) and a matching list of models represented as labeled graphs (D3-SVG force layout). Required: Establish the correct correspondence between the items in the two given lists.

Exercise Model Three Given: Syntax construction trees of several formulae with matching structure graphs (pointed in the case of modal logics, i.e. with designated actual worlds). Required: Decide for each pair if the (pointed) model satisfies or not the given formula.

Exercise Model Four Given: A list of formulae in symbolic notation (MathJax) and a matching list of (pointed) models in extensional representation (not as D3-SVG graphs). Required: For each formula in the list select the (points)/structures that satisfy/falsify it.

Exercise Model Five Given: A list of (pointed) models represented in graphical format (D3). Required: For each structure in the list upload into the formulae library M formulae that

are true/false in the model (at the actual world) and having (modal) depth at least N.

Exercise Model Six Given: A list of (pointed) models represented in extensional format. Required: For each structure in the list upload into the formulae library N formulae that are false/true in the model (at the actual world) and having (modal) depth at least M.

Exercise Model Seven Given: A list of formulae represented as syntactic trees (D3-SVG). Required: For each formula either upload into the corresponding model corpus a counter-example structure with at least N possible worlds(/objects) or claim validity otherwise.

Exercise Model Eight Given: A list of formulae represented symbolically (MathJax-LaTeX). Required: For each formula either find and write down in extensional form a counter-example structure with at least M possible worlds(/objects) or claim validity otherwise.

The previous exercise models can be used in classroom activities, progress evaluations and final exams. To asses the didactic relevance of the program a nonequivalent groups design can be used to measure the average grades in student groups that used the program or not. Another measure can be the response time for accurate answers. Early adopters have (only recently) started to use the program in teaching activities[1] and relevant data can be obtained by logging user interaction based on the previous list of exercise templates. As the design of the exercises suggests, the main testing parameter is the usefulness of interactively visualizing logical structure in reasoning tasks. Preliminary results [14] have been used so far only for internal feedback loop. These show that the test group significantly outperforms the control group for exercise model one, which is as expected. For exercise model three the grades average is also better in the test group, which is also not surprising. The really interesting result is that the test group outperforms the control group also for exercise model four (no visualizations). Moreover, when comparing averages for exercise model three with those for exercise models four, inside the same group, the former significantly outperform the later. This is even more relevant as this outcome holds not only for the test group, but also for the control group. All these suggest that interactive visualizations of logical structure in reasoning tasks can be didactically relevant. At the moment, not enough data is available for the remaining exercise models in order to warrant conclusion validity as a rigorous statistical experiment (more results will become available online as more data is collected).

A final didactic experience that deserves to be mentioned is that the tool facilitates types of exercise that might be impractical (if not impossible) to stage using traditional teaching media (like pen and paper or even chalk and a wide blackboard). Consider an example:

Example 1 Given: Rule of consensus, exception 1 from the propositional logic formulae library. Required: Find a counterexample structure using the semantic tableau method.

6 Conclusions

The paper presented a software tool for analysis and interactive engagement in various logical reasoning tasks. The main feature of the program is to provide the means to intuitively visualize and interactively generate the logical structure that propels customary logical reasoning tasks. A second feature consists in providing an interface for working with logic-specific repositories of formal knowledge. Starting from this we showed how both aspects have didactic potential and can be integrated in teaching activities to provide an engaging learning experience. Overall, this creates an ecosystem of interactive software tools modeling reasoning tasks driven by automated visualization of logical structure emerging in logics that are often underrepresented in or overlooked by existing approaches. We also illustrated

[1] Dr. Adrian Luduşan is using RƎ∀SON in teaching activities for logic courses at "Babeş-Bolyai" University

how this approach can be useful for didactic purposes. Besides presenting a possibly new logical reasoning software suite, the paper also revisited classical challenges for teaching logic and reasoning skills using a completely open source software stack for development, testing, deployment and hosting. The resulting program is openly available online [22], it is implemented completely platform independent and can be run in any modern browser.

Topics of further work include adding relational theories to the modal metatheory and customizing the modal tableau prover. Extending the modeled reasoning tasks to currently uncovered logic domains. Adding additional layers of interactivity to the interface and a customizable strategy for tableau expansion. Adding dynamic logic elements, etc.

7 Logical Syntax Grammars

Listing 1 Atomic Grammar Components (common for both Formulae and Model syntaxes)

```
propsym "proposition" = chars:([p-s0-9_\-]+) {return chars.join("")}
nominal "nominal" = chars:([i-k0-9_\-]+) {return chars.join("")}
metanominal "metanominal" = chars:([lt-w0-9_\-]+) {return chars.join("")}
predicate "predicate" = chars:([P-SA-EMN0-9_\-]+) {return chars.join("")}
function "function" = chars:([f-h0-9_\-]+) {return chars.join("")}
variable "variable" = chars:([X-Zx-z0-9_\-]+) {return chars.join("")}
constant "constant" = chars:([a-emn0-9_\-]+) {return chars.join("")}
```

Listing 2 Formula Syntax: Complex Grammar Components

```
start = metaformula / formula
formula "formula" = atomic / unary / binary / quanty / metaterm2form
metaformula "metaformula" = formula "@" metaterm
quanty "quantified" = "(" quantifier variable formula ")"
quantifier "quantifier" = "$" / "!"
binary "binary" = "(" formula "&" formula ")" / "(" formula "|" formula ")"
                / "(" formula ">" formula ")" / "(" formula "^" formula ")"
unary "unary" = "~" formula / "#" formula / "*" formula
atomic "atomic" = nominal / propsym / equality / predicate "(" term ")"
                / predicate "(" term "," term ")"
equality "equality" = "(" term "=" term ")"
term "term" = constant / variable / function "(" term ")"
            / function "(" term "," term ")"
metaterm "metaterm" = metanominal / metafunction "(" metaterm "," formula ")"
metaterm2form "metaterm2form" = "{" metaterm // reductionaxiom = "<" atomic
metafunction "metafunction" = "F*" / "F~#"
```

Listing 3 Model Syntax: Complex Grammar Components

```
start = model
model "model" = left:world ";" right:model / world
world "world" = metanominal ":" extensionnotationslist
extensionnotationslist "extensionnotationslist"
  = left:extensionnotation right:extensionnotationslist / extensionnotation / ""
extensionnotation "extensionnotation" = valsymb valuation / relsymb relations
  / domainsymb domain / monadicsymb monadicext / diadicsymb diadicext
extsymb "extsymb" =  valsymb / domainsymb / monadicsymb / diadicsymb
valsymb "valsymb" = "VL"
valuation "valuation" = left:literal "," right:valuation / literal / ""
relsymb "relsymb" = "RL"
relations "relations" = left:metanominal "," right:relations / metanominal / ""
literal "literal" = atomic / "~" atomic:atomic {return "~" + atomic;}
atomic "atomic" = nominal / propsym / equality / predicate "(" groundterm ")"
  / predicate "(" groundterm "," groundterm ")"
domainsymb "domainsymb" = "DO"
domain "domain" = left:groundterm "," right:domain / groundterm / ""
monadicsymb "monadicsymb" = "MP"
monadicext "monadicext" = predicate "{" objectslist "}"
monadicextslist "monadicextslist" = left:monadicext "," right:monadicextslist / monadicext / ""
objectslist "objectslist" = left: groundterm "," right: objectslist / groundterm / ""
diadicsymb "diadicsymb" = "DP"
diadicext "diadicext" = predicate "{" pairslist "}"
diadicextslist "diadicextslist" = left:diadicext "," right:diadicextslist / diadicext / ""
pairslist "pairslist" = left: tuple2 "," right: pairslist / tuple2 / ""
tuple2 "tuple2" = "(" groundterm "," groundterm ")"
groundterm "groundterm" = constant / function "(" groundterm ")"
                        / function "(" groundterm "," groundterm ")"
```

References

1 Barwise, Jon and Etchemendy, John and Allwein, Gerard and Barker-Plummer, Dave and Liu, Albert: Language, Proof and Logic, CSLI Publications, Stanford, USA, 2000.

2 Sieg, Wilfried: The AProS Project: Strategic Thinking and Computational Logic, Logic Journal of IGPL, vol. 5, no. 4, pages: 359-368, 2007.

3 van Ditmarsch, Hans and Manzano, María: Proceedings of SICTTL'06, Tools for Teaching Logic, Salamanca, Spain, Logic Journal of IGPL, vol. 5, no. 4, 2007.

4 Blackburn, Patrick and van Ditmarsch, Hans and Soler-Toscano, Fernando and Manzano, María: Proceedings of TICTTL'11, Tools for Teaching Logic, Springer-Verlag, 2011

5 van Benthem, Johan and van Ditmarsch, Hans and van Eijck, Jan and Jaspars, Jan: Logic in Action, Open Logic Courseware Project, 2009-2014.

6 Minică, Ştefan and Khodadadi, Mohammad and Schmidt, Renate A. and Tishkovsky, Dmitry: Synthesizing and Implementing Tableau Calculi for Interrogative Epistemic Logics, in Fontaine, P., Schmidt, R.A., Schulz, S. (eds.), PAAR, The Third Workshop on Practical Aspects of Automated Reasoning, pp. 109-123, Manchester, United Kingdom, 2012.

7 Minică, Ştefan: Dynamic-Epistemic Logic of Questions in Inquiry, Ph.D. Thesis, Institute of Logic, Language and Computation, University of Amsterdam, The Netherlands, 2011.

8 JQuery, available online from: `http://jquery.com/`, Underscore, available online from: `http://underscorejs.org/`, Backbone, available online from: `http://backbonejs.org/`, Marionette, available online from: `http://marionettejs.com/`.

9 MathJax, from: `http://www.mathjax.org/`, Bootstrap, at: `http://getbootstrap.com/`, Scalable Vector Graphics, available online from: `http://www.w3.org/Graphics/SVG/`, Data Driven Documents, available online from: `http://d3js.org/`

10 NodeJS, available online from: `http://nodejs.org/`, ExpressJS, available online from: `http://expressjs.com/`, MongoDB, available online from: `http://www.mongodb.org/`, BENM, available online from: `https://github.com/jkat98/benm`.

11 JSON: JavaScript Object Notation: `http://www.json.org/`, ASCII: American Standard Code for Information Interchange: `http://www.ascii-code.com/`, CRUD: persistent storage functions: `http://en.wikipedia.org/wiki/Create,_read,_update_and_delete`.

12 PEG.js, parser generator for JavaScript, available online from: `http://pegjs.org/`, Unify.js, package available online from NPM: `https://www.npmjs.com/package/unify`

13 Khan Academy, available online from: `https://www.khanacademy.org/a/0z1s`

14 `https://www.academia.edu/12056013/Class_experience_using_RAESON`

15 Modal Logic Playground, available online at: `http://rkirsling.github.io/modallogic/`

16 MetTeL2, Metric Tesselation Logic, a generic tableau prover, demo available online from: `http://www.mettel-prover.org./demo.php`

17 DEMO, a demo of epistemic modelling, code and examples available online from: `http://homepages.cwi.nl/~jve/demo/`

18 LoTREC, logical tableaux research engine companion, demo available online from: `http://www.irit.fr/ACTIVITES/LILaC/Lotrec/webstart`

19 TWB, the tableau WorkBench, a generic tableau framework for building tableau/sequent-based theorem provers for non-classical propositional logics, demo available online from: `http://twb.rsise.anu.edu.au/demolist`

20 InToHyLo: Inference Tools for Hybrid Logics, source code and demos available online from: `http://www.glyc.dc.uba.ar/intohylo/index.php`

21 MleanTAP: A Modal Theorem Prover, at: `http://www.leancop.de/mleantap/`

22 RƎ∀SON, a software tool for interactively visualizing logical structure in reasoning tasks, available online from: `http://raeson.mybluemix.net`

How to prove it in Natural Deduction: A Tactical Approach *

Favio E. Miranda-Perea[1], P. Selene Linares-Arévalo[1], and Atocha Aliseda[2]

1 **Departamento de Matemáticas, Facultad de Ciencias, Universidad Nacional Autónoma de México. Circuito Exterior s/n, Ciudad Universitaria C.P 04510, México D.F.**
favio@ciencias.unam.mx selene_linares@ciencias.unam.mx

2 **Instituto de investigaciones Filosóficas. Universidad Nacional Autónoma de México. Circuito Mario de la Cueva s/n, Ciudad Universitaria, C.P 04510, México D.F.**
atocha@filosoficas.unam.mx

Abstract

The motivation for this paper comes out of our experience with teaching natural deduction (ND) and with the way this formal system is implemented by the Coq proof assistant, namely by means of so-called tactics, which are heuristics that transform a goal formula into a sequence of subgoals whose provability implies that of the original formula. We aim at capturing some of these tactics into a system of ND for minimal logic. Our goal is twofold: formal and didactic. The former delivers a formal system with its underlying heuristics to build proofs, which in turn serves our latter purpose, that of making an ideal system for the teaching of ND at an undergraduate level in a computer science program.

1 Introduction

The importance of logic in mathematics and computer science is unquestionable. The use of proof-assistants, whose kernel are implemented logics, in the verification and certification of software and mathematical proofs is coming of age. A proof-assistant is a computer system that consists of a domain-specific language representing logical objects, as well as definitions and theorems about these objects, together with a mechanism that allows for the interactive construction and validation of proofs. Though proof assistants have been used to teach logic (e.g. [9]), we think there is still a gap between the traditional way of teaching deductive systems, in particular natural deduction (ND from now onwards), using [2] for example, and the use of proof-assistants to solve tasks of software construction and verification, like the ones tackled in a programming language foundations course (c.f. [7]). This paper claims to be a contribution to filling this gap.

Our main goal is to teach ND to undergraduate computer science students in the way this formal system is implemented by a proof-assistant, specifically Coq (`http://coq.inria.fr`). This way, the migration from studying logic, to using it, for example in a programming languages foundations or a formal verification course, will be smooth, as the transition will be mainly syntactical. We pursuit this goal from the theoretical side, that is, we do not discuss here how to use Coq to teach logic. To the best of our knowledge, our approach is novel, for we capture Coq's native proof search mechanism for first order logic by giving a formal definition of its procedural way of constructing proofs, which corresponds to what we may call "usual mathematical reasoning".

* This research has been supported by project UNAM-PAPIIT (IN400514-3).

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 157–166
Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

2 Natural Deduction

ND is, as the name says, a formalism that captures natural reasoning, as opposed to the formal reasoning given by Hilbert's axiomatic systems, recall for example the derivation of $A \to A$ in such a system. The flavor of ND we choose is one with localized hypotheses, called S-systems in [3], for this version is more akin to computer science applications. In these systems, inference rules are not applied over formulas, but over judgments (sequents), which are pairs of the form $\Gamma \vdash A$ where A is a formula and Γ is a set of formulas called the context. The intuitive meaning of such a sequent is that formula A is provable from the hypotheses in Γ. This way, all current hypotheses, i.e. formulae in Γ, are available at every derivation step. Therefore there is no need for a global discharging mechanism[1], unlike Fitch-style ND as presented in [2, 5]. The specific inference rules are:

$$\frac{}{\Gamma, A \vdash A}\ (Hyp)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to I) \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\to E)$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ (\wedge I) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ (\wedge E) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ (\wedge E)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ (\vee I) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ (\vee I)$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ (\vee E)$$

$$\frac{\Gamma \vdash A \;\; x \notin FV(\Gamma)}{\Gamma \vdash \forall x A}(\forall I) \qquad \frac{\Gamma \vdash \forall x A}{\Gamma \vdash A[x := t]}(\forall E)$$

$$\frac{\Gamma \vdash A[x := t]}{\Gamma \vdash \exists x A}(\exists I) \qquad \frac{\Gamma \vdash \exists x A \quad \Gamma,\, A \vdash C \quad x \notin FV(\Gamma, C)}{\Gamma \vdash C}(\exists E)$$

As usual, we have a starting rule or axiom scheme (Hyp); introduction rules (I), useful to prove a formula according to its logical form (i.e., its main connective or quantifier) and elimination rules (E), useful to obtain information from a formula. Note that neither the false constant $\bot$ nor the negation operator $\neg$ are present and therefore we are dealing with minimal logic. The reason is that we are interested in logic from an algorithmic point of view and classical logic departs from this view. Our notion of proof is therefore linear as opposed to a tree form. From a theoretical perspective, this choice is irrelevant, but the linear form is in accord with the way proof-assistants operate.

▶ **Definition 1.** A proof or derivation of judgement $\mathcal{J} =_{def} \Gamma \vdash A$ from the set of judgements $\mathbb{H}$ is a finite sequence of judgements $\Pi = \langle \mathcal{J}_1, \ldots, \mathcal{J}_k \rangle$ such that $\mathcal{J}_k = \mathcal{J}$ and for every $1 \leq i \leq k$ one of the following conditions hold:

[1] i.e. One that involves specific previous parts of the derivation.

- $\mathcal{J}_i$ is an instance of the (Hyp) rule
- $\mathcal{J}_i \in \mathbb{H}$
- For every $1 \leq i \leq k$, $\mathcal{J}_i$ is the conclusion of an instance of some inference rule whose premises are $\mathcal{J}_{l_1}, \ldots, \mathcal{J}_{l_n}$ with $l_1, \ldots, l_n < i$

We say that $\Gamma \vdash A$ is provable, or simply that $\Gamma \vdash A$ holds, if it is provable from the empty set of sequents $\mathbb{H} = \varnothing$.

▶ **Example 2.** Let $\Gamma = \{p \to q \vee r\ ,\ q \to r\ ,\ r \to s\}$, we want to show that $\Gamma \vdash p \to s$ holds. The following is a derivation of this sequent[2]:

1	$\Gamma,\ p \vdash p$	(Hyp)
2	$\Gamma,\ p \vdash p \to q \vee r$	(Hyp)
3	$\Gamma,\ p \vdash q \vee r$	$(\to E)\ 1, 2$
4	$\Gamma,\ p,\ q \vdash q$	(Hyp)
5	$\Gamma,\ p,\ q \vdash q \to r$	(Hyp)
6	$\Gamma,\ p,\ q \vdash r$	$(\to E)\ 4, 5$
7	$\Gamma,\ p,\ r \vdash r$	(Hyp)
8	$\Gamma,\ p \vdash r$	$(\vee E)\ 3, 6, 7$
9	$\Gamma,\ p \vdash r \to s$	(Hyp)
10	$\Gamma,\ p \vdash s$	$(\to E)\ 8, 9$
11	$\Gamma \vdash p \to s$	$(\to I)\ 10$

At this point, following definitions and a few examples, most books demand from the students the task of solving more exercises. There is no discussion as to how to master or even perform a derivation process! The very question of *how to prove it* is unanswered and left to the creativeness and luck of the student. There is however a marginal, though remarkable exception in philosophy, which goes back to the Greeks in their study of *analysis* and *synthesis*; the former depicts the backward process of working a mathematical proof, the latter concerned with the forward derivation in a proof. More recently, a modern pioneering work in the study of *heuristics* is [8], in which heuristic strategies and guidance are given to solve mathematical problems. Some other proposals which provide guidance to proof construction are found in a brief discussion in [2], and a much better one in [5].

As the above proof suggests, performing the derivation process is not easy, perhaps due to the rigidity of definition 1. Just note that more than half of the steps are instances of the (Hyp) rule, which is mainly needed to be able to apply $(\to E)$. These steps do not at all reflect a natural way of reasoning and thus make S-systems inconvenient, as opposed to Fitch-style systems.

The question of how to obtain such proof is quite difficult to answer directly, but a clever student can justify it by appealing to the following mathematical reasoning. The formula we have to proof is an implication, therefore we assume the antecedent p and prove the consequent s, which corresponds to steps 11 and 10 in the previous proof. Next we observe that $r \to s$ is part of the premises, so it suffices to prove $\Gamma,\ p \vdash r$ (step 8). Here the application of $(\to E)$ is implicit, but the proof must be explicit and forces us to add step 9. To prove the sequent at step 8, we might try to reason as in steps 10 and 11, using

[2] The last column is not part of the formal derivation, but consists of justifications for every step, according to definition 1.

the premise $q \to r$, which would lead us to seek a proof of $\Gamma,\ p \vdash q$. But after some failed attempts we realize that $q \vee r$ is derivable from the current premises (steps 1 to 3) and that by a case analysis on this formula we can easily obtain r as follows. If q happens then the premise $q \to r$ yields r by $(\to E)$ (steps 4 to 6), and if r is the case, we are done (step 7). Following this reasoning, the student is faced with the problem of constructing the formal proof, which requires that we start with instances of the (Hyp) rule and go forward to obtain what we need. By comparison with the reasoning used to obtain the steps of the proof, this is, quite unnatural. Intuitively, all ND proofs can be constructed by such mathematical reasoning, but it is not clear in general, how to transform this reasoning into a formal derivation. An attempt, that combines both, reasoning forward from the hypotheses and backward from the conclusion, is the intercalation calculus, implemented by the APROS system (`http://www.phil.cmu.edu/projects/apros/`). Our purpose here is not to elaborate further on this, but to put forward a formal notion of backwards derivation, which we call *derivation by tactics* (to be found in definition 5), one that produces proofs equivalent to the usual ones (equivalence proof will be stated as theorem 9) and that captures the procedural reasoning *"understood"* by proof assistants.

Let us start by stating some heuristics.

2.1 Heuristics

The following two strategies can be used systematically to build a proof of a given judgement:

- *Backward reasoning*: To show that $\Gamma \vdash A$ is derivable start by analyzing the judgement and work backwards towards the axioms. To generate a proof, mantain a queue of current goals (sequents whose derivations are to be searched for), which initially includes only the original judgement. Remove a judgment from this queue, and consider a rule whose conclusion is that judgement, adding the premises of such a rule to the queue, as subgoals. This process must be repeated, with the same starting queue, for each possible rule. The process terminates when the queue is empty, meaning all goals have been achieved.
- *Forward reasoning*: To show that $\Gamma \vdash A$ is derivable, start with axioms and work forwards towards the desired judgement. To generate a proof, extend a sequence S of already derived judgements, which is initially empty, by adding to it the conclusion of any rule whose premises are in S. This process generates several sequences and, assuming that all rules are considered to add new sequents, the strategy will eventually find a derivation of the original judgement.

We are interesed in developing a goal-directed system based on backward reasoning that allows for the automation of proof construction, using a mathematical reasoning, as sketched above. To do so we restate our strategy more formally, following the proof strategies discussed in [5].

- Our initial goal is the judgement we want to prove, say $\mathcal{G} =_{def} \Gamma \vdash A$.
- By analyzing Γ and A, we substitute the initial goal $\mathcal{G}$ by simpler subgoals, say $\mathcal{G}_1, \ldots, \mathcal{G}_k$, whose provability implies the provability of $\mathcal{G}$. After the substitution $\mathcal{G}_1$ becomes the *current goal*. This process can be done using one of the following *heuristic rules*:
- *Conclusion Analysis* (CA): The analysis of A leads us to identify its logical form (i.e. its main operator, a connective or a quantifier) and substitute the original goal by the premises of the corresponding introduction rule. Of course, if A is atomic this kind of analysis is not useful.

- *Premise Analysis* (PA): We focus on an specific formula B in Γ which, according to our previous experience and formerly derived judgements, can help us to prove the original formula A. This analysis generates new subgoals either by modifying the context according to the logical form of B or by modifying the conclusion.
- *Lemma Assertion* (LA): Sometimes the current goal is not directly provable by CA and PA, but follows from an intermediate judgement, like a lemma, which is derivable from the current hypotheses.
- A clever combination of the three previous items eventually yields a current goal which is evident (i.e. is an axiom, a hypothesis or an already proved judgement) and therefore discarded from the current sequence of subgoals.
- The process ends succesfully when there are no more subgoals to prove.

A justification for the adequacy of our heuristic rules, will be given in next section. According to the logical form of a conclusion or premise, the heuristics (CA, PA or LA) generate different subgoals. For example, if the current goal is $\Gamma \vdash B \wedge C$, CA yields two subgoals, namely $\Gamma \vdash B$ and $\Gamma \vdash C$, being $\Gamma \vdash B$ the new current goal. Each of these ways of generating subgoals is what we call a *tactic*.

Even if it may seem a bit redundant, we would like to rework example 2 in order to make explicit the application of our heuristic rules: the initial goal is (the judgement in) step 11, and applying CA to this sequent yields step 10; from this PA on the premise $r \to s$ leads to step 8, thus avoiding step 9. To prove step 8, we try PA on $q \to r$, which yields $\Gamma, p \vdash q$. This sequent is not derivable by our own means, but fortunately we realize that, according to the information on Γ, the formula r is a consequence of $q \vee r$, so we can use LA, the lemma being $q \vee r$. This is provable by means of PA on the premise $p \to q \vee r$, this avoids step 2 and yields step 1 as trivial goal, hence the lemma is proved. As we now know $q \vee r$, we can substitute the goal in step 8 by $\Gamma, p, q \vee r \vdash r$, and then use PA on $q \vee r$, which generates two subgoals, namely steps 6 and 7, the latter is a trivial goal whereas the former is consequence of step 4 by means of PA on premise $q \to r$, and thus step 5 is avoided. At this moment, step 4 is the only remaining subgoal, which is trivial and therefore the proof is finished.

Let us present this train of reasoning in a more systematic way, one that gives guidance to the student about *how to prove it*. The following sequence is not a derivation in the sense of definition 1, but a sequence of backward reasonings where step $i+1$ is obtained by applying our heuristic rules CA,PA or LA to the first sequent in step i.

1	$\Gamma \vdash p \to s$	Original Goal
2	$\Gamma,\ p \vdash s$	CA.
3	$\Gamma,\ p \vdash r$	PA($r \to s$).

The current goal is $\Gamma,\ p \vdash r$. As r is atomic, CA is useless; we try PA and find that $q \to r$ seems a natural candidate to continue the process, but this would lead us to a dead end when trying to prove sequent $\Gamma,\ p \vdash q$. Instead, LA comes in action, and generates two subgoals (separated by a semicolon):

4 $\Gamma,\ p \vdash q \vee r\ \ ;\ \ \Gamma,\ p,\ q \vee r\ \vdash r$ LA($q \vee r$).

The current goal is now the lemma $\Gamma,\ p \vdash q \vee r$ and PA yields

5 $\Gamma,\ p \vdash p\ \ ;\ \ \Gamma,\ p,\ q \vee r\ \vdash r$ PA($p \to q \vee r$).

The new current goal is $\Gamma,\ p \vdash p$, which is trivial and can be discarded

6 $\Gamma,\ p,\ q \vee r\ \vdash r$ trivial

The case analysis is now a consequence of $\mathrm{PA}(q \vee r)$. The specific tactic generates two subgoals, each assuming one of q and r. In the first case r is proved by means of $\mathrm{PA}(q \to r)$, and in the second the proof is trivial, for r is itself a premise. This way, all subgoals are proved and the original goal succeeds.

7	$\Gamma,\ p,\ q\ \vdash r$	;	$\Gamma,\ p,\ r\ \vdash r$	$\mathrm{PA}(q \vee r)$.
8	$\Gamma,\ p,\ q\ \vdash q$	;	$\Gamma,\ p,\ r\ \vdash r$	$\mathrm{PA}(q \to r)$.
9	$\Gamma,\ p,\ r\ \vdash r$			trivial.
10				trivial.

The last step indicates that there are no more subgoals left and therefore the proof is finished.

The aim of this work is to capture the above reasoning by means of a formal system. It is clear that this kind of derivation is sequential, meaning that each step depends solely on the previous one. Therefore we can view this process as a transition system.

2.2 A Tactical Approach to ND: Formalization

We are now in a position to formalize the previously discussed heuristic processes by means of a transition system that manipulates sequences of goals (judgements).

▶ **Definition 3.** A transition system is a triple of the form $\mathcal{T} = \langle \mathbb{S}, \mathcal{F}, \rhd \rangle$ where $\mathbb{S} \neq \varnothing$ is a set of states or configurations; $\mathcal{F} \subseteq \mathbb{S}$ is the set of final or terminal states; $\rhd \subseteq \mathbb{S} \times \mathbb{S}$ is a binary relation on $\mathbb{S}$, called the transition relation, such that for every $F \in \mathcal{F}$ and every $S \in \mathbb{S},\ F \not\rhd S$. That is, there are no transitions from final states.

For the sake of clarity, let us say that a goal $\mathcal{G}$ is a judgement, we denote a finite sequence of goals as $\mathcal{S} =_{def} \mathcal{G}_1; \ldots; \mathcal{G}_k$. The set of such sequences is $GSeq$, in particular we have $\square \in SeqG$, where $\square$ represents the empty goal sequence.

The transition system below captures the backward reasoning strategy by formalizing the process of substituting a goal with its corresponding subgoals according to the heuristic rules CA,PA or LA.

▶ **Definition 4.** The transition system of tactics is defined as $\mathcal{T} = \langle GSeq, \{\square\}, \rhd \rangle$. That is, a state is a sequence of goals; the only terminal state is the empty sequence of goals $\square$; and the transition relation $\rhd$ between states is given by the following rules, called *tactics*:

- Conclusion Analysis (CA):
  - `intro`: $\Gamma \vdash A \to B; \mathcal{S}\ \rhd\ \Gamma, A \vdash B; \mathcal{S}$
  - `split`: $\Gamma \vdash A \wedge B; \mathcal{S}\ \rhd\ \Gamma \vdash A; \Gamma \vdash B; \mathcal{S}$
  - `left`: $\Gamma \vdash A \vee B; \mathcal{S}\ \rhd\ \Gamma \vdash A; \mathcal{S}$
  - `right`: $\Gamma \vdash A \vee B; \mathcal{S}\ \rhd\ \Gamma \vdash B; \mathcal{S}$
  - `intro`: $\Gamma \vdash \forall x A; \mathcal{S}\ \rhd\ \Gamma \vdash A; \mathcal{S}$ where w.l.o.g., $x \notin FV(\Gamma)$
  - `exists`: $\Gamma \vdash \exists x A; \mathcal{S}\ \rhd\ \Gamma \vdash A[x := t]; \mathcal{S}$

- Premise Analysis (PA):
  - `apply`: $\Gamma, A \to B \vdash B; \mathcal{S} \ \triangleright \ \Gamma, A \to B \vdash A; \mathcal{S}$
  - `destruct`: $\Gamma, A \wedge B \vdash C; \mathcal{S} \ \triangleright \ \Gamma, A, B \vdash C; \mathcal{S}$
  - `destruct`: $\Gamma, A \vee B \vdash C; \mathcal{S} \ \triangleright \ \Gamma, A \vdash C; \Gamma, B \vdash C; \mathcal{S}$
  - `destruct`: $\Gamma, \exists x A \vdash C; \mathcal{S} \ \triangleright \ \Gamma, A \vdash C; \mathcal{S}$ where w.l.o.g. $x \notin FV(\Gamma)$
- Lemma Assertion (LA):
  - `assert`: $\Gamma \vdash C; \mathcal{S} \ \triangleright \ \Gamma \vdash A; \Gamma, A \vdash C; \mathcal{S}$
  - `cut`: $\Gamma \vdash C; \mathcal{S} \ \triangleright \ \Gamma \vdash A \to C; \Gamma \vdash A; \mathcal{S}$
- Discarding tactics:
  - `apply`: $\Gamma, \forall x A \vdash A[x := t]; \mathcal{S} \ \triangleright \ \mathcal{S}$
  - `trivial`: $\Gamma, A \vdash A; \mathcal{S} \ \triangleright \ \mathcal{S}$.

These tactics are classified according to the heuristic rules of the backward reasoning strategy: the first group is generated by each particular case of CA, according to the logical form of the conclusion; the second group comes from PA, in agreement with the logical form of the focused premise; the process of lemma assertion is given by the third group. Observe that both LA tactics are the same from a logical point of view, but, as we are dealing with goal sequences, they are quite different from an operational point of view. Finally, discarding tactics, that is, those whose application decreases the number of subgoals, are provided by the fourth group.

We now give a brief justification of the four tactic groups: discarding tactics correspond to proving axioms, that is, sequents given by the (Hyp) rule or sequents of the form $\Gamma, \forall x A \vdash A[x := t]$ corresponding to universal instantiation. LA tactics are justified by the so-called substitution property of the ND system: if $\Gamma, A \vdash C$ and $\Gamma \vdash A$ then $\Gamma \vdash C$ (the formula A is the lemma). LA tactics rules require the generation of a new formula (A), a difficult, if not an impossible task for theorem provers to implement the transition system. The operational mechanism of CA tactics is supported by the inversion principle[3] of the introduction rules. Finally, PA tactics are justified by the inversion[4] of the following admissible[5] inference rules which generate subgoals either by simplifying a premise or, in the case of implication, by modifying the conclusion:

$$\frac{\Gamma, A \to B \vdash A}{\Gamma, A \to B \vdash B}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C} \qquad \frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C} \qquad \frac{\Gamma, A \vdash C \ \ x \notin FV(\Gamma, C)}{\Gamma, \exists x A \vdash C}$$

It is now clear that the backward reasoning strategy is captured by the transition system, therefore a derivation by tactics is simply a sequence of transitions ending on the final state $\square$.

[3] That is, the derivability of the conclusion implies the derivability of the premises. This property becomes a theorem when the context Γ is empty.
[4] Again, this is a theorem for $\Gamma = \varnothing$.
[5] That these inference rules are admissible is a direct consequence of the elimination rules (more details in the extended version of this paper).

▶ **Definition 5.** Let $\mathcal{J} =_{def} \Gamma \vdash A$ be a judgement. A derivation of $\mathcal{J}$ by tactics is a sequence of states $\mathcal{S}_1, \ldots, \mathcal{S}_k$ such that $\mathcal{S}_1 = \mathcal{J}$, for every $1 \leq i < k$, $\mathcal{S}_i \rhd \mathcal{S}_{i+1}$ and $\mathcal{S}_k = \Box$. If such a derivation exists, we write $\mathcal{J} \rhd^+ \Box$.

We now show a derivation by tactics of our running example.

▶ **Example 6.** Let $\Gamma = \{p \to q \vee r,\ q \to r,\ r \to s\}$. The following sequence of transitions shows that $\Gamma \vdash p \to s \rhd^+ \Box$. In the right column of step $i+1$ we annotate the name of the tactic that allows for the transition from step i to step $i+1$.

1	$\Gamma \vdash p \to s$	
2	$\Gamma,\ p \vdash s$	intro
3	$\Gamma,\ p \vdash r$	apply $r \to s$
4	$\Gamma,\ p \vdash q \vee r \ \ ; \ \ \Gamma,\ p,\ q \vee r \vdash r$	assert $q \vee r$
5	$\Gamma,\ p \vdash p \ \ ; \ \ \Gamma,\ p,\ q \vee r \vdash r$	apply $p \to q \vee r$
6	$\Gamma,\ p,\ q \vee r \vdash r$	trivial
7	$\Gamma,\ p,\ q \vdash r \ \ ; \ \ \Gamma,\ p,\ r \vdash r$	destruct $q \vee r$
8	$\Gamma,\ p,\ q \vdash q \ \ ; \ \ \Gamma,\ p,\ r \vdash r$	apply $q \to r$
9	$\Gamma,\ p,\ r \vdash r$	trivial
10	$\Box$	trivial

Let us now present a more elaborated example.

▶ **Example 7.** Let $\Gamma = \{(x \vee p) \wedge q \to l,\ m \vee q \to s \wedge t,\ (s \wedge t) \wedge l \to x,\ p \to q\}$. The following is a derivation by tactics of $\Gamma \vdash m \wedge p \to x$.

1	$\Gamma \vdash m \wedge p \to x$	
2	$\Gamma,\ m \wedge p \vdash x$	intro
3	$\Gamma,\ m \wedge p \vdash (s \wedge t) \wedge l$	apply $(s \wedge t) \wedge l \to x$
4	$\Gamma,\ m, p \vdash (s \wedge t) \wedge l$	destruct $m \wedge p$
5	$\Gamma,\ m, p \vdash s \wedge t$;	
	$\Gamma,\ m, p \vdash l$	split
6	$\Gamma,\ m, p \vdash m \vee q$;	
	$\Gamma,\ m, p \vdash l$	apply $m \vee q \to s \wedge t$
7	$\Gamma,\ m, p \vdash m$;	
	$\Gamma,\ m, p \vdash l$	left
8	$\Gamma,\ m, p \vdash l$;	trivial
9	$\Gamma,\ m, p \vdash (x \vee p) \wedge q$	apply $(x \vee p) \wedge q \to l$
10	$\Gamma,\ m, p \vdash x \vee p$;	
	$\Gamma,\ m, p \vdash q$	split
11	$\Gamma,\ m, p \vdash p$;	
	$\Gamma,\ m, p \vdash q$	right
12	$\Gamma,\ m, p \vdash q$	trivial
13	$\Gamma,\ m, p \vdash p$	apply $p \to q$
14	$\Box$	trivial

In order to show how our approach works in first order (minimal) logic, here is an example.

▶ **Example 8.** We give a derivation by tactics of the sequent

$$\vdash \forall v(Pv \to Qv) \to \forall x\big(\exists y(Py \wedge Rxy) \to \exists z(Qz \wedge Rxz)\big).$$

1	$\vdash \forall v(Pv \to Qv) \to \forall x\big(\exists y(Py \wedge Rxy) \to \exists z(Qz \wedge Rxz)\big)$	
2	$\forall v(Pv \to Qv) \vdash \forall x\big(\exists y(Py \wedge Rxy) \to \exists z(Qz \wedge Rxz)\big)$	intro
3	$\forall v(Pv \to Qv) \vdash \exists y(Py \wedge Rxy) \to \exists z(Qz \wedge Rxz)$	intro
4	$\forall v(Pv \to Qv),\ \exists y(Py \wedge Rxy) \vdash \exists z(Qz \wedge Rxz)$	intro
5	$\forall v(Pv \to Qv),\ Py \wedge Rxy \vdash \exists z(Qz \wedge Rxz)$	destruct $\exists y(Py \wedge Rxy)$
6	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash \exists z(Qz \wedge Rxz)$	destruct $Py \wedge Rxy$
7	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Qy \wedge Rxy$	exists y
8	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Qy$;	
	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Rxy$	split
9	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Py \to Qy$;	
	$\forall v(Pv \to Qv),\ Py,\ Rxy,\ Py \to Qy \vdash Qy$;	
	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Rxy$	assert $Py \to Qy$
10	$\forall v(Pv \to Qv),\ Py,\ Rxy,\ Py \to Qy \vdash Qy$;	
	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Rxy$	apply $\forall v(Pv \to Qv)$
11	$\forall v(Pv \to Qv),\ Py,\ Rxy,\ Py \to Qy \vdash Py$;	
	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Rxy$	apply $Py \to Qy$
12	$\forall v(Pv \to Qv),\ Py,\ Rxy \vdash Rxy$	trivial
13	$\square$	trivial

We end by stating the theorem that guarantees that the usual concept of derivation given in definition 1, coincides with our proposed notion of derivation by tactics given in definition 5. The proof of this theorem may be found in the extended version of this paper.

▶ **Theorem 9** (Equivalence of $\vdash$ and $\rhd^+$). *For any sequent $\mathcal{J}$, $\mathcal{J} \rhd^+ \square$ if and only if $\mathcal{J}$ is provable.*

3 Final Remarks

Due to lack of space we were not able to discuss Coq properly. Nevertheless, we want to mention that its underlying first order logic mechanisms can be understood as our transition system $\mathcal{T}$. Actually, the name we give to each tactic in definition 4 is the name of its corresponding Coq command (c.f. [1]). For example, a Coq proof of our running example is given in the following script:

```
Hypotheses (p q r s : Prop)
           (H1: p -> q \/ r)
           (H2: q -> r)
           (H3: r -> s).

Theorem Example1: p -> s.
Proof.
intro.
apply H3.
assert (q \/ r) as H4.
apply H1.
trivial.
destruct H4.
apply H2.
trivial.
trivial.
Qed.
```

The reader may realize that once the tactical approach is understood, this computer-assisted proof is quite clear. Therefore, our goal for a smooth migration from teaching to using logic has been accomplished. Indeed, our backward reasoning strategy (as discussed in 2.1) has been presented to students in a computational logic course, and had a very good reception. At the moment, we will use this material in two of our courses (Computational Logic and Automated Reasoning), where we will test the tactical approach as a way to introduce Coq. The results of this experience will be reported later. The main difference between our approach and that of Coq is that in the latter hypotheses are labeled, in order to make it possible to have further reference to them. If we use these labels as variables of a λ-calculus, as is done by Coq, we can encode the application of every inference rule with a λ-term. From this encoding, the mechanism of derivation by tactics yields a λ-term which encodes such a proof, and we can decode this proof into a usual derivation. This is the well-known Curry-Howard correspondence. Part of our future work is to explore the benefits of this powerful result for teaching logic. With respect to classical logic, we may extend our approach by allowing the use of lemmas of the form $A \vee \neg A$ (without requiring its proof), or by adding, for instance, the tactic $\Gamma, \neg A \vdash B \ ; \ \mathcal{S} \rhd \ \Gamma \vdash A \ ; \ \mathcal{S}$, which corresponds to proving B by contradicting the hypothesis $\neg A$. However, the inversion principles that support our proposal are invalid in this case and therefore we cannot guarantee the feasibility of the tactical approach. A further line of research is to extend our approach to other logics. We are particularly interested in modal logic. In this respect, the second author of this paper (in [4]) has verified in Coq the definition and properties of the ND system for modal logic developed in [6], something which allows for a straightforward definition and implementation of our tactical approach for the case of modal logic.

Acknowledgements

We thank the two reviewers for their comments and suggestions, which helped to improve this paper. We acknowledge the support of the project UNAM PAPIIT IN400514-3.

References

1 Bertot Y. and Castéran P. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. Springer 1st ed. 2004.
2 Huth M. and Ryan M. *Logic in Computer Science: Modelling and Reasoning about Systems.* Cambridge University Press. Second edition, 2004.
3 Indrzejczak. A. *Natural Deduction, Hybrid Systems and Modal Logics.* Trends in Logic. Vol. 30. Springer 2010.
4 Linares-Arévalo P. S. *Deducción Natural en Lógica Modal: una implementación en Coq.* Master's thesis, Universidad Nacional Autónoma de México. 2015.
5 Manzano M. and Huertas A. *Lógica para principiantes.* Alianza Editorial, 2004.
6 Pfenning F. and Davies R. *A Judgmental Reconstruction of Modal Logic.* Mathematical structures in computer science, vol. 11. 511- 540. 2001.
7 Pierce C.B. *Lambda, the Ultimate TA: Using a Proof Assistant to Teach Programming Language Foundations.* In Proceedings of the 14th ACM SIGPLAN International conference on Functional programming. 2009.
8 Polya G. *How to solve it. A new aspect of mathematical method.* Princeton University Press. 1945.
9 Hendriks, M., Kaliszyk, C., van Raamsdonk, F., Wiedijk, F. *Teaching Logic using a state-of-the-art Proof Assistant.* URL: `http://www.cs.vu.nl/~femke/ps/formed08.pdf`.

Tools for teaching logic as reflected in my contribution to the book: The Life and Work of Leon Henkin

Nitsa Movshovitz-Hadar

Department of Education in Science and Technology
Technion-Israeli institude of technology
Haifa, 3200003, Israel
nitsa@technion.ac.il

Abstract

Leon Henkin was a rare combination of a kind human being, a sharp logician and a talented pedagogue. The book this session is focused on [1], mirrors very nicely this combination. In my talk I'll share a few ideas related to the general theme of this conference as reflected in two chapters of the book:

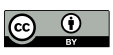
- Tracing Back 'Logic in Wonderland' To My Work with Leon Henkin; and
- Pairing Logical and Pedagogical Foundations for the Theory of Positive Rational Numbers—Henkin's Unfinished Work.

In early 2013 Atara Shriki and I published Logic in Wonderland. This was about 40 years after I started my Ph.D. thesis with Leon Henkin. The time gap did not diminish his influence. Following a few anecdotes from these days, two sample tasks from Logic in Wonderland appear in one of the two chapters. In my talk I'll elaborate on the way we structured the book so as to sweeten the bitter pill of an introduction to logic course, the target audience of which is prospective and in-service mathematics teachers. As for the other chapter, back in 1979 Leon Henkin outlined five different ways of "founding" the mathematical theory of positive rational numbers for further logical development. He prepared them in the form of notes for a future paper, wishing to explore how varying modes of deductive development can be paralleled with varying classroom treatments rooted in children's experience and activities. This dream never came to a full fruition. In my talk I'll present briefly some preliminary ideas about corresponding pedagogical embodiments for each logical development. The next generation of researchers in mathematics-education and curriculum developers are invited to contemplate with them, while bearing in mind that Henkin's work in itself may need additional polishing.

Keywords and phrases Henkin, logic, tools for teaching, mathematics education

References

1 Manzano, M., Sain, I., & Alonso, E. *The Life and Work of Leon Henkin: Essays on His Contributions*. Birkhauser, Switzerland, 2014



4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 167–168

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

An Introductory Course in Logic for Philosophy Students: Natural Deduction and Philosophical Argumentation*

Carlos A. Oller[1] and Ana Couló[2]

1 **Departamento de Filosofía,Facultad de Filosofía y Letras, Universidad de Buenos Aires & Facultad de Humanidades y Ciencias de la Educación – IdIHCS, Universidad Nacional de la Plata**
Argentina
carlos.a.oller@gmail.com

2 **Departamento de Filosofía, Facultad de Filosofía y Letras, Universidad de Buenos Aires**
Argentina
anacoulo@yahoo.com.ar

Abstract

This paper tries to justify the relevance of an introductory course in Mathematical Logic in the Philosophy curriculum for analyzing philosophical arguments in natural language. It is argued that the representation of the structure of natural language arguments in Freeman's diagramming system can provide an intuitive foundation for the inferential processes involved in the use of First Order Logic natural deduction rules.

Keywords and phrases introductory logic courses; natural deduction; argument diagramming; philosophical arguments

1 Introdution

In the 20th century Logic diverged from other philosophical fields by becoming a mature formal discipline. By reaching this stage, Logic, under the form of Mathematical Logic, was able to break away from its ancient interest in natural language arguments and its evaluation. It has been said that the mathematization of Logic implied not just a change in methods, but in the object of study as well. Furthermore, this entailed a pedagogical outcome: it became possible to design a good course in Elementary Formal Logic without ever mentioning the word "argument" in an ordinary sense. A great many Elementary Formal Logic courses consist solely or mainly of First Order Logic and its metatheory, that is, of a predominantly mathematical subject. Consequently, a significant number of philosophy students feel that the compulsory taking of such a course is not only dull and difficult but utterly irrelevant regarding their philosophic training (especially in those countries where analytic philosophy is not the predominant philosophical tradition). However, these complaints are frequently considered out of place, and moreover many logic teachers ignore, do not take into consideration or disregard every pertinent criticism coming from the contemporary fields of Informal Logic or Argumentation Theory [6]. FOL has been shown to be both insufficient and inadequate to address the whole wide field of argumentation, especially philosophical argumentation. Besides, students' protests become reinforced by the current philosophical atmosphere, that

* This work was partially supported by UBACyT grant 01/Q532.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 169–176

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

has been developing a certain mistrust in formal Logic as a trustworthy guide for setting philosophy "on the sure path of science".

In this paper we suggest a way of integrating the study of natural language arguments with FOL in introductory logic courses for the Humanities (especially Philosophy). We propose, firstly, that the relationship between argumentative strategies that are identifiable in natural language arguments and natural deduction rules of inference for FOL should be explicitly addressed. In addition to this we recommend using a diagramming system for argument structure representation as a means to offer an intuitive foundation for the inferential processes involved in using those rules. These steps imply taking into consideration the aforementioned criticisms, and at the same time, offering a plausible justification for the presence of FOL as an introductory level mandatory subject in Philosophy departments.

2 Reasons for teaching FOL as an introductory level mandatory subject to Philosophy students

Several reasons might be given for the presence of FOL as part of the essential basic training for philosophy students. A familiar one is the claim that studying FOL will develop students' abilities in identifying, reconstructing and analyzing natural language arguments. However, both teachers' experience and relevant research seem to disprove this claim.

The underlying idea here is that the evaluation of a deductive argument expressed in a natural language (such as Spanish, French or English) depends fundamentally on the logical form of that argument when it is translated into the artificial language of a logical system (such as FOL). This assumption frequently entails that even if an important part of philosophers' professional activity consists in building, analyzing and evaluating arguments, the theories and procedures that underlay those tasks are seldom, if ever, explicitly addressed in other subjects in philosophy departments and curricula. This job is usually left in the hands of introductory logic courses, which consist mainly in an introduction to deductive FOL. And, correlatively, introductory logic courses rarely take the opportunity to integrate formal logic elements with philosophical problems at large.

However, classic research such as that of Cheng, Holyoak et al.[2], brings attention, at the very least, to the fact that explicit instruction in FOL does not necessarily entail an amelioration of reasoning abilities: "Our results have clear educational implications. We have shown that deductive reasoning is not likely to be improved by training on standard logic." Beebee [1] offers an interesting example of the difficulty of applying formal logic as a means to elucidate the logic behind a philosophical argument. She starts by asking about the relevant goals and contents of a significant introductory formal logic course for a philosophy department. She then sketches an exam item where she asked students "to identify the main rule of inference used in the following argument: If God exists, he is omniscient, benevolent and all-powerful. There is evil in the world. Suppose that God exists. Then, being omniscient, he knows there is evil in the world. Being all-powerful, he could have prevented that suffering. But he has not prevented it. This is incompatible with the claim that he is benevolent. So God does not exist." She found out that only 10 to 15 percent of the students were able to identify *reductio ad absurdum* as the main rule of inference used in the argument, while a significant larger portion could employ it adequately in a formal proof. But this suggests that one of the goals that Beebee states for an introductory formal logic course for first year philosophy students – understanding the logic behind philosophical arguments – cannot be reached just by this means. Beebee's example seems to complement the results reported by Cheng and Holyoak: academic training in mathematical logic not only is not enough to

enhance the ability for deductive reasoning but it is also incapable by itself of enhancing the ability to analyze deductive arguments expressed in natural language.

A second reason for the crucial question about the reasons that might be given for the presence of FOL and its metatheory as an essential part of the basic training for philosophy students has to do with its relevance in enabling students to pursue further studies in Logic, Epistemology or Philosophy of Science that would include advanced courses of Logic as part of the curriculum. But this argument begs the question of the relevance of Logic, and it does not take into consideration the diverging interests of students that would choose other traditions or "streams". Furthermore, Logic, Epistemology and Philosophy of Science majors would certainly need to deepen their knowledge by taking more than one course of Logic, so as to achieve a much deeper level of understanding of this discipline.

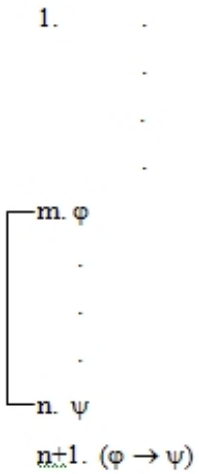
A compromising alternative to our question would be to save part of the course for an informal introduction to logic that includes, as someone has quaintly put it "all those issues that would never be part of a paper in the Journal of Symbolic Logic". This perspective tends to adopt a propaedeutic stance that addresses these issues rapidly and superficially, so as to get to the really important questions. It is easily found in many introductory Logic texts, readings or handbooks for Humanities students. But sadly, most texts do not integrate the "informal logic" and "formal logic" sections, and this can bring about some problems. For instance, the central notions of "argument" and "logical form" that appear in the "informal logic" section usually differ from those appearing in the "formal logic" one. The notion of "argument" in a natural language context – in which philosophical arguments are enunciated- involves a pragmatic conception by reason of which we recognize an argument by identifying in a text the intention to support of the truth or the acceptability of a sentence –conclusion- by the truth of acceptability of other sentence(s) (or argument(s)) –premises. On the other hand, the concept of argument belonging to a formal logic approach is a mathematical notion, devoid of any pragmatic underpinnings: an argument is an ordered pair whose first member is a (possibly empty) set of well-formed formulas (the premises), and whose second member is a well formed formula (the conclusion). Naturally, interpreting an argument in the second sense does not necessarily produce an argument in the first sense, and the translation of an argument expressed in natural language to a FOL language would be incapable of conveying the pragmatic element that is essential for it in the first place.

3 Integrating the teaching of natural language argumentative strategies and natural deduction rules of inference

On the basis of the aforementioned discussion, our proposal for an introductory course in Logic for philosophy students aims at weaving a tighter web between an informal presentation of argumentative strategies typical of philosophical arguments – *reductio ad absurdum*, hypothetical reasoning, reasoning by cases, universal instantiation, etc. – and the FOL rules that codify those strategies. In our experience, familiarity with rules and proof building in FOL does not automatically mean proficiency in detecting those inferential strategies that these rules and proofs translate into philosophical arguments. On the contrary, it seems desirable to start by presenting and examining those strategies by means of philosophical arguments in natural language so as to help students understand the intuitive grounding of natural deduction rules that will be presented in the course section set apart for FOL. Indeed, as we have argued, when building proofs, most students tend to apply FOL natural deduction rules in a mechanical way, without fully grasping their sense.

A good example of this difficulty in understanding the intuitive rules of inference meaning

can be found in the basic rule of conditional introduction. This rule involves reasoning from assumptions or hypotheses and is an essential element in the presentation of logic as a natural deduction system. Also, it is a frequent argumentative strategy in philosophical argumentation. We can represent this logical rule in the usual way using the following diagram:



Students usually learn quickly to build logical derivations that include this rule, by following the advice that recommends to assume the antecedent and try to derive the consequent, whenever one wants to derive a conditional. However, they frequently display difficulties when it comes to the point of having to understand the argumentative strategy that is formalized by that rule if it is embedded in philosophical arguments such as the following: Ryle [8] affirms that the "intellectualist legend" states that an intelligent act is intelligent so far as it is the result of an internal, previous plan that must be intelligent itself for the act to be so.At the same time, planning is a mental operation which, if it is to be deemed intelligent,must itself inherit that characteristic from a previous act of planning: the planning of planning. This previous act, for its part, may be intelligent or stupid. But if it is intelligent, it must itself inherit that characteristic from a previous act of planning, and so and so forth world without end. Since we can find human beings that perform intelligent acts, we can conclude that – under the assumption that the intellectualist legend is right – some human beings are capable of performing an infinite number of mental acts. Therefore, if the intellectualist thesis is true, there is (at least) one human being who is capable of performing an infinite number of mental acts. The difficulties that students display in understanding this argumentative strategy may be due, at least in part, to the fact that in reasoning from assumptions (as we did in the example) the conditional conclusion is supported by a complete (sub)argument rather than by simple statements. But students do not expect this kind of support because it is not considered by the informal definition of argument that is usually given by the textbooks. However, this peculiarity that is formally expressed in the rules that involve assumptions in the presentation of FOL as a natural deduction system – and that intend to reflect an argumentative strategy regularly present in mathematical reasoning (and in other fields as well) – is rarely taught in basic logic texts for the Humanities. Therefore, it is no wonder that many students can be perfectly capable of mechanically applying the conditional introduction rule when building a proof, and, at the same time, do not fully understand the strategy exemplified in Ryle's argument.

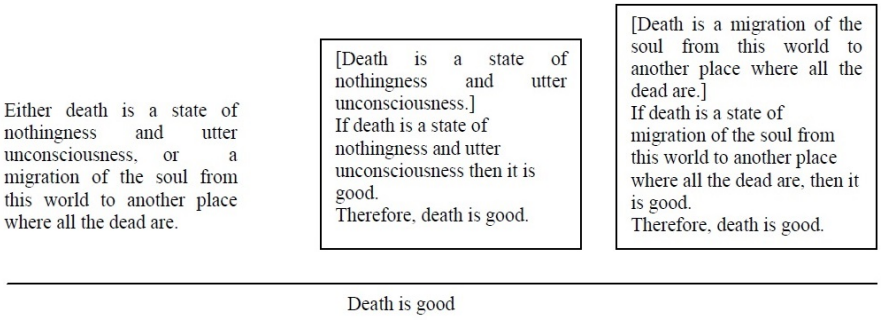
4 Argument diagramming and proofs in a natural deduction system

We have found useful introducing the standard technique of argument diagramming when trying to integrate the study of natural language arguments and the study of arguments formulated in the mathematical logic languages. Argument diagramming allows students to identify and represent the inferential relationships between the sentences that constitute the arguments in natural language, without having to resort to the formalization of those sentences in FOL language. In the philosophical tradition pertaining to the analysis of argument structure, this technique can be found, for instance, in James Freeman's works [3][4].

Even if argument diagramming can be considered to be typical of informal logic strategies in the analysis of argument, it is also closely related to many issues linked to formal logic inferences. In fact, the tree structure typical of the standard argument diagramming, allows students to understand the intuitive meaning of proof building in FOL. Most introductory logic textbooks present FOL proofs using Jaśkowski-Fitch style of natural deduction representation, a graphical method that presents proofs as linear sequences of formulas [5]. But, Gentzen original presentation [9] conceived of them as finite trees: the root of the tree is the formula to be proved, the leaves of the tree are the assumptions and the other formulas are obtained by the application of an inference rule from the formulas standing immediately above it.

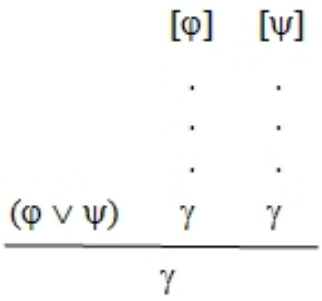
The Gentzen representation of proofs allow us to display the logical support structure of arguments and to identify the subarguments of which complex arguments are built, i.e. what Freeman calls "the macrostructure of arguments". In this way, by identifying the argumentative strategies that natural deduction rules intend to codify, and by portraying derivations as special instances of Gentzen style diagrams, a reasoned and historically situated transition from arguments in natural language to mathematical logic derivations can be made possible.

In order to illustrate this proposal, let us look at the following version of an argument presented by Plato in his Apology [7]. Death is one of two things: either death is a state of nothingness and utter unconsciousness, or, as men say, there is a change and migration of the soul from this world to another. Now if you suppose that there is no consciousness, but a sleep like the sleep of him who is undisturbed even by the sight of dreams, death is good. But if death is the journey to another place, and there, as men say, all the dead are, then death is good. Therefore, in any case, death is good. This argument exemplifies the argumentative strategy of reasoning by cases and its standard diagram is the following:



The diagram – where the subarguments that support the conclusion are enclosed within a box, and assumptions are enclosed between brackets- makes evident that this argument is a case of reasoning from assumptions, and, in particular, an example of reasoning by cases. This strategy is represented by the rule of disjunction elimination, which in Gentzen's natural

deduction system adopts the following form:



The diagram that represents Plato's argument as a tree whose conclusion is supported by linked premises clearly suggests the inferential strategy that can be applied to the FOL translation of the argument in order to derive its conclusion and provides an intuitive understanding of the inference rules involved in the derivation. In this way the close relation between the macrostructure of natural language arguments and derivations in First Order Logic is made evident.

5 Conclusions

In this work we have presented a proposal that aims at the integration of natural deduction and philosophical argumentation in an introductory course of Logic for Philosophy students. We drew from, and conceptualized, the pedagogic experience obtained teaching the mandatory undergraduate Logic course offered by the Philosophy Department at the University of Buenos Aires.

On the one hand, we advised for the integration of the informal presentation of some argumentative strategies that are commonly found in philosophical argument with the FOL rules that codify those strategies in natural deduction systems.

On the other hand, we proposed that those courses incorporate the standard technique of argument diagramming that allow for the identifying and representation of the inferential relationships between sentences that are part of arguments in natural language. These techniques offer students an opportunity to grasp the intuitive sense of the building of proofs in FOL, and discover its relationship with the inferential structure of arguments in natural language.

Based on these premises we aim at building a closer integration between the sections reserved for informal logic and those set apart for mathematical logic in introductory courses and textbooks of logic for the Humanities. It is to be hoped that this integration will bring to the fore the relevance of the mathematical logic content included in those courses for the study of natural language arguments, and especially for the study of philosophical argument.

References

1 H. Beebee, Introductory Formal Logic: Why do we do it?. *Discourse: Learning and Teaching in Philosophical and Religious Studies*. 3 (1): 53-62, 2003.

2 Cheng, P. W., Holyoak, K. J., Nisbett, R. E., and L. M. Oliver. Pragmatic versus syntactic approaches to training deductive reasoning. *Cognitive Psychology*. 18(3): 293–328,1986.

3 J. B. Freeman. *Dialectics and the Macrostructure of Argument: A Theory of Argument Structure*. Foris, Berlin, 1991.

4 J. B. Freeman. *Argument Structure. Representation and Theory*. Springer, Dordrecht, 2011.
5 A. P. Hazen and F. J. Pelletier. Gentzen and Jaśkowski Natural Deduction: Fundamentally Similar but Importantly Different. *Studia Logica*. 102: 1–40, 2014.
6 R. H. Johnson and J. A. Blair. Informal Logic and the Reconfiguration of Logic. In D. Gabbay, R. H. Johnson, J.-J Ohlbach, and J. Woods (eds.), *Handbook of the Logic of Argument and Inference: The Turn toward the Practical*. Elsevier, Amsterdam, pages 339-396, 2002.
7 Plato. *Euthyphro, Apology of Socrates, Crito*. Clarendon Press, Oxford,1924.
8 G. Ryle. *The Concept of Mind*. Hutchinson, London, 1949.
9 M. E. Szabo (ed.). *The Collected Papers of Gerhard Gentzen. Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1969.

Multiple choice parameterized exercises in Logic*

Jacinta Poças, João Pedro Cruz, Luís Descalço, and Paula Carvalho

Department of Mathematics, Aveiro University
3810-193 Aveiro, Portugal
{jacinta.pocas, pedrocruz, luisd, paula.carvalho}@ua.pt

Abstract

The first contact of students with logic is not straightforward since they are not familiar with its notation and with logical thinking. For this discipline to become more attractive and in order to avoid this intimidating first contact, interactive didactic tools can be useful. In the University of Aveiro there exists a group of teachers developing an interactive platform that provides students some guidance in their independent study in several courses. We propose and describe the use of this platform as a complement for teaching the logic course of Mathematics Department in this University.

1998 ACM Subject Classification F.4 Mathematical logic and formal languages and K.3.1 Computer Uses in Education

Keywords and phrases Logic, interactive platform, multiple choice parameterized exercises

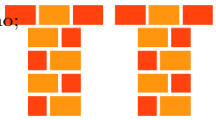
1 Introduction

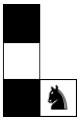
Logic is taught in several departments in the University of Aveiro. Usually, for students, it is not easy to start learning logic because this domain of mathematics is completely unfamiliar in their knowledge background in Portugal. Therefore, we need didactic material, but there exists few exercises in literature for our mathematical logic syllabus and almost all of it is in English language. Furthermore, the theoretical framework of logic that we can find in literature, is introduced with different notations, what makes the study difficult. Indeed, logic requires a special vocabulary with its respective notation and a way of thinking that students have never contacted before. Hence, we have looked for attractive tools to teach logic. We want to propose exercises in Portuguese language that will be linked with a theoretical textbook containing uniform notation and all of this in an interactive fashionable form.

In the Mathematics Department of the University of Aveiro, in Portugal, there exists a group of teachers who are developing two systems, the first MEGUA (Mathematics Exercises Generator, University of Aveiro) [1] allows teachers to construct multiple choice parameterized exercises in a shared cloud environment and the second system SIACUA (Sistema Interativo de Aprendizagem por Computador, Universidade de Aveiro) [5] is used by students to learn. From the authoring tool, teachers send their exercises to the students learning system.

In the first section, we present our motivation to construct interactive exercises using this platform. Then we explain all the software that we need and how we use it. In third section, we give two examples of multiple choice exercises, constructed in MEGUA, with detailed answer used for logic. Next section shows some advantages in using these tools for teaching logic, from the point of view of students and teachers. Afterwards, we describe some

* This work was partially supported by CIDMA ("Center for Research & Development in Mathematics and Applications") and FCT ("FCT–Fundação para a Ciência e a Tecnologia") through project UID/MAT/04106 /2013. The first author was supported by the project FCOMP-01-0124-FEDER-028923 (Nasoni).





4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 177–184

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

difficulties that we have met to implement multiple choice parameterized exercises about logic. Finally, we conclude by suggesting improvements to the platform for better teaching.

2 Motivation

In the University of Aveiro in Portugal, there exists an interactive platform already used in Calculus subjects, that propose multiple choice questions with a detailed answer and truth/false exercises (without resolution) for students. They are also developing exercises for College and Secondary School with the support of school teachers.

Before implementing this kind of project for logic subject, we made a short survey to check if this interactive platform is really efficient for students. The survey was distributed to a population of about three hundred students and contains thirteen questions, one of them is an open question in order to give them the opportunity to describe the advantages and disadvantage of this platform. A study throughout this survey reveals that students often use this platform and consider that multiple choice exercises with detailed answer are more useful rather than truth/false exercises without any detailed explanation of the correct solution. Furthermore, all of them recognize that it is crucial to have an interactive platform like this allowing independent study instead of only PDF materials for download. Moreover, most of them want to use this platform in their partial assessment and consider that it promotes their learning. The presentation of a detailed answer of exercises is the most important element, in the point of view of students, since it clarifies doubts and allows them to independently answer other similar exercises. The platform gives information about their evolution in the different topics. These topics are shown as a tree of concepts we have previously defined. For students, it is important to know immediately if they answered correctly the multiple choice question, and in the future, they would like that the platform suggests exercises according to their knowledge and mistakes. Finally, almost all students would like that other disciplines use this platform.

Taking into account the results of this survey and our concern to diversify tools for teaching logic, we are creating, in the platform, parameterized exercises for logic in the context of the syllabus teaching in Mathematics Department of University of Aveiro, that is, History of logic, in particular Aristotle syllogism and paradox, Classical Propositional Logic and First-Order Logic. Moreover, we want to propose a new didactical component, different from the traditional one, in the sense that we are making a lot of exercises available in a variety of types, where we suggest a detailed answer.

3 Platform description

The platform is composed of: authoring system (MEGUA) and student learning system (SIACUA).

The first allows an author to write and run exercises in the cloud. We are using Sage Notebook [3] which allows anyone to create, collaborate on and publish interactive worksheets. In each worksheet, we can write code using Python language, the Sage Mathematics library and other software included in Sage [4] which is a free open-source mathematics software. Coding with this type of environment is very easy, it is not necessary to have a training in programming. It is easy to find in the internet several short examples to program for HTML, LaTeX and Python, what is enough for writing exercises. The library that allows exercise authoring, MEGUA [1], is an open-source software [2]. MEGUA borrows from Sagemath the Sage Notebook tool for editing online and sharing work with others. Exercises are created

and edited as Sagemath notebook worksheets. Sagemath also gives an extensive library of mathematical routines that help calculations in each exercise.

MEGUA package was created aiming the authoring of parameterized problems which is one of our objectives. Indeed, we can take a classical exercise and parameterize it, so that we obtain several exercises in few minutes which assess the same notion in logic. Each exercise is characterized by a summary, a problem, a multiple choice part, and a detailed answer for the exercise. Each author (or project) has his own database of parameterized exercises and can share each exercise with other authors for collaboration. We can choose a set of questions from a database and print them to create a small textual study book or a paper exam, with or without resolution.

The other principal software, SIACUA [5], is an experimental project which implements a Bayesian user model. Each user is associated to a Bayesian network that includes initial study concepts map which is updated after each evidence of knowledge (or no knowledge) of the student. Exercises which are created in MEGUA are generated randomly by the author and few of them are selected and sent to this website [6]. This software allows student to solve interactively all exercises created in Sage Notebook by teachers and gives them some feedback about their progress. Indeed, they can know how much times and when students were connected, and evidently what they answered. This software is crucial in our viewpoint, because Sage Notebook is interesting in creating parameterized exercises and obtaining many questions in few time, but it is more helpful for teachers the possibility to make available, throughout internet, these exercises for student learning.

Both systems MEGUA and SIACUA are being developed at the University of Aveiro. The computer system is described in detail in [7].

4 Examples of parameterized exercises in logic

Syllogism is a part of our program in logic and we give here an example of how to create a multiple choice parameterized exercise about this subject. We want to check if students can deduct a valid conclusion given two premises using Venn diagram. In figure 1, we show the statement of the exercise, two premises and one question, and we give four choices ("escolha" in Portuguese) where the first is always the correct and the remaining ones are false before the system shuffles answers for students. In system MEGUA, the first choice is the correct one, then the choices are randomly ordered in SIACUA and four of them, always including the correct one, are presented to the student. In figure 2, we show a detailed answer of the exercise by using a Venn diagram and the corresponding valid conclusion. It is easily to parameterize this type of exercise substituting the words "seagulls", "cows" and "fly" by others. The difficulty level and the objectives of the question remain the same.

In some exercises we parameterize connectives of logic, formulas, inference rules, etc. Some parameters are numbers and can be chosen from a predetermined set of numbers, some parameters can be nones, expressions, functions and even a set of symbols like $\vee$, $\rightarrow$, $\wedge$, the connectives symbols used in logic.

For some kind of exercises this parametrization can be more difficult. We present an example where the aim is to evaluate if students can translate sentences in natural language to Mathematical formulae, more precisely in formal logic, and find the solution using Classical Propositional Logic. In figure 3, we show the statement of the exercise and the four choices. We present, in figure 4, a detailed answer for the exercise giving a translation of the text using propositional variables and the respective truth table from which we have the solution of problem.

Consider the following statements:

All seagulls can fly.
Some cows cannot fly.

Which is a possible conclusion to obtain a valid syllogism?

Escolha:
Some cows are not seagulls.

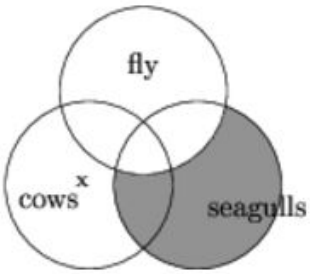
Escolha:
No cows are seagulls.

Escolha:
All seagulls are cows.

Escolha:
Some seagulls are cows.

Figure 1 Syllogism

We use Venn diagram:



We have a X in a part of circle of cows which is not in seagulls.
Thus we conclude that there are some cows that are not seagulls.

Figure 2 Detailed resolution of Syllogism

5 Advantages in using the platform

This platform allows an independent study using only computers and internet. Before answering an exercise in logic, students can look at theoretical contents in PDF format. Thus, they can at anytime and anywhere study their course without printed paper. Furthermore, this platform may be a principal tool for working students. Indeed, they do not spend much time in the university and consequently, sometimes, they do not have printed content that teachers transmit in classes. Since, we may access this platform at distance, it may be used as a tool for teaching logic in any country of Portuguese language.

Moreover, the presentation of a detailed solution is very important for students. They can test their own knowledge by answering correctly the proposed question and, in that very moment, see a proposed correct answer made by the teacher. This fact is new, because there exists on the internet and books lots of multiple choice or true/false exercises in logic but rarely they offer a detailed solution. The aim to solve exercises is to check if students are understanding all mechanisms of logic being taught and this platform is a frame to evaluate this notion.

One of the advantages for teachers is that, although the time spent to construct interactive exercise is long, in the future they save time because they are creating a database that contains theoretical lessons in PDF format and many exercises. Each teacher can fill this database with new parameterized exercises and can share it. In addition, this platform allows

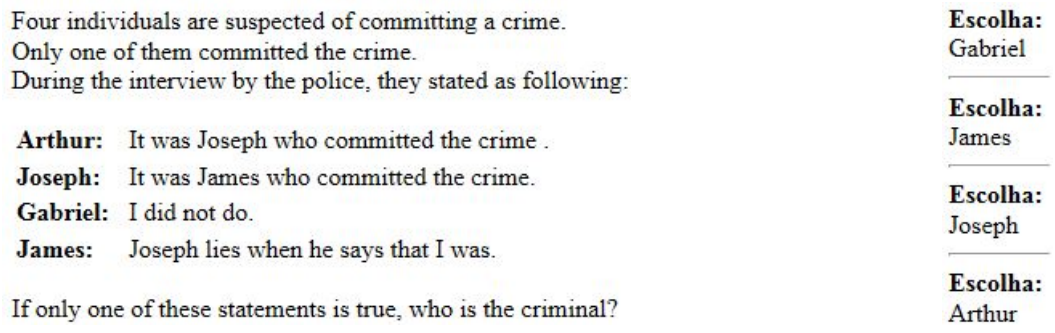


Figure 3 Translation and Resolution

to keep in the cloud all exercises that are created. In other words, exercises are not stored in one computer for only one teacher, they are reachable for all teachers they want to invite to collaborate on their construction.

6 Difficulties of implementation

Generally, a "classical" question is formulated as a question where student must prove or solve something. The main difficulty that we are confronted with is how to reformulate a question in multiple choice question formulation with only one true choice and the three others false. Sometimes the construction of such an exercise is very complex. In some exercises, we ask, which one is the wrong answer instead of asking for the correct answer, as usual. For instance, which of the following expressions is not a formula in Classical Propositional Logic? Or which of the following inference rules is not a derived inference rule of the system? In these cases, student has two possibilities: he can find a proof of the correct answers and obviously the remaining one is false, or he can obtain directly the answer that is not correct.

The main difficulty, that we find, is how we can parameterize some exercises with different detailed solutions, depending on the parameters. We think that it is for this reason that we can find lots of interactive exercises in internet but without detailed solution. In spite of this difficulty, we keep present the detailed solution, otherwise the exercise is not complete in our point of view, and do not offer a complete independent study solution.

7 Conclusions and Future Work

We present a platform to learn logic in an interactive way. It offers a place for students (and teachers) where they may find all the information, theoretical-lessons, exercises and explanations about logic as taught in the University of Aveiro. This system permits self assessment for students with feedback about students knowledge progress.

On the future, we expect to diversify the type of exercise suggested, other than the current multiple choice type, for example, exercises where it may be possible to have several correct answers and not only one. Indeed, with this kind of exercise, students must analyze each answer independently. Therefore the difficulty of the same exercise is higher and it reduces the probability to find the correct answer by elimination of the others.

Moreover, we want to improve platform suggestion of exercises according to the evolution of knowledge of the student and its difficulties, towards an individual and adapted learning. Furthermore, teachers may have better feedback of the student's knowledge and may

Considering the following propositional variables:

a	Arthur is the criminal
j	Joseph is the criminal
g	Gabriel is the criminal
t	James is the criminal

We have the following representation:

It was Joseph who committed the crime	j
It was James who committed the crime	t
I did not do	$\neg g$
Joseph lies when he says that I was	$\neg t$

Since only one statement is true, we have the following cases to analyse:

	j	t	$\neg g$	$\neg t$
case 1	1	0	0	0
case 2	0	1	0	0
case 3	0	0	1	0
case 4	0	0	0	1

Cases 1 and 3 can not occur because, with the same valuation,
a formula cannot be tautology and contradiction.
In case 2, variable t has valuation 1, thus James is criminal.
Moreover, variable $\neg g$ has valuation 0, so variable g has valuation 1, thus Gabriel is criminal.
But there is one and only one of them criminal, therefore we also eliminate this case.
In case 4 we have:
Variable j has valuation 0, thus Joseph is not criminal.
Variable t has valuation 0, thus James is not criminal.
Variable $\neg g$ has valuation 0, so variable g has valuation 1, thus Gabriel is the criminal.
This is also compatible with the last statement.

Evaluating all cases, if only one statement is true, the criminal must be Gabriel.

Figure 4 Detailed resolution of Translation and Resolution

recommend other material to study according to their skills.

Due to the simplicity of programming, needed for creating exercises in the platform, teachers may propose students to create one exercise of logic using directly the system. This idea presents two advantages. One is that we can augment the database of exercises with the collaboration of students. The other is to propose high level concept themes for students. Indeed, students must think to construct the exercise in several ways. They write a question, in which all of the information must be complete, then they propose several choices as possible answers, in which only one is correct but the others must be according to the possible error that students usually do so that the correct answer is not trivial. And finally, they propose a detailed answer for the problem. We believe that if a student acquires all these notions, that are necessary for the construction of an exercise, we have the proof that he knows very well the topic in study. This is the same as, if a student explains the notions to another student and the second achieves correctly the problem and the solution, then the first student really knows this topic of logic.

References

1 Pedro Cruz, Paula Oliveira and Dina Seabra. Exercise templates with Sage. Tbilisi Mathematical Journal, Vol 5(2), pp. 37-44, 2012. (http://cms.ua.pt/megua)

2 `http://code.google.com/p/megua/`

3 William A. Stein et al., Sage Mathematics NOTEBOOK (Version 5.2), The Sage Development Team, 2013, nb.sagemath.org

4 William A. Stein et al., Sage Mathematics software (Version 5.2), The Sage Development Team, 2013, http://www.sagemath.org.

5 Eva Millán, Luís Descalço, Gladys Castillo, Paula Oliveira and Sandra Diogo. Using Bayesian networks to improve knowledge assessment. Computers and Education, Vol 60(1), pp. 436-447, 2013.

6 `http://siacua.web.ua.pt`

7 Paula Carvalho, João Pedro Cruz, Luís Descalço, Paula Oliveira and Dina Seabra. Using Bayesian networks and parameterized questions for independent study. To appear in the proceedings of EDULEARN1S, Barcelone, July 2015.

Transitioning to Proof

Diane Resek[1] and Dan Fendel[2]

1 **Department of Mathematics, San Francisco State University**
1600 Holloway Ave, San Francisco, USA
`resek@sfsu.edu`

2 **Department of Mathematics, San Francisco State University**
1600 Holloway Ave, San Francisco, USA
`dfendel@sfsu.edu`

Abstract

This paper describes some strategies used in a 'transition' course. Such courses help undergraduate mathematics majors move from learning procedures to learning to function as critical mathematicians in order to understand and work with abstract concepts. One of the co-authors of this paper was a student of Leon Henkin. His influence on her helped shape the strategies used in the course, and is described at the end of the paper.

Keywords and phrases logic, proof, transition course, Henkin.

1 Introduction

In the United States we often hear students complain: "I understand the math, but I just can't write proofs." A few probing questions usually reveal that the student really does not understand the mathematics involved. She or he may be able to parrot back the key definitions, but doesn't know what inferences can be drawn or what relationship needs to be proved. What is really happening here? First, students often believe that they understand mathematical concepts after they have read about them in the text or attended a lecture on the material. They believe that, if they find they can't do the homework problems, whether proofs or other non-mechanical exercises, then there must be something wrong with the book or the lecture. Students need to learn what people who have become mathematicians instinctively know and probably can't remember being taught. That is, in order to understand a new idea, they must play with it, examine examples, look for counterexamples, ask themselves questions about it, and more. Currently, many universities in the United States offer courses to students before they take proof oriented courses such as abstract algebra or real analysis. Such courses are designed to help students to "transition" between courses that focus on procedures, such as calculus as taught in the U.S., and more conceptually oriented courses. The courses are aptly named "transition courses." The authors of this paper wrote a text [1] for such a course, 1990). The text and the course sought to help students learn how to cope with open-ended, unstructured questions; how to formulate conjectures; how to evaluate the reasonableness of a statement; how to make plausibility arguments; how to make constructive use of examples; etc. These skills are difficult to learn, and some time needs to be spent acquiring them when the student and the instructor do not have other important agendas. One of the authors of this paper and the text, Diane Resek, was a PhD student of Leon Henkin, and worked closely with him on mathematics education projects as well as mathematical logic. This work shaped her thinking about logic and the teaching of logic. She has written about some of her intellectual inheritance from Henkin in the chapter "Lessons from Leon" in the book, The Life and Work of Leon Henkin [2]. In this paper we will discuss some of the techniques used in the transition course and its text and

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 185–190

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

then explain how the influence of Leon Henkin helped shape the ideas behind them. The techniques are:

- emphasizing exploration;
- including exercises entitled "Get your hands dirty";
- introducing logic connectors via the idea of counter-examples and not truth tables; and
- asking students to judge the correctness of proofs.

We will discuss each of these elements in turn.

2 Exploration

One way to motivate students' interest in proof writing is to raise questions as to the truth of a statement. This method is especially motivating if students are engaged in arguments with their peers about the correctness of the statement. The course begins with the presentation of the rules for a simple game. Students play the game in groups and are asked to raise questions about the game. They come up with a number of questions, but settle on trying to find a winning strategy. The activity ends with students trying to prove that a given strategy works. For homework they work on strategies for other games.

As the course progresses, they meet new subject matter such as sets, integers, quantifiers, functions and sequences. Each time a new subject is introduced, students work at exploring that subject before they begin writing proofs in that domain. For example, when quantifiers and connectives are introduced, students are asked:

Consider "$(\exists x \in S)(p(x)$and $q(x))$" . How does that statement compare to "$(\exists x \in S)(p(x))$and$(\exists x \in S)(q(x)))$"? Can the original statement be true and the combined statement false? Or vice versa? Try some examples with specific open sentences and look for general principles.

Students' feelings of frustration are acknowledged in the text. They are not used to being directed away from collecting a set of answers and toward the process of thinking about mathematics. They are not used to being left with many open problems in a mathematics course. We let them know that, for this course and for their future work with mathematics, they need to get used to the fact that they will sometimes be coming away from their work with more questions than answers. We remind them that they can always return to their questions on another day, but we emphasize that coming away with good questions is a positive accomplishment.

The object of exploration in our text and course is not simply to motivate students to prove things, but rather, to build their intuition about a given topic. We seek to instill in students a habit of exploring on their own each time they come up against a new mathematical concept.

3 Get Your Hands Dirty

Throughout the text, students meet exercises labeled: "Get Your Hands Dirty" (GYHD). They are told "This means you must stop being an armchair mathematician and get to work." They are also told:

Mathematics is not a spectator sport. Even when you are just reading mathematics, you need to be an active participant, or you won't understand. Advanced mathematics texts expect you to explore, though they don't explicitly tell you to do this. For example, when a mathematics book gives you a new definition, you must "play with it." That's the best way

you can come to understand a new idea. Advanced textbooks will expect you to have an understanding of the new idea as they proceed.

We don't expect students to know how to play with definitions when they come to our transition course. We try to teach them that. For instance, when we introduce the definition of a power set, we give them examples. Then we give them a GYHD exercise in which they are asked to find the power sets of some small sets, and later ask them (in another GYHD) whether something can be simultaneously both an element of and a subset of some other set. Similarly, when quantifiers are introduced, students are given the definition of what it means to say that set A is a subset of set B: $(\forall x \in A)(x \in B)$. They are then asked, in a GYHD, to come up with an expression using quantifiers for what it means to say that A is not a subset of B.

GYHDs are different from exploration exercises in that GYHDs are generally very short activities through which students come to understand a theorem or a definition. The object of GYHDs is to train students to "play" with new definitions and theorems. For instance, they need to learn to ask themselves why certain qualifications such as "$x \neq 0$" are given, or why the definition is not more general. An "exploration" is a longer exercise in which students are expected to make conjectures and then try to prove or disprove them.

Since we used GYHDs when introducing new definitions, we expected that students would learn how to make sense of new definitions by the end of the course. So, on the final examination, we gave students the following formal definition: A set of real numbers S is called connected if $\forall x \forall y \forall z(x \in S$ and $y \in S$ and $x \leq z \leq y \rightarrow z \in S)$

We expected the definition to be totally new for them, since the concept doesn't come up in earlier courses. We intentionally did not give them any context for the definition, because we wanted them to decipher it on their own. We then asked them to come up with examples of sets of real numbers that fit the definition and sets that did not. The goal was for them to learn how to build an intuitive understanding based on a formal definition. Finally, we asked them to prove a specific statement that followed easily from the definition.

4 Logic Without Truth Tables

Probably everyone teaching the truth table of the conditional connective has come up against resistance from students to the notion that A→B is considered true anytime A is false. This idea simply does not make sense to many students. We therefore decided to ignore truth tables in the main part of the book, and work instead with predicate logic. That is, instead of working with truth tables, we emphasized the role of counterexamples in assessing the truth of conditional sentences. So, for our purposes, the key idea is that a conditional sentence is true unless there is a counterexample. Proofs of conditional statements then become demonstrations that there is no counterexample.

Of course, as one would hope, this definition of the truth of conditionals is equivalent to the one given by truth tables. But in this approach, we did not meet the usual resistance to the definition or the use of the definition.

Thus, students see that to prove that a conditional statement is true, one can assume that we have some unspecified object that makes the hypothesis true, and we then need to show that the conclusion must be true. If we can do this (knowing nothing about the object except that it makes the hypothesis true), then we have shown that there can be no counterexample.

They can also use and justify proof by contrapositive in the same way: start with an unspecified object that makes the conclusion false and demonstrate that the object makes

the hypothesis false, and thus, again, the object is not a counterexample. Similarly, proof by contradiction is justified by assuming one has a counterexample and showing that this leads to an impossible situation. Thus, there can be no counterexample. Note: We did put some more formal logic in an appendix to make the book useful to a wider audience of instructors.

5 Proof Evaluation

We found that a particularly useful exercise in helping students develop their proof writing abilities was to give them "potential proofs" and ask them to evaluate whether the given piece of writing was an actual proof. This meant that they needed first to decide whether the "theorem" (i.e., what was supposedly being proved) was in fact true, and, if so, to then determine whether the given "proof" was in fact valid. If it was not valid, they were to identify the flaw.

Here's an early example: ["$P(A)$" is our notation for the power set of A.]

"Theorem": For any sets A and B, if $P(A) \cap P(B) \neq \emptyset$ then $A \cap B \neq \emptyset$.

"Proof" Assume $P(A) \cap P(B) \neq \emptyset$. Our goal is to show $A \cap B \neq \emptyset$. Since $P(A) \cap P(B)$ is non-empty, there is something in this intersection; call it X. So, $X \subseteq A$ and $X \subseteq B$. Now let y be any element of X. So $y \in A \cap B$, and so $A \cap B \neq \emptyset$. This concludes the proof.

This type of exercise helped students to understand some subtleties of proof and alerted them to some pitfalls that they might come across. We wanted students to distinguish between "omissions of detail" and real mistakes. (For instance, in this "proof," we could have stated the definition of power set in going from saying that X is in $P(A) \cap P(B)$ to saying $X \subseteq A$ and $X \subseteq B$. That is simply an "omission of detail" but the step is a valid one) .

Often it was helpful to ask students to look for a counterexample, and then find the first statement in the "proof" that would actually be false for that counterexample.

6 The Influence of Leon Henkin: Personal Reflections from Diane Resek

Each of the techniques discussed here—exploration, "Get Your Hands Dirty," use of predicate logic rather than propositional logic, and "proof evaluation"—was designed to make students' work with mathematical ideas more intuitive, more natural, and more like what mathematicians actually do.

Leon always sought in his own teaching to make ideas intuitive. As a thesis advisor he pushed me to try new approaches and to look at concrete examples whenever possible. In my dissertation work on infinite dimensional cylindric algebras, his approach led me to think of each dimension as a separate piece of paper. I actually wrote out some examples on a large number of pages and this helped me come to a major conjecture.

Those pieces of paper worked for me. This experience and others—seeing the power of getting one's hands dirty while working with concrete examples—led me to use the strategy while teaching students.

Another influence of Henkin on me, and thus on the transition course, came from his drive to make mathematical knowledge available to as wide an audience as possible. He said that although there might be a level of mathematical knowledge that average people could not attain, he believed that this level was beyond what is needed to complete an undergraduate mathematics major. One goal of the transition course was to make the higher-level courses in the mathematics major accessible to more students.

References

1 Fendel, D., & Resek, D. *Foundations of Higher Mathematics: Exploration and Proof*. Pearson Education, Ltd., New York, USA, 1990
2 Manzano, M., Sain, I., & Alonso, E. *The Life and Work of Leon Henkin: Essays on His Contributions*. Birkhauser, Switzerland, 2014

Teaching Logics through Their Philosophical Commitments: "Logical Worldviews"

Creighton Rosental

Associate Professor of Philosophy
Mercer University
rosental_c@mercer.edu

Abstract

I have developed a pedagogy and textbook for teaching logic centered on what I call "logical worldviews". A logical worldview examines the close connection between philosophical commitments and the logical principles and method for a particular historical logical system. The class examines multiple historical logical worldviews to show how philosophical positions and logical systems have been closely intertwined, and how changes in philosophical positions have over time yielded corresponding changes in the development of logic. Such an approach has great benefits for teaching logic to undergraduates.

Keywords and phrases history of logic, logic education

1 A "New" Approach to Teaching Logic to Undergraduates: Logical Worldviews

Since 2006, I have developed an approach to teaching logic to undergraduate philosophy majors that bases logic instruction in the philosophical background underpinning historic logical systems. Starting in 2013, I expanded this pedagogical approach to a general education introductory logic class, which is open to students of any major and at any stage of their college career. This curricular development has culminated in a textbook designed for an introductory logic course that can be used by teachers with minimal or no experience in the history of logic.

In the textbook I present a re-contextualization of classical formal logic as just one of many approaches to logic, each having its own philosophical commitments and unique value. I suggest that there is more to logic than is construed by only considering its contemporary formal aspects; after all, formalism in mathematics and logic is a relatively recent innovation, adopted in part because of other, non-mathematical considerations and commitments.

I hold that the history of logical thought includes a series of logics, or logical systems, each of which has non-formal philosophical commitments. I further hold that these commitments are not separate and independent add-ons to the logical system, but that the logical method and fundamental principles of the system are shaped and determined by these commitments. Though a historical logic can be considered formally and in the abstract, such a perspective yields an inaccurate account of what the logical system was intended to do by its author, of the true power of its methods of reasoning, and of the precise meaning of the logical principles employed. When the logical system is considered in conjunction with the intertwined philosophical commitments, I characterize this broader conceptualization as a "logical worldview."

2 Using Philosophy to Better Understand a Logical System

It will be useful at this point to consider a particular example, and Aristotle is perhaps the best place to start: after all, as the first to fully develop a logical worldview, many later

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 191–198

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

logics were (as I present in the textbook) a response to rejecting one or more Aristotelian philosophical commitments. What follows is a very brief (and incomplete) characterization of some of the philosophical commitments in Aristotle's logic.[1]

- Theory of truth: Correspondence. Propositions are true if and only if they correspond to how the world actually is; false otherwise. Somewhat anachronistically, we might refer to propositions as being true if and only if they correspond to facts about the world. This view also includes bivalence – propositions are either true or false.[2]
- Access to truth: Direct Realism. Human beings can directly access the world, and the properties and entities they perceive are (generally, in principle) the same as the actual properties and entities that exist in the world.[3]
- Theory of mind: Simple Apprehension and Judgment. The human mind has a capacity to discern the true properties of things. Further, we are capable of abstracting from particular properties to universal forms shared by more than one individual entity, and recombining such forms in the imagination and in memory. Finally, we can combine previously unseen combinations of forms and judge whether or not such propositions correspond to reality.[4]
- Metaphysics: Substances, Forms and a Hierarchical Ontology. In Aristotle's account, the world is constituted by individual entities, or primary substances, each of which has a number of properties, or forms. Substances can be grouped together into secondary substances, or species, for which some forms are essential, others not. Species can be grouped together into a higher-order grouping, or genus, which have their own essential forms. Collectively the interconnection of species and genus through essential forms constitutes a hierarchical ontology, which is rationally discernible.[5]

In my textbook I explain how these (and other) features of Aristotle's philosophical worldview determine the specifics of his formal logic. I'll sketch this dependence briefly here.

First, individual substances and their forms are the fundamental perceivable elements of reality. Collectively, these are referred to as "terms", which are the atoms of Aristotle's logical system.

Second, propositions are assertions about the actual facts of the world. Any fact will fundamentally involve two components: an identified subject and one of its forms. Since propositions correspond to these fundamental facts, every proposition will have only two terms: a subject term (what the proposition is about), and a predicate term (what is being said of that subject).

Third, through experience, induction, abstraction, and natural talent, humans are capable of discerning some of the essential qualities of substances. This allows us to generalize beyond

[1] Here and throughout this paper, I present the philosophical views in an introductory and casual manner, to match the form in which they are presented in the textbook. Though some of these Aristotelian interpretations are contested by scholars of Aristotle, a close and well-defended history of Aristotle's logical and philosophical positions would not be to the point. What matters for the sake of this pedagogical approach is that philosophically and historically inexperienced students see that Aristotle's logical system is amenable to being better understood by framing it in terms of philosophical positions that Aristotle (plausibly) held.

[2] See Aristotle *Categories* chapter 14.

[3] See Aristotle *de Anima*, particularly Book II chapter 12 through Book III chapter 4.

[4] See Aristotle *On Interpretation* chapter 1 and Posterior Analytics Book II chapter 19.

[5] See Aristotle *Categories*; also *Metaphysics.*

particular substances to be able to quantify the scope of the assertion (all, some, one) and whether the subject and predicate are linked (affirmation) or separated (denial).

Fourth, through our capacity to discern the ontological structure of reality, we are able (in principle) to choose any two propositions and connect them by means of something in the hierarchy that links them (a middle term). The fundamental form of reasoning from one proposition (with two terms) and another (sharing one of those terms) is through a bridge proposition, sharing a "middle" term with our premise and conclusion. This is the minimum size of an argument productive of new knowledge, and thus the syllogistic form must have three propositions sharing exactly three distinct terms.

This is a rough and quick sketch of how Aristotle's philosophical commitments determine the form of his logical method, which culminates in the syllogism. This perspective on Aristotle's logical worldview also helps to explain many other curiosities about Aristotle's logic that have perplexed many trying to understand his logic. I can't get into them in detail here, but here are a few positions in Aristotle's logic that are explained by considering his philosophical commitments.

- Reduction to the first figure. Rather than be satisfied with identifying valid syllogistic moods, he takes great pains to reduce them to four figure one moods. He does so because he believes that these four moods are the fundamental natural and self-evident reasoning patterns for human beings.[6]
- The missing fourth figure and two moods. In inventorying valid moods, Aristotle neglects to identify two valid moods in figure one and figure four altogether. These moods do not fit the "natural" reasoning patterns determined by Aristotle's view and are thus neglected as inappropriate forms of human reason.[7]
- Existential import. Many have found that Aristotle's inference that "All S are P" implies "Some S is P" to be logically invalid. However, given his philosophical commitments, no universal claim could ever be made without first establishing the truth of the particular, so the truth of the universal will always imply the particular.

Though this exposition of this logical worldview is seriously abridged, it should demonstrate that the form, method and logical principles of Aristotle's logic are heavily dependent on Aristotle's philosophical commitments. Those who wish to teach Aristotle's logic to students can use the logical worldview in order to have Aristotle's logic make much more sense to students, and show how his logic is closely connected to many of Aristotle's other philosophical positions.

3 Using Philosophical Commitments to Explain Transitions from One Logical System to Another

I'll briefly discuss how later logical systems can be understood as having rejected or modified some part or other of Aristotle's logical worldview. I'm not suggesting that these later logics were the result of simply abandoning some philosophical commitment or other of Aristotle's; rather, by rejecting one or more commitments, new opportunities for logical method are opened up, which in combination with some innovations led to new ways to do and conceive logic. I'll briefly consider here two logical worldviews post-Aristotle.

[6] See Aristotle *Prior Analytics* Book II chapters 1-15.
[7] For a good discussion of this issue see Lynn E. Rose, "Aristotle's Syllogistic and the Fourth Figure," *Mind* vol. 74, no. 295 (July 1965): 382-389.

Bacon's rejection of Aristotle's theory of mind: a motivation for natural history and modern natural sciences. In Bacon's *New Organon*, he takes Aristotelian philosophers to task for relying too heavily on our natural capacity to understand the forms present in reality and argues that nature is far too subtle for our unaided sensory and cognitive capacities. He argues that we must develop new instruments to enhance our senses (microscopes, thermometers, etc.) and to enhance our cognitive discernment of what we are seeing by means of experiments.[8] Since our capacity to naturally discern the forms in reality is flawed, we must develop a new logical technique, which is basically induction from a very large number of particular instances towards generalizations of increasing universality and confidence.[9] Therefore, the syllogism, which relies on gathering information about forms and then deducing new truths is for Bacon the wrong logical method. Bacon does not reject Aristotle's other philosophical commitments mentioned above, and to varying degrees and with some modification, the natural sciences still endorse a correspondence theory of truth, a capacity to discern forms of substances, and a hierarchical ontology.

Boole's rejection of Aristotle's Direct Realism and metaphysics: an effort to mathematize logic. To a great extent Newton's mathematical approach to physics follows similar rejections as Boole, but for the sake of what will soon become clear, Boole's contribution to logic is easier to present to students. Starting with the Early Modern philosophers in the 17th century and culminating in the modern approach to science in the 19th, the rejection of Direct Realism became fairly common. Philosophers were no longer inclined to believe that we had direct access to the substances and forms through our senses exactly as they are in reality. Because we lacked the capacity to see the world as it actually was, Boole changed the focus of logic from terms to classes in his seminal work The Laws of Thought. Instead of the logical atom being a term, which required discerning the true forms of individuals, he made the fundamental element the class, a collection of individuals gathered in some (perhaps unknowable, perhaps arbitrary) way. One could identify a class without having to know exactly what was required for membership. His particular brilliance was to show that the logical method of Aristotle could be encompassed within an algebra for classes. By shifting from term to class, Boole needed a new logical method, which he found in algebra. Though Boole mathematized logic, one should note that he did not abandon Aristotle's philosophical commitments completely: he still largely conceived of reality as being metaphysically ordered in a hierarchy, and also that truth was established by correspondence. Further, he did believe that he had discovered the "laws of thought": the actual laws by which human minds reasoned, maintaining a close link between logic and theory of mind.[10]

I hope that these two brief sketches outline how disparate logical systems in history may appear to be radically different, but that their differences are in part due to rejecting or modifying some basic philosophical (and non-logical) positions. These transitions can show

---

[8] He makes this case over the course of Book I.

[9] This technique is laid out in Book II.

[10] See Boole *The Laws of Thought* chapter 1. As an interesting philosophical divergence into the history of computing and artificial intelligence, I often have students learn and discuss the development of logic machines in the 1870s following Boole's logical system. These machines prompted a brief but lively debate amongst American logicians such as C. S. Peirce as to whether machines that followed the laws of thought actually were thinking. (see C. S. Peirce, "Logical Machines," *The American Journal of Psychology*, vol. 1, no. 1 (Nov. 1887): 165-170) The view seemed generally to be in the affirmative. Eventually, however, those doing algebra of logic dropped their commitment to linking logic to laws of thinking. (see, for example, chapter 1 in Louis Couturat, *The Algebra of Logic* (Chicago: Open Court Publishing, 1914))

students that historical logics are not useless or obsolete; instead, they are evolutionary predecessors to later logics (and ultimately contemporary classical logic).[11] These historical logics can still be valuable and useful to the contemporary student, should one embrace the philosophical commitments underlying the logical worldview.

Over the years, I have taught and analyzed a number of different historical logical systems and perspectives by means of a close examination of their philosophical commitments. In addition to those systems mentioned above, I have found that at least the following logical approaches can be analyzed in this way: Medieval faith and reason, propositional and predicate logic, mathematical physics (a la Newton), Ancient/Medieval problem of universals, nominalism, Leibniz's logical innovations, logical fatalism, logical atomism, logicism, and the theory of relations. I anticipate many more, if not most, historical logics and logical issues would also be amenable to this pedagogical approach.

4 Pedagogical Benefits of Teaching Logical Worldviews

In teaching introductory formal logic to undergraduates one may have encountered the following scenario: those students who have an affinity to formal reasoning (e.g. math, computer science, and science students) take rather well to logic, but other students (e.g. humanities and social science students) struggle. This should not be too surprising, given the mathematical basis for contemporary formal logic. But what to do with those students left behind? And though the power and flexibility of formal logic is well demonstrated, especially relative to earlier approaches to logic (such as Aristotelian logic), students that are not mathematically inclined find formal logic both confusing and limiting. Such students may be capable of reasoning well, and may have a great deal of experience and success in reasoning, but often find that their previously successful reasoning practices are not well captured by the methods found in classical formal logic. One solution to this mis-fit is addressed by courses in informal logic (sometimes called "critical thinking"). And though such an approach to reasoning more closely fits the more natural, intuitive, and practical forms of reasoning of interest to many college students, it typically achieves this result by minimizing or avoiding formal reasoning altogether.

The contemporary setting of formal logic in introductory undergraduate education is a difficult, if not paradoxical one. Students not already open to formal reasoning often find formal logic not very "useful", and many struggle to see the value or purpose of learning it. This is further compounded when formal logic is taught by philosophy departments, a discipline that otherwise frequently engages in applied reasoning in a variety of fields of contemporary relevance (science, religion, art, ethics, etc.), and in "big" questions that are quite meaningful to students in their lives ("What is the meaning of life?" "What is the best life one can live?" "What is the right thing to do?"). It is rare for a logic class to turn from the study of formal reasoning methods to connecting those to issues of current interest ("informal" or philosophical reasoning), and so students who do ultimately learn formal logic have great difficulty applying their new skills to questions of particular interest to them,

[11] By the way, in my textbook I use logical worldviews to show how contemporary propositional logic results from abandoning a commitment to a correspondence theory of truth and the resulting modification of the algebraic method of Boole's logic. This philosophical position is partly inspired by formalism – often identified with Giuseppe Peano and many of his Italian colleagues. See Paolo Mancosu, Richard Zach, and Calixto Badesa, "The Development of Mathematical Logic from Russell to Tarski, 1900-1935" in *The Development of Modern Logic*, edited by Leila Haaparanta, p. 318-470. Oxford: Oxford University Press, 2009.

unless they continue on with more advanced philosophy courses.

By understanding logical systems and methods in the context of their philosophical commitments, students are exposed to the close connection between logic and philosophy. They learn why logic has the method it does, which helps them engage and understand the material more deeply than simply challenging them to master the mathematical methods of contemporary formal logic. Further, students learning logical worldviews are more capable of knitting together formal logic and "informal" or philosophical reasoning, as these approaches are explicitly linked. Finally, logical worldviews demonstrate how logic is closely connected to philosophy, and it is rather easy to introduce philosophical questions of interest to students in the midst of learning a logical system.[12]

5 Benefits of Teaching Logical Worldviews to History of Philosophy

Most logic textbooks, and by implication, most undergraduate logic classes teach a system of formal logic developed in the 20th century (except, in some cases, a cursory examination of syllogistic reasoning). This approach to logic has some disturbing implications for the 2,500 years of philosophical reasoning that preceded the 20th century. The considerate student (as well as many professors) may draw any or all of the following conclusions about pre-20th century logical reasoning:

1. Logic from these time periods was either wrong, incomplete, or unacceptably limited in scope and methods. After all, if such logics were suitable, why develop contemporary formal logic that encompasses and surpasses such logics?
2. By extension from this first point, philosophical reasoning based on earlier logics could similarly be considered to be flawed, obsolete, or otherwise useless. Some schools of philosophical thought have explicitly embraced this perspective (e.g. the Logical Positivists); today, this perspective is often implicit in attitudes about historical philosophy from a wide variety of fields (every so often one will make such an assertion explicitly, as Stephen Hawking recently did).[13]
3. Many contemporary philosophers (students and professors) educated in contemporary formal logic enough to use it to parse examples of natural reasoning will use it to parse examples of philosophical reasoning that pre-dates mathematical logic. This is anachronistic to say the least, and results in a misleading conception of the actual reasoning made by these historic figures. Further, it is likely to construe historical philosophy in a bad light, since it makes it appear that these historical figures engaged regularly in invalid reasoning for most of the intellectual history of Western thought.

For some, these considerations are not of much concern: many do, in fact, believe that history is rife with examples of bad reasoning, just as up to the scientific revolution, history was plagued with examples of bad science. This perspective is reinforced by the (mostly

[12] For example, students can explore logical fatalism (and their more philosophically interesting versions concerning free will and determinism or divine foreknowledge) by exploring the consequences of Aristotle's commitment to a correspondence theory of truth and bivalence. Also, students can better understand Anselm's ontological argument and Guanilo's reply by seeing how Anselm's position is grounded in Aristotle's own reductio proofs, and Guanilo's on Aristotle's theory of mind and direct realism.

[13] See Stephen Hawking and Leonard Mlodinow, *The Grand Design* (New York: Bantam Books, 2010) chapter 1: "Traditionally these are questions for philosophy, but philosophy is dead. Philosophy has not kept up with modern developments in science, particularly physics. Scientists have become the bearers of the torch of discovery in our quest for knowledge."

legitimate) observation that some of history's most highly regarded philosophical reasoners seem to have made grave errors in their knowledge claims (Aristotle is a significant target here).

In math and science an ahistorical perspective may be appropriate: after all, most science and math textbooks teach the completed current state of knowledge, not the long and difficult (and in many cases wrong) path that led us here. Outside of these fields, however, the value of accurately understanding past perspectives is more apparent: many areas of study in the humanities and social sciences to this day engage well-reasoned theories dating back to Ancient Greece.

Setting concerns about history of philosophy aside, a large number of people presently endorse Aristotle's philosophical commitments enumerated above and teachers would likely find that most of those people would have little problem with syllogistic reasoning. In contrast, I expect that most teachers of logic find that many students find much of contemporary formal logic counter-intuitive and alien to how they reason. In contrast, logical worldviews can help the logic teacher show that in certain philosophical contexts, particular historic logics may be the best approach; but that should those philosophical commitments be rejected or disproved in other reasoning contexts, different logics are called for. A student's intuitive reasoning can be embraced rather than replaced, which would certainly seem to be a desirable outcome for a logic class.

6 Benefits for the Teacher of Logic

I'd like to finish up by discussing a few benefits for the logic teacher who adopts this approach. I've designed the textbook to contain a number of different modules, each examining a different logical worldview. As I complete more modules, the textbook will fill up with a fair number of unique instances of logical systems spanning the last 2,500 years. Each module contains several key components: (1) an inventory of the philosophical commitments of the logical system; (2) a trace of the connection between the philosophical commitments and the development and form of the logical method and principles employed; (3) an introduction and discussion of some interesting philosophical issues that arise under this worldview.

Each module is designed to be teachable by those with only a minimal expertise in the history of philosophy.[14] The characterization of a logical worldview does depend on a historical interpretation of the philosophy and logic under study, but a background in historical scholarship is not required in order to understand or teach the main components of each module. Instead, anyone with a good background in philosophy should be able to comprehend and easily teach the first and third components (inventory of philosophical commitments, and linkage to philosophical issues). The second component of each module (tracing the connection between the philosophy and the logic) is more subtle and complex, but should be within reach for anyone experienced in teaching logic.

Modules are designed to be able to be taught independently, although many are closely related to others. This allows the teacher to pick and choose a path through the history of logic, and thereby emphasize the impact of different philosophical views on the development of logic and/or our understanding of certain logical concepts and methods. For example, in a logic class that culminates in teaching propositional and predicate logic, one might

[14] As mentioned earlier, this involves presenting a simplistic interpretation which historians of philosophy might find unsupported by sufficient research and scholarship. However, most textbooks are likely subject to the same complaint, and thus this publication genre permits, if not requires, such oversimplification.

start with Aristotle and then pick modules that progressively show how logic became more mathematical and abstract (by for instance, selecting logical worldviews of Leibniz, Boole and Russell as transitional modules). On the other hand, a course interested in the logic of the scientific method might feature the logical worldviews of Aristotle, Medieval faith and reason, Bacon, and Newton.

In conclusion, I have found this approach to be of great benefit to students and to myself by expanding teaching of logic beyond the boundaries of formal methods and systems. Students seem to derive a deeper understanding of logic by seeing how it connects with other philosophical positions. Further, this approach helps make logic more comprehensible to the broad pool of students who are either disinclined or not well suited towards the strictly formal, mathematical approaches to teaching logic so commonly taught today.

Teaching natural deduction in the right order with Natural Deduction Planner

Jeremy Seligman[1] and Declan Thompson[2]

1 **The University of Auckland**
New Zealand
`j.seligman@auckland.ac.nz`
2 **The University of Auckland**
New Zealand
`dtho139@aucklanduni.ac.nz`

Abstract

We describe a strategy-based approach to teaching natural deduction using a notation that emphasises the order in which deductions are constructed, together with a LaTeX package and Java app to aid in the production of teaching resources and classroom demonstrations. Our approach is aimed at students with little exposure to mathematical method and has been developed while teaching undergraduate classes for philosophy students over the last ten years.

1998 ACM Subject Classification K.3.1 Computer Uses in Education

Keywords and phrases Natural deduction, strategy, proof assistant

1 Natural Deduction as a Creative Process

Teaching modern logic to students with little background in mathematics is notoriously hard. The philosophy student, adept at reading complex prose and composing artful essays is usually not well prepared for manipulating symbols and constructing rigorous proofs of theorems. Acquisition of at least the following three skills are needed.

The first is using the language of propositional and predicate logic to represent one's thoughts in formal notation and understand what has been written by others. This is usually achieved by learning to translate to and from natural language. Many resources are available.

The second is manipulating the symbols of formal notation according to precise rules. This is a basic skill necessary for almost all of logic, from applying mechanical methods of argument evaluation to acquiring an appreciation of the autonomy of the syntactic realm, without which the major theoretical results of logical theory cannot be understood. It can be acquired by learning how to produce truth tables, truth trees, and in many other ways. Again many resources are available.

The third is reading and writing rigorous arguments, of the kind used in mathematics. This is a much more difficult skill to acquire, requiring mastery of the first two skills, and in addition, a level of mathematical maturity that is attained by mathematics students only after years of practice with algebra, geometry, analysis, etc. Consequently, this side of logic education is often neglected by philosophy undergraduate programmes. Although many introductory textbooks include some discussion of logical theory, such as soundness and

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 199–206

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

completeness, the emphasis is on understanding the theorems rather than developing the skills to prove them. Few are aimed directly at acquiring the skill of creating proofs from scratch.[1]

One solution is to require logic students to take a substantial number of courses in mathematics, so that they acquire the necessary skills in the same way as mathematics students. In the long run, a broad experience with mathematical methods is certainly useful for research in logic, if not absolutely essential. But the huge gap that must be filled is daunting and dispiriting for most philosophy students, most of whom decide that it is just too big to breach.

Is there another solution? The obvious candidate is to teach students the skill of rigorous argumentation using the very formalisms that they have already learned: propositional and predicate logic. From a theoretical perspective, we know that our various systems of deduction can duplicate all that a mathematics student learns by a much more indirect and less explicit route. Why then is it so difficult for a philosophy student who has learned a formal system of deduction to transfer her skills to the production of informally rigorous arguments of the kind needed for progress in her subject?

It is generally recognised that axiomatic systems, while elegant and theoretically parsimonious, are wholly inappropriate for learning deduction. Instead, most logic programmes for philosophy students include some system of natural deduction, in which axioms are replaced by rules which mirror patterns of reasoning used in natural language argumentation. In the classic approach of Irving Copi, numerous rules are added, so as capture as many such patterns as possible.[2] Yet there is often an insufficient level of attention to any systematic discussion of the process of *creating* deductions. Typically, students are given an introduction to the rules, motivated by their natural language correlates, a few examples of complete deductions, and are then left to fend for themselves on a large number of exercises, with the hope that they will develop their own strategies by trial and error.

An alternative is to teach the *strategies* of creation explicitly. As well as helping students to learn formal deduction, these are the strategies that will prepare the student for the harder task of creating rigorous informal arguments of the kind needed to do postgraduate work in logic, and which mathematics students learn implicitly through their application to a wide range of mathematical topics. Teachers of natural deduction in the traditional style may be fully aware of this point, but the effective learning of explicit strategy is made almost impossibly hard by several factors.

The first is simply the number of rules used by logic textbooks aimed at mirroring patterns in informal reasoning, which include both proof by cases (Disjunction Elimination, $\vee\mathbf{E}$), and Disjunctive Syllogism, if not also Constructive Dilemma. While each of these is relatively easy to explain in isolation, the more rules, the harder it is to master their strategic interactions, which the student must consider when creating her own deductions.

A second, related factor is the lack of structure to the set of rules. From the perspective of teaching strategy, one would prefer a simpler set of rules, organised in a way that cor-

[1] A notable exception is "How to Prove It: A Structured Approach" [4, 5]. There are also countless introductions to mathematical method, e.g. "The Nuts and Bolts of Mathematical Proof" [2], "How to Read and Do Proofs: An Introduction to Mathematical Thought Processes" [3].

[2] "Introduction to Logic" [1] is now in its 14th edition. Many other textbooks on natural deduction employ a similarly lengthy list of rules.

responds to patterns of use in the creation of deductions, and exactly this is provided by Gentzen's original system, which uses the idea of introduction and elimination to expose the structure and symmetry of proof. There is no room here to explain and justify this choice, so we will just mention a few relevant points. Firstly, the fact that the intuitionistic fragment of the system has only a pair of rules for each logical operator allows one to develop general strategies: one for Introduction rules and one for Elimination rules, concerning the management of resources and simplification of goals. Moreover, an orthogonal classification of rules allows us to distinguish between cases in which a choice is required (e.g., $\vee$**I** and $\exists$**I**) and those that are 'automatic', in the sense that they can be applied without the need for further choice. Even the symmetry-breaking oddity of the non-intuitionistic rule $\neg\neg$**E**, which can be applied to any conclusion, raises an important strategic question: how to manage the creative steps of deduction? And this leads to an explicit discussion of back-tracking in problem solving and the need to recognise dead-ends. While such matters of strategy are implicit in more complicated systems, they are highlighted in systems in which the number of rules is small and well-balanced.

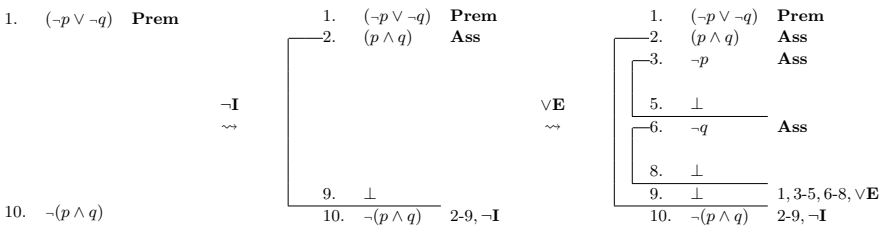
Standard presentations of Gentzen-style natural deduction, such as those of Fitch or Lemon, still have an important deficiency. Designed for reading rather than writing, the argument is displayed with the premises at the top, the conclusion at the bottom and with each line justified by lines higher up on the page, according to a formal rule. This makes the process of checking the correctness of the deduction relatively easy, but the process of generating the deduction itself unnecessarily hard.

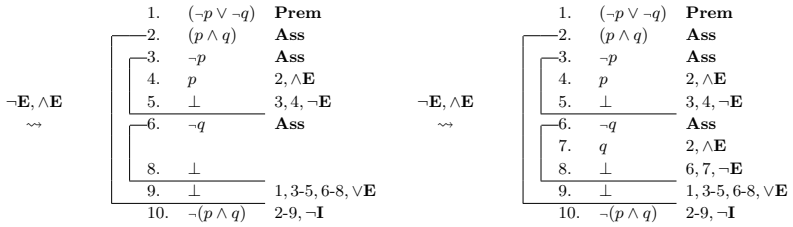
1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
3.	$\neg p$	**Ass**
4.	p	2, $\wedge$**E**
5.	$\bot$	3, 4, $\neg$**E**
6.	$\neg q$	**Ass**
7.	q	2, $\wedge$**E**
8.	$\bot$	6, 7, $\neg$**E**
9.	$\bot$	1, 3-5, 6-8, $\vee$**E**
10.	$\neg(p \wedge q)$	2-9, $\neg$**I**

*On the left is a correct deduction using a version of Gentzen's rules. Hypothetical reasoning is indicated by marking the assumption (***Ass***) and a vertical bracket ending below the hypothetical conclusion. The symbol $\bot$ is used to mark a contradiction.*

Information about the process of *creating* the deduction is lost in this representation, which wrongly suggests that it was written from top to bottom, starting with the premises and ending in the conclusion. (One of the most common mistakes made by students is to follow this order.) There is no record of the strategies used to construct the deduction; no record even of the order in which it was constructed. The student who fails to produce her own deduction of $\neg(p \wedge q)$ from $(\neg p \vee \neg q)$ will not learn much from looking at the above solution.

If we were to display a full sequence of steps leading to the creation of this deduction, we might write the following:





This is much too cumbersome for practical use in textbooks, and leaves the assignment of line numbers somewhat mysterious. How to know the deduction will use ten lines? But a simple change in notation can help. Instead of numbering the lines of the deduction from top to bottom, we number them in the order they were created. The above sequence can then be represented with just one deduction, as shown:

	1.	$(\neg p \vee \neg q)$	**Prem**
┌	3.	$(p \wedge q)$	**Ass**
│ ┌	5.	$\neg p$	**Ass**
│ │	9.	p	$3, \wedge\mathbf{E}$
│ │	6.	$\bot$	$5, 9, \neg\mathbf{E}$
│ ┌	7.	$\neg q$	**Ass**
│ │	10.	q	$3, \wedge\mathbf{E}$
│ │	8.	$\bot$	$7, 10, \neg\mathbf{E}$
│	4.	$\bot$	$1, 5\text{-}6, 7\text{-}8, \vee\mathbf{E}$
	2.	$\neg(p \wedge q)$	$3\text{-}4, \neg\mathbf{I}$

First, the premise and conclusion are written as lines 1 and 2, with a generous space between. We then apply $\neg\mathbf{I}$ to the conclusion to get a hypothetical deduction with assumption $(p \wedge q)$ on line 3 and conclusion $\bot$ on line 4. Next, we apply $\vee\mathbf{E}$ to line 1 to get two nested hypothetical deductions, from $\neg p$ on line 5 to $\bot$ on line 6, and from $\neg q$ on line 7 to $\bot$ on line 8. The first of these is completed using $\neg\mathbf{I}$ to get p on line 9 (justified by $\wedge\mathbf{E}$ from line 3). The second is completed similarly, with line q on line 10. In this way, the line numbers match the order of construction of the deduction precisely, which is thereby emphasised to students as they create it.

The discipline of numbering in the order a deduction is created helps students (and instructors) to think strategically. The goal is to provide a justification for the conclusion given the resources in the premises, and seen this way deduction is just planning how to use the resources to satisfy a goal. While this is a familiar idea in automated reasoning research, it rarely enters the classroom. By using the above system of numbering, students cannot avoid thinking in this strategic way and learning that introduction rules serve to split the goal into subgoals, whereas elimination rules deploy resources. Strategic concepts such as back-tracking, management of decision points, and an awareness of risk are brought to the fore. Certain rules, such as Disjunction Introduction are seen as “choice rules" to be used with caution and postponed as long as possible, whereas others, such as Implication Introduction are “automatic" - they can and should be used immediately with no risk of having to undo.

The use of a new notation has the disadvantage that teaching resources, especially solutions to exercises, have to be produced from scratch. And it was to aid in this that we decided to produce both a LaTeX package for formatting our deductions easily, and a Java app to aid in the generation of both LaTeX code and various other formats for classroom demonstration.

2 Natural Deduction Planner

Efficiently creating large numbers of typeset sample deductions can be a daunting prospect. On pen and paper, even a challenging proof can be completed within minutes. However typesetting a proof in software such as LaTeX requires a great deal more effort. With custom packages, structural features such as the scope lines used above can be automated well, but the task of inputting formulas is still cumbersome. Where the hand can draw any symbol at much the speed of any other, typesetting special characters often requires lengthy commands. We began development of a proof assistant software application with the primary goal of overcoming these difficulties, but the result is useful in many more ways than typesetting. We call the result the Natural Deduction Planner (NDP). It generates LaTeX code for use with a custom package.

Our interface essentially replicates the pen and paper proof process, using the same layout and notation of Gentzen's system, as above. Users input sequents using a set of special characters available onscreen. No special formatting (such as prefix notation) is required - a correctly inputted sequent appears as it would on the page. A range of proof systems are available, such as **NJ**, **NK** and Peano Arithmetic. Proofs appear graphically onscreen exactly as they would be typeset. Initially, any premises appear at the top of the window, with a space before the conclusion.

NDP is similar to the Proof Developer tool created by Daniel Velleman.[3] The interfaces are very alike, but where Proof Developer focusses on informal proof writing, NDP is concerned with formal deductions. Both approaches use ideas of strategy, goals and resources to conduct proofs. Another similar tool is PANDA, developed at the Institut de Recherche en Informatique de Toulouse.[4] PANDA uses a proof tree style, rather than the Fitch-style calculus implemented in NDP.

At each stage of an NDP deduction, the user can apply any valid rules, and their outcome is immediately displayed. To apply a rule, a *current goal* must first be selected, which can be any unjustified line. Selecting a current goal allows any possible introduction rules for that line to be applied. Selecting a goal also allows a *current resource* to be selected, enabling its elimination rules. Even if the first step in the proof involves an elimination rule, the user must select a current goal. These explicit realisations of goal and resource help to reinforce the use of strategies in deductions. Rather than beginning with premises working downwards, the user is encouraged to begin at the bottom of the deduction (the goal) and efficiently choose those resources which are needed. By breaking from a strictly linear approach, selecting goals encourages users to consider which available rules are automatic, and which require choice.

Lines are numbered in the order of creation, again reinforcing the way the deduction is constructed. This is also useful when reaching a dead-end in the proof - the user can see exactly how they reached this point, and what will happen when they retrace their steps. Highlighting is used to indicate the current goal and resource and also serves to indicate any lines out of scope of the goal. Once all lines have been justified the proof is complete.

[3] Proof Developer is a Java web applet built to accompany Velleman's textbook "How to Prove It" [4, 5] `http://www.cs.amherst.edu/~djv/pd/pd.html`

[4] PANDA is a Java application designed for teaching computer science students in logic, developed by Olivier Gasquet, François Schwarzentruber, and Martin Strecker `http://www.irit.fr/panda`

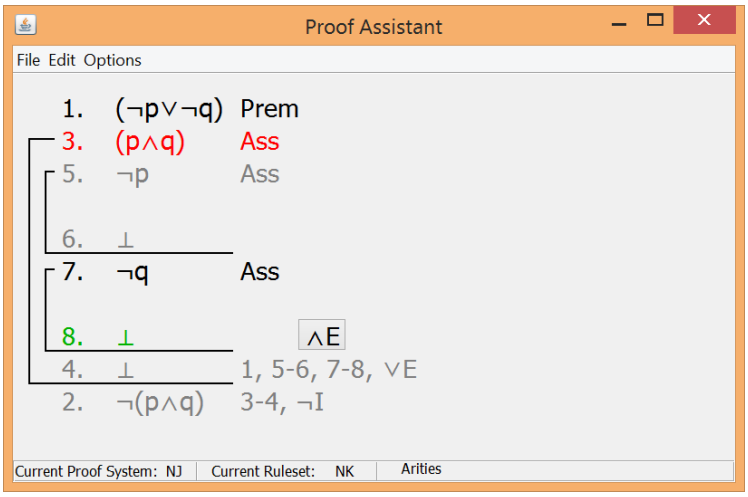


Figure 1 An incomplete proof showing current goal (green), current resource (red) and possible rules (∧**E**). All **NK** rules are available, but currently only **NJ** rules have been used.

By automating the writing of each proof line, users are able to move through deductions faster, focussing more on the strategy involved. Speed and the ability to easily "undo" mistakes also removes hurdles from the bulk trial and error method of learning strategy. A student unsure of the next step in a complicated pen and paper proof may be overly wary - a wrong move would result in writing out the whole proof again. On NDP, however, she can chose a rule in the knowledge that the current proof state can easily be retrieved. Similarly, students sometimes find rules like disjunction elimination, which requires creating four new lines and two scopes, to be intimidating and tiresome. Yet disjunction elimination is an automatic rule and should be applied as quickly as possible. In NDP disjunction elimination is achieved with two clicks, a less daunting task.

A *Rule Palette* allows individual rules to be (de)activated independently of the proof system chosen. The rule palette's layout shows the symmetry of Gentzen's rules, and gives some indication as to how the rules fit into different logical systems. By only activating certain rules, students can complete exercises in subsets of a system before being introduced to it fully, and see where certain rules are needed. For example, the rule palette can be used to demonstrate the importance of double negation elimination in **NK**, by attempting a proof of $(p \vee \neg p)$ without double negation elimination to see how far it goes. Once we get stuck, we turn on double negation elimination to finish the proof. Users can try out their own systems too, to see how different rules interact with each other.

Upon finishing a proof, it can be saved as either an editable proof or a demonstration proof. A demonstration proof has interaction disabled, providing a means to follow through an already complete proof. This is essentially the step by step deduction given above but in electronic form. Editable proofs behave similarly, but allow a user to take over the proof at any point. No work further than completing the deduction is required to generate these. Proofs can also be exported to unicode format and as an image. Complete proofs can also easily be animated in `.gif` format, for use in slides or online. A primary feature of NDP is its ability to export proofs to LaTeX code. This interacts with a LaTeX package (based on Ti*k*Z) which generates nicely typeset deductions. The task of producing exercises and their solutions involves little more than completing deductions on NDP - no fiddling about with

alignment or trying to recall commands required.

We have used NDP as part of a course teaching natural deduction strategies. All the deduction exercises for the course were generated by the software, and it was also made available for students to download. Many students did so, and used NDP to complete exercises and study for tests. We released exercise solutions both as text documents and editable proof files. A novel use NDP was put to was in catching up on missed lectures. Since NDP applies each rule correctly, by studying what happens students could learn the rule themselves. While the motivation and strategy discussed in lectures was absent here, the correct manipulation of the formula was learned. NDP's automated rule application had some downsides though. Some students found overuse of NDP resulted in over reliance - you don't have to remember how to set out implication introduction if the software does it for you. Since tests were by pen and paper, this proved problematic. The best combination seemed to be use of pen and paper to practise rules, and NDP to practise strategy.

In the context of tutorials, NDP allowed for greater flexibility in presentation. Again due to "undo" it was easier to recover from bad choices, encouraging student participation. Also, a source of potentially confusing transcription errors - the tutor's handwriting - was removed. In one on one situations, NDP allowed for a greater flow of conversation. Discussed strategies for stuck proofs could be implemented quickly and results considered in much less time than would be required to write 10 lines of formulas by hand.

Though originally intended to cover only propositional and predicate logic and Peano Arithmetic, we have begun extending NDP to cover other logics, and to consider new features. We've implemented a system of modal logic and hybrid logic using a labelled deduction method. These rules are available in the standard rule palette but are not thoroughly tested. A very rudimentary second order logic is also available, easily implemented due to Java seeing no distinction between predicate and variable symbols when making substitutions. In an attempt to automate the proof process, a *Magic Mode* is provided. This applies any rules which require no extra input for up to 10 iterations. In exceptional circumstances Magic Mode can complete proofs, but in general will only move forward one or two steps. Finally, a method to include custom axioms has been implemented.

2.1 Implementation

NDP was implemented in Java using the `Swing` and `SwingX` graphical user interface libraries. The code was written to be extendable and with a goal of modularity, to allow different interfaces to interact with the same backend.

Formulas are held as strings in a TEX macro format, using prefix notation for easier argument parsing. This also simplifies the process of exporting proofs to LATEX code. Each line of a proof is an `NDLine` object, which contains information such as the formula, the line number and the justification.

A `ProofMethods` class forms the core of the program. This holds the current proof state as an array of `NDLine`s. The application of a rule results in a relevant modification of the proof array and any lines within it. Rules themselves are methods within the `ProofMethods` class, and the system can be extended by adding new methods to give new rules. In practice, this means that additional rules (such as Disjunctive Syllogism) or extensions to the system can be added fairly easily. `ProofMethods` is designed to be as self-contained as possible; methods for printing the proof array to the command line mean it could be used without a

graphical interface. This is how early development proceeded.

On top of `ProofMethods` sits the `ProofPanel` class, a modified `JPanel` which provides user interaction with `ProofMethods`. `ProofPanel` interprets the proof array and arranges the deduction onscreen. The function of the rule palette is implemented entirely within the `ProofPanel`. If conjunction introduction ($\wedge\mathbf{I}$) is turned off then the option to apply that rule becomes unavailable on the `ProofPanel`. That is, even with $\wedge\mathbf{I}$ "disabled" `ProofMethods` is still able to apply that rule - there is just no way for the command to do so to reach it. The rule palette makes extensive use of the `SwingX` library.

A modified `JFrame` constitutes the main window of the Proof Assistant and controls tasks such as New Proof, Save, Open and Export. Proofs are saved in plain text files which contain complete undo histories and settings profiles. There is no difference between `.ndp` (editable) and `.ndu` (demonstration) files - they are read in differently but their contents are identical.

NDP is available on SourceForge at `http://sourceforge.net/p/proofassistant/`.

References

1 I.M. Copi, C. Cohen, and K. McMahon. *Introduction to Logic: Pearson New International Edition*. Pearson Education, Limited, 2013.
2 A. Cupillari. *The Nuts and Bolts of Proofs*. Elsevier Academic Press, 2005.
3 D. Solow. *How to read and do proofs: an introduction to mathematical thought processes*. Wiley, 2002.
4 D.J. Velleman. *How to Prove it: A Structured Approach*. Cambridge University, 1994.
5 D.J. Velleman. *How to Prove It: A Structured Approach*. Cambridge University Press, 2nd edition, 2006.

ARG: A Virtual Tool for Teaching Argumentation Theory

Nailton Silva[1], José Moura[2], and Patrick Terrematte[3]

1,2 Academic Center of Rio Grande do Norte State (UNI-RN)
Law Department
Natal – RN – Brazil
nailtongomes@ig.com.br, joseeduardomoura@unirn.edu.br

3 Federal University of Rio Grande do Norte (UFRN)
Metropole Digital Institute (IMD)
Group of Logic, Language, Information, Theory and Applications (LoLITA)
Natal – RN – Brazil
patrickt@imd.ufrn.br

Abstract

Researchers look for new virtual instruments that can improve and maximize traditional forms of teaching and learning. In this paper, we present the ARG system, a virtual tool developed to help the teaching/learning process in argumentation theory, especially in the field of Law. ARG was developed based on Araucaria by Reed and Rowe, Room 5 by Ronald P. Loui, as well on systems such as Argue!-System and ArguMed by Bart Verheij. ARG is a platform for online collaboration and applies the theory of Stephen Toulmin to produce arguments that are more concise, precise, minimally structured and more resistant to criticism.

1998 ACM Subject Classification K.3.1 Computer Uses in Education.

Keywords and phrases Teaching Argumentation Theory; Juridical Argumentation; Toulmin's Layout; Didactic Software.

1 Introduction

> *"The use of argumentation implies that one has renounced resorting to force alone, that value is attached to gaining the adherence of one's interlocutor by means of reasoned persuasion, and that one is not regarding him as an object, but appealing to his free judgment. Recourse to argumentation assumes the establishment of a community of minds, which, while it lasts, excludes the use of violence."*
> — Chaim Perelman

This study was initiated in mid-2010 to help undergraduate students understand the argumentative structures used by magistrates in the justification of their judicial decisions, adopting the accurate analysis of judgments pronounced by judges of the Rio Grande do Norte State Court as their methodology.

Following the senior thesis presented by Nailton Silva [6], this study ccome to three findings, which were crucial to the development of ARG. The first being (i) identification of fragility in the composition of those Magistrates' arguments which were uncritical and did not observe rules pertinent to any argumentation theory.

Further, it was also observed that the substantiation of judicial decisions were being organized based on the same patterns that were already used in the very beginning of juridical rhetoric which accentuates, as an argumentative structure, the "Legal Syllogism", creating the illusion of certainty in a sphere of uncertainty. Finally, it was noted that (iii) some arguments make use of assumptions and occult inference rules, establishing conclusions by implication.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 207–214


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

There are an obstacle for complete exercising of jurisdiction, of constitutional warrantees and credibility of the decisions and judicial institutions. Such conclusions served to be the justification for the search of higher rigor and consistency in the judicial determinations, and thus led to the development of the tool presented here.

We have analyzed how students of law are taught critical thinking and argumentation, focusing on argumentative structures used in the Brazilian legal practice. This analysis has led to the development of an electronic/virtual tool to assist in the task of argument construction. The aim of this work is to present the ARG system[1], a system designed to be a multilingual online tool able to assist in the production and improvement of practical arguments – not only those juridical in nature.

Our solution adopts Stephen Toulmin's theory of argumentation [7, 8], and we consider the value of his "layout" of argument, which enables us to visualize and follow the moves necessary for the construction of an argument of juridical practice. Toulmin's theory of argumentation, it is important to say, was already applied to compound electronic tools of representation and to helping in the composition of arguments[2], fitting very well when applied in numerous experiences of Artificial Intelligence and Argumentation and Law. This may be due to its heuristic properties, which is exploitable in information systems.

2 State-of-the-art argumentation tools

> *"We can't understand where we are now, without understanding how we got there. And of course, once we recognize the need to understand how we get to a particular point, we have to recognize also that the work of our successors will supersede our own ideas, and we must be modest in recognizing that the best we can do is indeed the best we can do."*
> — Stephen Toulmin. Reasoning in Theory and Practice, 2006

There are well-known uses of informatics in Law[3], such as in the Mechanical Jurisprudence, CCLIPS - Civil Code Legal Information Processing System, JUDITH, British Nationality Act, HYPO, Zeno, Room 5, PROSUPPORT, Araucaria and ArguMed projects. All of them essentially focus on the comprehension, representations and/or reproduction of judicial arguments. However, none of these serves as a solution for the problem highlighted in the previous section.

Four of these tools were the main tool available and have some of the functions of ARG. The first one was Room 5, developed by Ronald P. Loui, Jeff Norman, Joe Altepeter, Dan Pinkard, Dan Craven, Jessica Linsday and Mark Foltz, from Washington University. It was a website that provided a mechanism to represent, in a structured way, juridical argumentation, allowing visitors to make moves like those made by cases decided on the Supreme Court of the United States of America, inserted into a format that structures the discussion as either pro-author or pro-defendant [3].

The second and third systems that inspired us were ARGUE!-SYSTEM and ArguMed[4], which were systems developed by Bart Verheij [9, 10, 11], from the University of Maastricht in order to mediate the process and the argumentation of one or more users. In ARGUE!, the user provides argument data, such as assumptions, questions and reasons, and the system

[1] The ARG is available at `http://assistentearg.herokuapp.com/` with a video presentation at `http://youtu.be/BB0YDiOPam8`.
[2] See `http://www.phil.cmu.edu/projects/argument_mapping/`.
[3] About the subject see: `http://austhink.com/critical/pages/argument_mapping.html` and `http://www.phil.cmu.edu/projects/argument_mapping/`.
[4] Available at: `http://www.ai.rug.nl/~verheij/aaa/`.

determines the status of the justification: whether they were justified, unjustified, or neither [9]. Its successor, ArguMed, with a more transparent subjacent argumentation theory and a more intuitive interface, provides the user with the gradual construction of arguments, through the filling-in of patterns, each of them corresponding to an "argument move", such as adducing a reason or making a declaration [9].

Lastly, Araucaria[5], developed in 2001 by Chris Reed and Glenn Rowe [5] from the University of Dundee, is a tool that helps in (re)construction, typesetting and evaluation of arguments through a graphical interface ("point-and-click").

These tools, however, are limited to English speakers. Such applications do not include an interactive guide centered in on gradual composition and evaluation of arguments with the criterion of evaluation and improvement. They also do not provide an environment of discussion where students are able to explore the collective knowledge with technical support in theories of argumentation.

With analysis of the systems of automation, mediation and representation of reasoning encountered, it is possible to formulate the following conclusions that also serve the as basis of the new system: (i) arguments can be represented graphically through systems of electronic data processing; (ii) representation of arguments make their dialectical structure more evident; and (iii) theories of argumentation can be used in addition to an electronic tool to construct arguments.

In all argumentation situations we can use a methodology to clarify the structure of the arguments at stake. These tools previously mentioned took into account or created a theory of argumentation subjacent to their own judicial system. For similar purposes, ARG (abbreviation of "*argumentum*") was developed: with the clear objective of helping in the composition and management of arguments, and to address the root of the problem of inconsistency and fragility in argumentation that occur in Brazilian judicial practice. As a secondary objective, our tool also intends on supporting the teaching of critical thinking to students of philosophy, literature and science, helping their comprehension of theories of argumentation.

3 The Argumentation Assistant: ARG

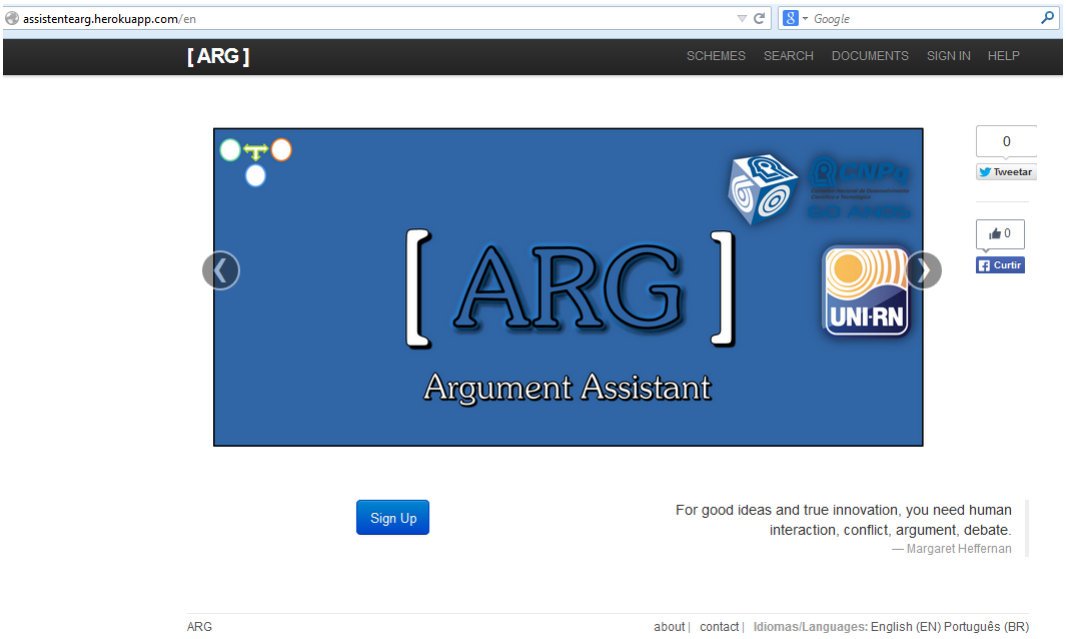
> *"Arguments are human interactions through which such trains of reasoning are formulated, debated, and/or thrashed out."*
> — Stephen Toulmin. An Introduction to Reasoning, 1979.

The aim was to construct and make public an online a multilingual teaching tool on the theory of argumentation based on participatory methodology with open source-code that could make it possible to store arguments and to visualize them in a clear and concise way. We planned on making available interactive guides to help users learn the techniques of arguing well by providing them with problems that stimulate them to think and produce good arguments. For this, we used the following technologies: (i) Ruby 1.9.3p125; (ii) Ruby on Rails 3.2.3; (iii) PostgreSQL 9.1.3; (iv) Github; and (v) Heroku. These tools were suitable for the development of multilingual information systems, are integrated with social media, and also have a large repository of documentation available online.

ARG is able to implement many functions. Some of them will be laid out briefly in this work, particularly the functions of (i) stimulation, (ii) evaluation, (iii) help/instruction and (iv) improving arguments.

[5] Available at: `http://araucaria.computing.dundee.ac.uk/doku.php`.

Figure 1 ARG's homepage.



The first function, stimulation, consists of prompting argument through cases provided by managers and/or professor-moderators of the system or by other academic students. Every week, for example, a new case might be introduced or a new problem regarding a previous case, opening a discussion where the users and moderators evaluate the arguments used, assigning a grade that it deems fitting, taking into account suggested criteria.

In ARG it is possible to post the abstract of a factual situation, attach elements of a police investigation and/or show a video and request, for example, that the users elaborate arguments that they have to compose. This could be an accusatory exordial (complaint) presented by the Prosecutor, for instance. After this, users could be asked to formulate the main argument of the answer to the accusation.

All the arguments posted are stored in an external data bank and accessible in real time, with daily indexation. We have also created a repository of arguments oriented to cases and specific themes.

Regarding the second function, in the evaluation system, each user or moderator can, besides comment, attribute a maximum grade of five stars or a minimum of one star to any argument according to the quality of reasoning presented, ideally observing the presence of the following properties:

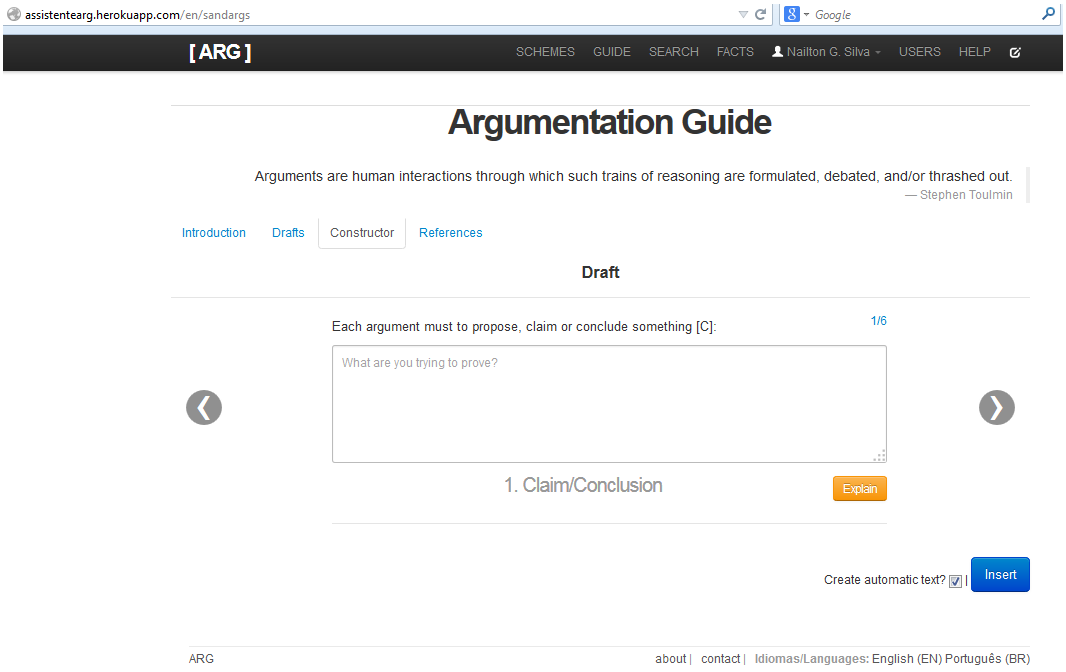
> 1. Clarity about what is considered in the argument; 2. Necessary and sufficient data to support the conclusion; 3. Relevant support and proof of the case under discussion; 4. Strength of the conclusion explicitly gained and the possible refutations. [8, 2]

In order to evaluate the arguments, it is necessary to observe that the following are avoided: "1. Absent data; 2. Irrelevant data; 3. Deficient data; 4. Unjustified suppositions; 5. Ambiguity"[8].

In regards to the third function, in ARG it is possible to teach users the basics of any theory of argumentation, as it offers an interactive guide oriented by the theory of Toulmin

(see Fig. 2). Through this guide, it is requested that the user fills in the fields to generate a "skeleton" of an argument, treated by ARG as a "draft" or "sketch".

Figure 2 ARG's page to guide the argumentative construction.



Finally, the fourth function focuses on improving arguments. ARG intends to help the prevention of eventual mistakes or neglect during its elaboration, disposing of a group of schemes of argumentation with many critical questions [12]. Usually argument schemes have a number of critical questions that must be answered satisfactorily when there is its application in a particular case. Some of those questions refer to the acceptability of the premises and other issues related to the exceptional circumstances in which the scheme cannot be applied [4].

4 Solving a case with ARG: the Speluncean Explorers

> *"The true revolution will arrive when computer systems now in development begin to perform or assist in the process of legal reasoning."*
> — Computers and Legal Reasoning. Garry S. Grossman and Lewis D. Solomon, 1983.

As a hypothetical scenario, let us suppose that we introduced on ARG the case of the Speluncean Explorers[6], and that the character J. Keen, Magistrate of the Supreme Court, needs help in constructing an argument for his vote.

[6] The work describes the judgment of an appellation resource where the accused were processed and condemned to death by the Court of Jury that recognized the practice of homicide. In synthesis, the victim was sacrificed by the accused to enable their survival, since they had been without any resources or food, involuntary and unpredictably for more than twenty days because of a landslide at the only access to a cave [1].

There is a page in the system that provides a guide for argumentation and an argument "constructor/builder". In this case, the user is asked to fill in the appropriate fields with elements that, according to Toulmin, compound any argument. So, Keen, stimulated by questions[7] and solicitations programmed in ARG, will fill the elements in with his arguments in the following way:

- **Claim** [C] = The sentence must be confirmed, with the maintenance of condemnation of the accused.
- **Grounds** [G] = (1) It is not of competence for the judge to measure if the act was just or unjust, good or bad, but to apply the law and not their convictions of morality; (2) The judiciary cannot create exceptions to the application of the law; (3) The judge cannot search the purposes of the law. In doing so, he legislates and, consequently usurps the competence of the Legislative Power; And (4) the conduct perpetrated by the accused fits the legal dispositive perfectly
- **Warrant** [W] = The judge must be loyal to the written law and interpret its evident meaning, without references to personal desire and its own concepts of justice.
- **Backing** [B] = Article 12-A of the criminal code.

The tool, considering the quantity of provided elements, will be able to generate an automatic text for the argument following the aforementioned guidelines:

> This is based on article 12-A [B] of the criminal code. We assume that [W] the judge is be loyal to the written law and interprets it on its evident meaning, without interferences or personal desires and own conceptions of justice. Given [G]: (1) it is not the duty of the judge to measure if the act was just or unjust, good or bad, but to apply the law and not its conceptions of morality; (2) the judiciary cannot create exceptions to the applications of the law; (3) it is not up to the judge to search the purpose of the law, but under the penalty of usurping the competence of the Legislative Power; and (4) the conduct perpetrated by the accused fits the legal dispositive, therefore, [C] it must confirm the sentence, keeping the condemnation of the accused ones.

Subsequently, offers the user a roll of verification [8] so that it is possible for the improvement of the argument. This roll corresponds to the structured abstract of what is necessary to compose a good argument, based on Toulmin [8] and Hitchcock [2].

Thus, it is up to the user to verify the nature of the argument that it intends to propose and to observe if it is proper to the form and to the main proper questions. So, J. Keen, the judge in this example, could consult the scheme "Argument from the Constitution of Causal Laws"[9] and realize its task by believing that this constitutes a good exercise to establish and comprehend the structures and the cares one must have with each argument.

In ARG, arguments are not treated as the instantiation of an abstract inference scheme, which would refer only to the evaluation of its formal aspects. We want to provide arguments schemes and critical issues that allow the user to make stronger arguments, resistant to

---

[7] I.e. "What are you trying to prove?" or "Why do you have those assumptions?"

[8] With questions such as: 1. Is it clear and sure the claim the argument tries to make? 2. Is there clarity and certainty in what is implicitly purposed (is there any purpose)?; 3. Are the data/reasons relevant?; 4. Are the data/reasons enough?; 5. Are the data/reasons justified?; among others.

[9] Available at: `http://araucaria.computing.dundee.ac.uk/downloads/schemesets/katzav-reed.scm` and `http://assistentearg.herokuapp.com/en/schemes`.

criticism, and avoid the incidence of reasoning faults - technical "argument-scheme approach" by Prakken [4].

With the purpose of testing ARG, a pilot study was carried out in November 2014. The task was applied to 206 undergraduate law students of the Academic Center of the Rio Grande do Norte State University (UNI-RN) and involved an adaptation of the quoted case of "the Speluncean Explorers". The students had to give an argument, from the point of view of J. Keen, in favor of or against the maintenance of the condemnation of the accused.

After this experiment, we compared the arguments before and after the use of the ARG-system. The academics who used the tool made arguments remarkably more concise, precise, minimally structured and more resistant to criticism. In addition, the system has not received negative reviews by users, as they did not report the occurrence of error in the application. However, we need more tests for more accurate conclusions.

In sum, the advantages of ARG include: (i) being available in a scale that goes beyond the classroom; (ii) being a useful tool for dissemination of knowledge, exposing the argumentation process in a concise way; and (iii) being a complement to face-to-face learning, as well as having online tasks, as a "blended learning" strategy.

5 Final Remarks

> *"The best we can do, that is, is to consider when we live, where we live, how we live, what the most reflective and best-informed experience of our colleagues in different areas has been, and what options are open in the future for the people who will come after us, and will revise and move beyond our ideas. "*
> — Stephen Toulmin. Reasoning in Theory and Practice, 2006.

ARG presents a system that can be used in any area of knowledge as an instrument of teaching critical thinking and argument production.

It is good to note that all the functionalities already implemented in ARG : (i) storing of arguments; (ii) visualization of arguments in a clear and concise way; (iii) interaction among users by tracking arguments of selected users; (iv) assistance in compounding arguments and help in improving them; (v) development of the intellectual faculties in argumentation theories; (vi) availability of multilingual scheme structures of arguments; and (vii) an evaluation system of arguments in two dimensions (community and moderators).

The system is available in two languages: English and Brazilian Portuguese, and is currently being translated into Spanish and French. In the future we aim to:

- Adapt anti-plagiarism solutions with the processing of language to identify similarities in the argumentative texts.
- Automate an exportation and importation of arguments from the professor/moderator.
- Integrate with a virtual environment of learning, e.g. the Moodle.

With such modifications, we hope to improve and transform it to a strong instrument for the learning of argumentation theory.

Acknowledgements The authors acknowledge all undergraduate law students who have contributed to the project during the Logic and Techniques of Argumentation course at UNI-RN. We would also like to acknowledge PIBIC/CNPq for their support of this research.

References

1 Lon L Fuller. The case of the Speluncean Explorers. *Harvard Law Review*, pages 616–645, 1949.

2 David Hitchcock. *Arguing on the Toulmin model: New Essays in Argument Analysis and Evaluation.*, chapter Good reasoning on the Toulmin model., pages 203–218. Springer, 2006.

3 R.P. Loui, J. Norman, J Altepeter, D. Pinkard, D. Craven, J. Lindsay, and M. Foltz. Progress on room 5: A testbed for public interactive semi-formal legal argumentation. *The Sixth International Conference on Artificial Intelligence and Law. Proceedings of the Conference. ACM, New York.*, pages 207–214, 1997.

4 Henry Prakken. Ai & Law, Logic and Argument Schemes. *Argumentation*, 19(3):303–320, 2005.

5 Chris Reed and Glenn Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14:961–980, 2004.

6 Nailton Gomes Silva. Argumentação e inteligência artificial: o desenvolvimento de um assistente de argumentação jurídica nacional, 2014. (Senior thesis. Department of Law).

7 Stephen Edelston Toulmin. *The Uses of Argument.* Cambridge University Press, 2003.

8 Stephen Edelston Toulmin, Richard Rieke, and Allan Janik. *An introduction to reasoning.* Macmillan Publishing Co., Inc. New York, 2003.

9 Bart Verheij. Argue! - An Implemented System For Computer-Mediated Defeasible argumentation. In *The Tenth Netherlands/Belgium Conference on Artificial Intelligence (NAIC '98*, pages 57–66, 1998.

10 Bart Verheij. Automated argument assistance for lawyers. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, pages 43–52. ACM Press, 1999.

11 Bart Verheij. *Virtual Arguments: On the Design of Argument Assistants for Lawyers and Other Arguers.* The Hague: T.M.C. Asser Pres, 2005.

12 Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation Schemes.* Cambridge University Press, 2008.

Logic Considered Fun

John Slaney

Australian National University
`John.Slaney@anu.edu.au`

Abstract

This report describes the development and use of an online teaching tool giving students exercises in logical modelling, or *formalisation* as it is called in the older literature. The original version of the site, 'Logic for Fun', dates from 2001, though it was little used except by small groups of students at the Australian National University. It is currently in the process of being replaced by a new version, free to all Internet users, intended to be promoted widely as a useful addition to both online and traditional logic courses.

1998 ACM Subject Classification K.3.1 Computer uses in education

Keywords and phrases Logic teaching, online learning, logical modelling, first order logic

1 Background: Formalisation

In teaching formal logic to undergraduates, we attempt to impart a range of skills. In a typical "Logic 101" course, the most prominent of these involve manipulation of calculi: devising proofs, usually using some form of natural deduction, constructing semantic tableaux or the like. We also ask students to formalise natural language sentences—often specially constructed to involve awkward nesting of connectives or strings of quantifiers—and may hope that they acquire some facility in critical reasoning and perhaps an appreciation of some wider issues connected to logic, be they mathematical, computational, philosophical or historical. Some of these things we teach better than others. Most students do become tolerably adept at handling the technical details of natural deduction proofs, but in many cases they remain depressingly unable to write a well-formed formula to express even a simple claim about a domain of discourse.[1] Barwise and Etchemendy, for instance, comment:

> The real problem, as we see it, is a failure on the part of logicians to find a simple way to explain the relationship between meaning and the laws of logic. In particular, we do not succeed in conveying to students what sentences in FOL mean, or in conveying how the meanings of sentences govern which methods of inference are valid and which are not. [1], p.13

We should find this situation alarming. Mechanical symbol-pushing for the purposes of simple proofs is easy to teach, fairly easy to learn and almost useless once the course is finished. On the other hand, the ability to read and write in the notation of formal logic, to use this as a medium for knowledge representation, to analyse and to disambiguate, is the most important skill students can take away from an introductory logic course, and it is a skill most of us can claim less success in teaching.

[1] Evidence for this claim is anecdotal, but strong. My own appreciation of it was sharpened in 2011, when analysis of results from a class of 61 students showed grades on proof construction that were on average above their grades for other courses, but after 13 weeks of study more than a third of them were unable to express 'The bigger the burger the better the burger' adequately in the notation of first order logic.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 215–222

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

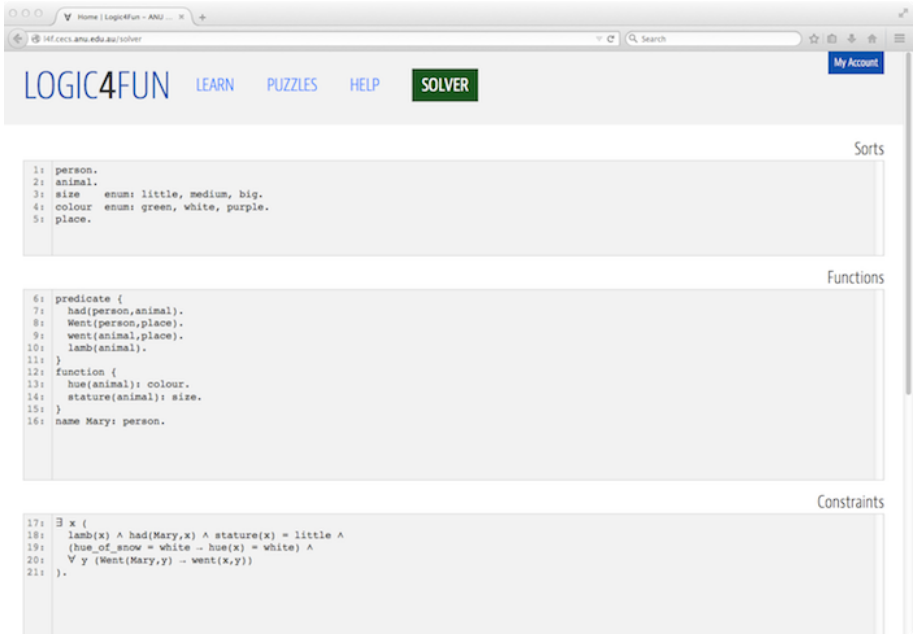


Figure 1 Screenshot of the window for problem input

It was against this background that the tool *Logic for Fun* [3, 4] was devised around 15 years ago. *Logic for Fun* is a website on which users are invited to express a range of logical problems and puzzles in such a way that a black-box solver on the site can produce solutions. The language in which problems are to be expressed is that of a many-sorted first order logic, extended slightly with a few built-in expressions and modest support for integers. The solver takes as input a set of formulae in this language and searches for finite models of this set. If it finds a unique model, this is almost certainly a solution to the problem. More usually it either reports syntax errors in the encoding or else says that there is no solution. The user (i.e. the student) then debugs the encoding until it is correct. This has several advantages over traditional formalisation exercises:

1. The sentences to be formalised constitute a meaningful problem, rather than looking like isolated examples of things it may be tricky to express formally;
2. Feedback is always accurate and *immediate*, rather than coming a week after the homework was handed in;
3. Because there is no feedback latency, because there is a goal (solving the problem) and because the machine is infinitely patient, students will put time and effort into their work, to a degree never seen in the traditional setting.

An example (not a puzzle in this case, but a first order theory) will help to illustrate the process required of the student. Figure 1 shows the form used for text input. It consists of three boxes: in the first are listed the "sorts" or domains over which variables are to range; in the second is the vocabulary of non-logical symbols (predicates, names, etc) with their types; in the third are the constraints, written as first order formulae. There are also buttons, not visible in the screenshot, for running the solver—one for generating solutions and another for running in a lightweight mode to check syntax. In the example, there are 5 sorts (persons, animals, sizes, colours and places) of which two (sizes and colours) are given by explicit enumeration while the other three are left for the solver to decide.

```
Sorts:
  person.
  animal.
  size enum:  little, medium, big.
  colour enum:  green, white, purple.
  place.
```

The domains of these sorts are required to be disjoint. We wish to say that Mary had a little lamb, so as vocabulary we declare 'Mary' as a name, 'had' as a relation between persons and animals, 'stature' as a function of animals and 'lamb' as a predicate picking out a subset of the animals:

```
Vocabulary:
  predicate {
    had(person,animal).
    Went(person,place).
    went(animal,place).     % note:  case-sensitive
    lamb(animal).
  }
  function {
    hue(animal):  colour.
    stature(animal):  size.
  }
  name Mary:  person.
```

For knowledge representation purposes, it is very convenient to use a many-sorted logic and to specify the types of predicates and function symbols separately from the formulae in which they occur. The hue and stature functions, for example, map animals to their colours and sizes; to model the constraints will be to determine which functions to assign to them.

To finish the example, here is the constraint:

```
∃ x (had(Mary,x) ∧ stature(x) = little ∧ lamb(x) ∧
    (hue_of_snow = white → hue(x) = white) ∧
    ∀ y (Went(Mary,y) → went(x,y))).
```

Having written the description, the student clicks "Solve" and the back-end reasoner (essentially a SAT solver adapted to finite domain first order problems) starts searching for solutions. Naturally, it finds models of this little theory with *very* small domains, including the expected interpretation in which there is one person (Mary), one animal (the little white lamb) and one place which neither of them visits. Amusingly, the solver also finds unexpected solutions, for instance in which the lamb is green—but it is still just as white as snow because the hue of snow is purple! In the pedagogic context, this provides a good opportunity for the teacher to make some points about interpretations and truth conditions and the semantics of the material conditional. When this kind of situation arises in modelling a puzzle which should admit only one solution, the student must invent more constraints to supply the missing information (e.g. that snow is white and that Mary went to school). In order to do this, they have to think about the semantics of the problem, render it into first order logic

and understand the relationship between the formulae they have written and the satisfying formal interpretations.

2 Structure of the exercises

The problems given as exercises on the site are divided into five levels: Beginner, Intermediate, Advanced, Expert and Logician. The boundaries between levels are not really definite, but students like the idea of progressing through levels in the style of a game. "Beginner" problems are fairly trivial to represent and solve, and are designed to get students through the phase of learning to use the site, teaching them how to declare vocabulary and the like. "Intermediate" ones are mainly logic puzzles of the kind found in popular magazines, often calling for bijections between sets of five or so things satisfying a list of clues. "Advanced" puzzles are not necessarily harder, but have features requiring more sophisticated logical treatment—nested quantifiers and the like. The "Expert" puzzles are more challenging, and include several state-transition problems from AI planning, for instance. They require students to supply less obvious vocabulary and axiomatisation to represent preconditions, postconditions and frame conditions of actions and the like. Finally, the "Logician" section contains problems which hint at applications of logic, for instance to finite combinatorics and to model-based diagnosis.

It is important that all of the suggested problems can be solved quickly by the software behind the site, without requiring coding tricks. This is because the aim is to teach correct logical expression, not constraint programming. Especially for some of the "logician" puzzles, efficiency does matter, as the underlying problems are NP-hard at best, and solver behaviour can be affected by non-obvious things like the order in which functions are declared, but as far as possible the site de-emphasises efficiency and instead lays stress on correctness.

An interesting feature of learning logical modelling in this way is that concepts are introduced in approximately the opposite order from that in the parallel lecture course. In a typical logic course, we work from the more abstract levels via propositional connectives, then to names, predicates, quantifier-variable notation, and then introduce identity as a special relation symbol and go on finally to deal with function symbols and general terms. We do not usually get as far as many-sorted logic. Logical modelling *starts* with sorts, equations, names and functions. Then generality is introduced (the universal quantifier is used much more than the existential one), and only after that the basic connectives. Introductory logic textbooks, though they vary greatly in emphasis and style, tend to follow the same overall direction as logic courses.[2] My experience of using *Logic for Fun* as part of a standard introductory logic course is that the reversed order of topics requires that the lecturer put work into pointing out the relationship between the logical modelling exercises and the rest of the course, as the two strands do not converge until the end.

3 Errors and feedback

The workflow of the site is one of dialogue between student and machine, whereby the problem in natural language is initially proposed as a challenge to which the student responds by writing formal encodings of all or part of the problem which the solver evaluates. Feedback in the form of error messages or solutions (or lack of solutions) informs the student's next

[2] There are exceptions. One of the most notable is the little introduction by Wilfred Hodges [2] used over 30 years ago as a textbook by the Open University and still in print.

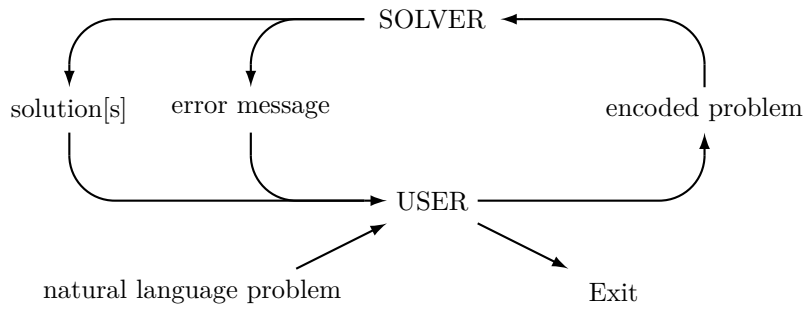


Figure 2 *Logic for Fun* workflow. The main loop is a dialogue between a user and the solver.

attempt. The cycle is broken when the student decides to terminate or suspend it. Work may be saved at any point for future reference.

Clearly, the educationally effective part of this process is the correction of errors. To put it simply: the tool is only doing useful work when its users are making mistakes. Feedback is therefore the essence of the process. Errors (apart from accidental slips) are fundamentally of two kinds, syntactic or semantic, evoking very different responses from the system. Errors of syntax are caught by the parser or the type checker and reported with explicit messages. For example, if the user writes

```
had(Mary, ∃x(lamb(x))).
```

(presumably trying to say "Mary had some x such that x is a lamb") the solver replies:

```
Input error on line 32:  had(Mary, SOME x lamb(x)).
Type mismatch with argument of had

Detailed diagnostics:  in the formula
    had(Mary,SOME x lamb(x))
the main operator "had" expects argument 2 to be of type animal
but argument 2 is
    SOME x lamb(x)
which is of type bool.
```

It would go on to suggest possible causes of this kind of error, advising for instance to check for misplaced parentheses and wrong names (not guilty in this case), and would also write out the parse tree of the formula as far as the parser was able to get before raising an exception. This kind of detail in error messages is an important feature, but such verbosity can become irritating so a future version of the site will place detailed drill-down under user control.

Semantic errors are harder to classify and harder to deal with. There are no "canned" solutions written in, so if the encoded problem gets past the parser and type checker, all the solver can do is search for solutions and report what it finds. Hence the only symptom of misunderstanding on the semantic level is an unexpected solution or (more often) no solution. This is the case whether the error is due to basic misunderstanding of semantics, such as confusion between the conditional and the biconditional, or whether it is a matter of problem representation—using logic correctly to say the wrong thing.

The case in which there is no solution is common, and of course the solver's response "No solution found" provides the user with a minimum of information. Techniques for making the feedback more informative include commenting out lines of the encoding and re-running to see whether solutions exist. This can sometimes be effective in pinpointing incorrectly expressed constraints, though it is laborious and the results are not always helpful. A diagnosis tool

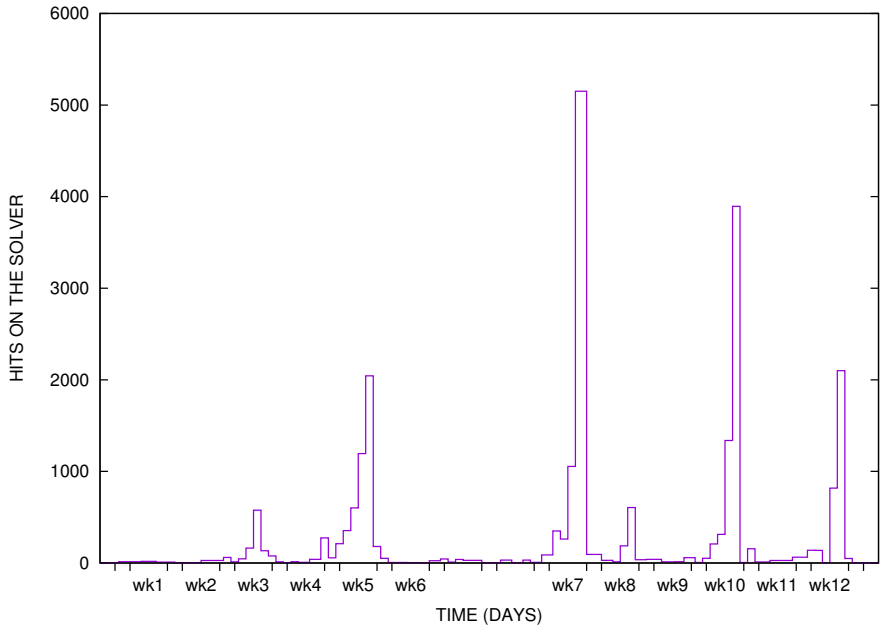


Figure 3 Number of times the solver was run on each day over 12 weeks of a logic course

designed to help automate the process of isolating incorrect constraints in cases where the encoded problem globally has no solution is currently under development: a prototype exists, but has not yet been incorporated into the website, so there are no results yet concerning its usefulness in practice.

4 Site usage

Logfiles produced by the scripts on the site can be mined for data on usage patterns, and provide some insight into how students set about mastering the web-based tool and using it to solve problems of logic. At the simplest, aggregated statistics for the number of hits on the site allow us to observe class behaviour. Figure 3 shows the number of times the solver was run each day by a class of around 50 undergraduates during a 12-week semester in 2013. Note that there was a 2-week break between weeks 6 and 7.

Students had a piece of homework to do each week, and had to hand it in for assessment before midnight each Friday evening. These assignments in weeks 5, 7, 10 and 12 consisted of problems to be solved using *Logic for Fun*, as can clearly be seen from the bursts of activity in those weeks. The assignment in week 8 was a problem concerning the semantics of some first order formulae, which the students were asked to compute on paper, using semantic tableaux, before comparing their answers with models of the same formulae produced by *Logic for Fun*. The small peak in week 3 is associated with the point at which they were introduced to the site and asked to complete some easy exercises to familiarise themselves with it. The homework in weeks 3, 4, 6, 9 and 11 did not call for students to use the site.

The heaviest usage of the semester occurred on the Friday of week 7, when the solver was run on average almost 100 times per student. This represents an extraordinary amount of work by the class on what was a fairly modest piece of logic homework. The problem in question (see Appendix A) was designed to turn on the correct handling of quantifiers. Since the usage log from that period only records who ran the solver at what time, not the

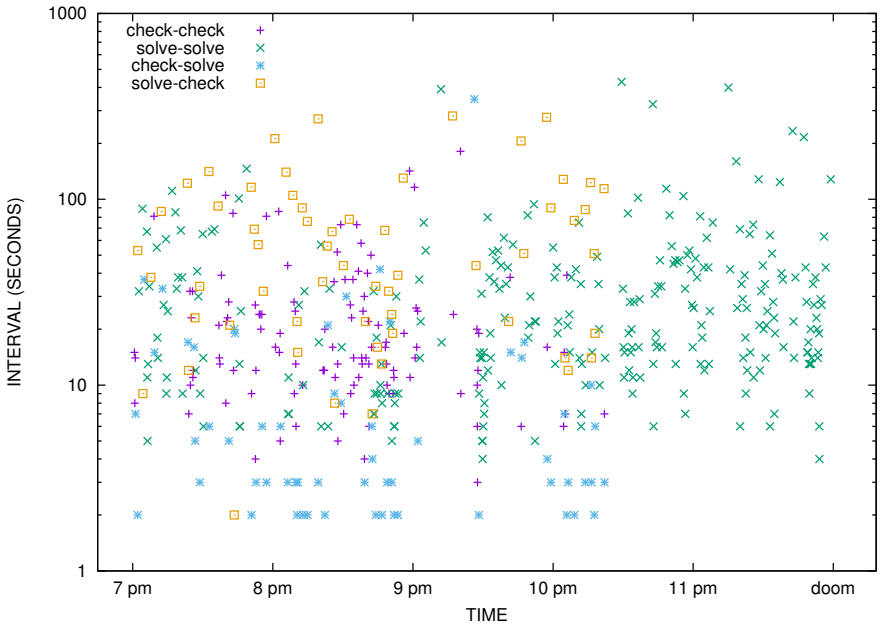


Figure 4 Five hours of work by one student. Each point is a run of the solver, showing the interval since the previous run plotted against the time of day. Different shapes show whether the run was a syntax check or a "solve", and whether the previous run was a check or a solve.

full text of what they sent to the solver, we have no way now of knowing what particular difficulties caused the class to spend so much more effort on this problem than on the others.

More detail is revealed by studying the work patterns of individual students. Some behaviours are quite striking: there are, for example, students who will run the solver 200 times or more on one problem, many of the runs being only seconds apart. Figure 4 shows an example of the activity of one such student over the five hours before the submission deadline. Note that this is a huge amount of work compared with the few minutes which students normally spend on a hand-written formalisation exercise. At no point during the five hours does this student pause for much more than 5 minutes. Most of the runs which occur within 10 seconds of the previous run are cases in which a syntax check is immediately followed by a "solve", presumably because there was no syntax error. We see some patterns in the record: for instance, at some point (a little before 10:30 pm) this student abandons explicit syntax checks and simply uses the "Solve" button. It is unclear why.

5 Current and future work

Logic for Fun was completely re-scripted in 2013–14, partly because the look and feel of the old site dated from another millennium, partly to extend its functionality in significant ways, and partly to have a version built with modern tools which would be easy to maintain. The new site is scripted entirely in Python, though the solver behind it is still the original, written in C some 20 years ago. The beta version of the new site [4] is now freely available to all web users, in contrast to the old site which required them to have accounts and to pay fees. This is in line with the contemporary expansion of free access to educational tools.

The biggest enhancement from the user's viewpoint is a facility to save and reload work, allowing it to be carried over easily from one session to another. Students who join a class

(called a "group" on the site) can also submit their work to the group manager for feedback or assessment. Low-level improvements, such as organising the display so that the natural language version of a puzzle can be in the same browser window as the student's logical encoding of it, also do much to enhance the user experience.

There is an ambitious plan for continued upgrading of the site. Tools currently under development include the diagnosis assistant already noted above, for use when a problem as written by the user has no solution. This tool allows the user to see minimal unsatisfiable cores of the problem, either on the high (first order) level or at lower levels from the clausified and compiled versions of the logic. It can also present approximate solutions—assignments of values which satisfy *almost* all of the constraints. A prototype of this tool looks promising, but incorporating it into the website raises issues for the user interface which have yet to be satisfactorily resolved. It is also planned that the system will maintain a detailed database of user activity, recording every character of text sent to the solver. This information will be used in a project aimed at deeper understanding of the logic learning process.

An important piece of future work, to be conducted when the new site is stable, is an effectiveness study. It is obvious that doing formalisation exercises online rather than on paper causes students to put much more effort into getting their answers right, but measuring the extent to which they learn more as a result is much harder.

Acknowledgements I wish to thank Matt Gray, Kahlil Hodgson, Daniel Pekevski, Nathan Ryan and Wayne Tsai for help in scripting *Logic for Fun*, and the many logic students over the last 15 years who have helped by testing it to destruction.

A Example: the homework in week 7

Logic Games

Our annual logic competition came to a final showdown between five teams: the Aces, Buccaneers, Cougars, Demons and Eagles. The contest was a round robin, each team meeting each other team once. At the end, the judges announced:

- Every team won at least once, and some team won all its games.
- The Buccaneers beat only the Cougars.
- Exactly one match was drawn, and it didn't involve the Cougars.
- The Aces defeated every team that the Eagles defeated, but they didn't defeat the Demons.
- Not every team that defeated the Aces defeated every team that the Aces defeated.

With that, they left us to work out the full set of results. Well?

References

1 Jon Barwise and John Etchemendy. *Language, Proof and Logic.* CSLI Publications, Stanford, CA, USA, 1999.

2 Wilfred Hodges. *Logic: An introduction to elementary logic, 2nd edn.* Penguin, London, 2005.

3 John Slaney. Logic for Fun (claasic version). `http://logic4fun.rsise.anu.edu.au/`.

4 John Slaney. Logic for Fun, Version 2 Beta. `http://L4F.cecs.anu.edu.au/`.

Euler diagrams as an introduction to set-theoretical models

Ryo Takemura

Nihon University
5-2-1 Kinuta, Setagaya-ku, Tokyo 157-8570, Japan
takemura.ryo@nihon-u.ac.jp

Abstract

Understanding the notion of a model is not always easy in logic courses. Hence, tools such as Euler diagrams are frequently applied as informal illustrations of set-theoretical models. We formally investigate Euler diagrams as an introduction to set-theoretical models. We show that the model-theoretic notions of validity and invalidity are characterized by Euler diagrams, and, in particular, that model construction can be described as a manipulation of Euler diagrams.

1998 ACM Subject Classification I.2.0 Philosophical foundations

Keywords and phrases Euler diagrams; Set-theoretical model; Counter-model

1 Introduction

Logic is traditionally studied from the different viewpoints of syntax and semantics. From the syntactic viewpoint, formal proofs are investigated using proof systems such as natural deduction and sequent calculus. From the semantic viewpoint, set-theoretical models of sentences are usually investigated. In contrast to a proof, which shows the validity of a given inference, we usually disprove an inference by constructing a counter-model. A counter-model is one in which all premises of a given inference are true, but its conclusion is false. The notions of proofs and models are traditionally defined in the fundamentally different frameworks of syntax and semantics, respectively, and the completeness theorem, one of the most basic theorems of logic, provides a bridge between them. In university courses, logic is usually taught along such lines.

As the notion of a proof appears naturally in mathematics courses, students are, to some extent, familiar with it. However, the notion of a model can be difficult for beginners. A set-theoretical model consists of a domain of entities and sets, and an interpretation function that assigns truth values to sentences. Models are usually defined symbolically in the same way as syntax, and beginners often find it difficult to distinguish between syntax and semantics, as well as to understand the notion of the interpretation of symbols. Thus, some diagrams are used to introduce set-theoretical models. One famous example is Tarski's World of Barwise and Etchemendy [1], which is designed to introduce model theoretic notions diagrammatically. Another traditional example concerns Euler diagrams, which were originally introduced in the 18th century to illustrate syllogisms. A basic Euler diagram consists of circles and points, and syllogistic sentences are represented using inclusion and exclusion relations between the circles and points. A circle in an Euler diagram can be considered to represent a set, and a point can be considered as an entity of a given domain. Inclusion and exclusion relations between circles and points can then be considered to represent set-theoretical notions such as subset and disjointness relations, respectively.

Besides the traditional informal semantic view, Euler diagrams have recently been investigated syntactically as the counterparts of logical formulas, which constitute formal

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 223–232


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

proofs. Euler diagrams are rigorously defined as syntactic objects, allowing set-theoretical semantics to be defined. Inference systems are also formalized, and they have been shown to be equivalent to some symbolic logical systems. Consequently, fundamental logical properties such as soundness and completeness have been investigated (e.g., [7, 5, 6]). In this way, Euler diagrams are, on the one hand, regarded as models of given sentences, and on the other hand, they are exploited as syntactic objects that constitute formal proofs.

In this paper, we present a formal investigation of Euler diagrams as set-theoretical models. We characterize the notions of validity and invalidity using our Euler diagrams. (In [2], Barwise and Etchemendy also suggested a possible application of Euler diagrams to construct counter-examples.) The advantages of our view of Euler diagrams as models are as follows: (1) diagrams provide concrete images of set-theoretical models and counter-models; (2) diagrams provide natural images of the interpretation of sentences; (3) model and counter-model construction can be captured as concrete manipulations of diagrams. However, there are the following limitations: (1) the expressive power of our diagrams is restricted to conjunctions of extended syllogistic sentences; (2) not all sets are represented by the circles or simple closed curves of our diagrams; (3) diagrams are sometimes misleading if their meaning and manipulations are not precisely defined. Regardless, Euler diagrams have recently been defined rigorously from logical viewpoints, and we believe that this restricted fragment is sufficient for an introduction to model theory.

In Section 2, we first review the Euler diagrammatic system of [6], in which Euler diagrams are regarded as syntactic objects corresponding to first-order formulas. Then, in Section 3, we investigate a view of Euler diagrams as counterparts of set-theoretical models. Finally, we investigate how Euler diagrams provide counter-models of given inferences in Section 4.

2 Euler diagrams as syntactic objects

Our sentences have the following extended syllogistic form:

a is B, a is not B, There is something B, There is something not B,
All A are B, No A are B, Some A are B, Some A are not B,

where a is a constant and A, B are predicates. These basic sentences are denoted by $S, S_1, S_2, \ldots$, and we also consider their conjunctions.

From a semantic viewpoint, every constant a is interpreted as an element $I(a)$, and a predicate B is interpreted as a *nonempty* set $I(B)$ in a set-theoretical domain. Then, our syllogistic sentences are interpreted as usual. For example, a is B is true iff $I(a) \in I(B)$; All A are B is true iff $I(A) \subseteq I(B)$, and so on (cf. [6]). Note that we impose the so-called *existential import*, i.e., the interpretation of B is nonempty. See Remark after Example 12.

We first review the Euler diagrammatic system of [6], in which Euler diagrams are regarded as syntactic objects corresponding to formulas.

▶ **Definition 1.** Our Euler diagram, called EUL**-diagram**, is defined as a plane with **named circles** (simple closed curves, denoted by $A, B, C, \ldots$) and **named points** ($p, p_1, p_2, \ldots$). Named points are divided into **constant points** ($a, b, c, \ldots$), which correspond to constants of the first-order language, and **existential points** ($x, y, z, \ldots$), which correspond to bound variables associated with the existential quantifier. Diagrams are denoted by $\mathcal{D}, \mathcal{E}, \mathcal{D}_1, \mathcal{D}_2, \ldots$.

Each diagram is specified by the following inclusion, exclusion, and crossing relations maintained between circles and points.

▶ **Definition 2.** EUL**-relations** consist of the following reflexive asymmetric binary relation $\sqsubset$, and the irreflexive symmetric binary relations $\vdash$ and $\bowtie$:

$A \sqsubset B$	"the interior of A is *inside of* the interior of B,"
$A \vdash B$	"the interior of A is *outside of* the interior of B,"
$A \bowtie B$	"there is at least one *crossing* point between A and B,"
$p \sqsubset A$	"p is *inside of* the interior of A,"
$p \vdash A$	"p is *outside of* the interior of A,"
$p \vdash q$	"p is *outside of* q (i.e., p is not located at point q)."

EUL-relations are denoted by $R, R_1, R_2, \ldots$.

The set of EUL-relations that hold on a diagram $\mathcal{D}$ is uniquely determined, and we denote this set by $\mathsf{rel}(\mathcal{D})$. For example, $\mathsf{rel}(\mathcal{S}_1 + \mathcal{S}_2)$ in Example 3 is $\{x \sqsubset A, x \sqsubset B, x \sqsubset C, A \bowtie B, A \bowtie C, B \sqsubset C\}$. In the description of $\mathsf{rel}(\mathcal{D})$, we omit the reflexive relation $s \sqsubset s$ for each object s. Furthermore, we often omit relations of the form $p \vdash q$ for points p and q, which always hold by definition. In the following, we consider the equivalence class of diagrams in terms of the EUL-relations.

To clarify the intended meaning of our diagrams, we describe the translation of diagrams into the usual first-order formulas. See [10, 11] for a detailed description of our translation. Each named circle is translated into a unary predicate, and each constant (resp. existential) point is translated into a constant symbol (resp. variable). Then, each EUL-relation R is translated into a formula R° as follows:

$$(p \sqsubset A)^\circ := A(p); \qquad (p \vdash A)^\circ := \neg A(p);$$
$$(A \sqsubset B)^\circ := \forall x(A(x) \to B(x)); \qquad (A \vdash B)^\circ := \forall x(A(x) \to \neg B(x));$$
$$(A \bowtie B)^\circ := \forall x((A(x) \to A(x)) \wedge (B(x) \to B(x))).$$

Let $\mathcal{D}$ be an EUL-diagram with the following set of relations: $\mathsf{rel}(\mathcal{D}) = \{R_1, \ldots, R_i, x_1 \square A_1, \ldots, x_1 \square A_k, \ldots, x_l \square A_1, \ldots, x_l \square A_k\}$, where $\square$ is $\sqsubset$ or $\vdash$, and no existential point appears in $R_1, \ldots, R_i$. Then, the diagram $\mathcal{D}$ is translated into the following conjunctive formula:

$$\mathcal{D}^\circ := R_1^\circ \wedge \cdots \wedge R_i^\circ \wedge \exists x_1(\overline{A_1(x_1)} \wedge \cdots \wedge \overline{A_k(x_1)}) \wedge \cdots \wedge \exists x_l(\overline{A_1(x_l)} \wedge \cdots \wedge \overline{A_k(x_l)}),$$

where $\overline{A(x)}$ is $A(x)$ or $\neg A(x)$ depending on $\square$.

Note that our diagram is abstractly a "conjunction of relations." Note also that we interpret the $\bowtie$-relation so that it does not convey any specific (inclusion or exclusion) information about the relationship between circles. (This interpretation is based on the convention of Venn diagrams. See [6] for further details.) Thus, $A \bowtie B$ is translated into a tautology as above.

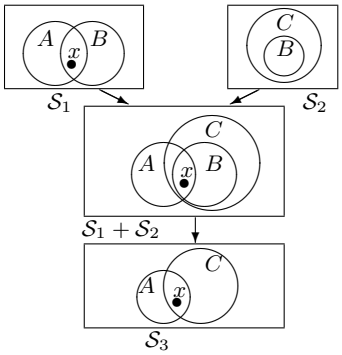
From a semantic viewpoint, by interpreting circles (resp. points) as *nonempty* subsets (resp. elements) of a set-theoretical domain, each "syntactic object" EUL-diagram is interpreted in terms of the relations that hold on it. See [6] for details.

▶ Remark. Our system is essentially the same as the *region connection calculus RCC* [8]. RCC is a topological approach to qualitative spatial representation and reasoning, and is applied, for example, to Geographic Information System (GIS). RCC investigates eight basic relations, including our inclusion, exclusion, and crossing (partially overlapping), between spatial regions (circles in our framework). Although RCC investigates general n-dimensional spaces of spatial regions, we concentrate on 2-dimensional diagrams. Thus, without any named points, our system can be considered as a subsystem of RCC. See, for example, [3, 4] for surveys of RCC.

We next review the Euler diagrammatic inference system of [6], called the Generalized Diagrammatic Syllogistic inference system GDS. GDS consists of two kinds of inference rules: Deletion and Unification. Deletion allows us to delete a circle or point from a given diagram. Unification allows us to unify two diagrams into one diagram *in which the semantic information is equivalent to the conjunction of the original two diagrams.*

▶ **Example 3.** Some A are B, All B are $C \models$ Some A are C

The validity of the given inference is shown by the Unification of the two diagrams $\mathcal{S}_1$ and $\mathcal{S}_2$, which represent the given premises. In this unification, circle B in $\mathcal{S}_1$ and $\mathcal{S}_2$ is identified, and C is added to $\mathcal{S}_1$ so that B is inside C and C overlaps with A *without any implication of a specific relationship between* A *and* C. $\mathcal{S}_3$, which represents the given conclusion, is then obtained by deleting B.



Two types of constraint are imposed on Unification. One is the *constraint for determinacy*, which blocks disjunctive ambiguity with respect to the location of points, and the other is the *constraint for consistency*, which blocks representing inconsistent information in a single diagram. Unification can only be applied when these constraints are satisfied.

The notion of **diagrammatic proof**, or **d-proof** for short, is defined inductively as a tree structure consisting of Unification and Deletion steps. The provability relation $\vdash$ is defined as usual in terms of the existence of a d-proof. See [6, 9] for details.

GDS is shown to be sound and complete with respect to our set-theoretical semantics. To avoid introducing the diagrammatic counterpart of the so-called absurdity rule in our system, we impose a consistency condition on the set of premise diagrams $\mathcal{D}_1, \ldots, \mathcal{D}_n$ in our formulation of completeness, i.e., the set of premise diagrams has a model. See [6, 9] for further discussion and a detailed proof.

▶ Proposition 1. Let $\mathcal{D}_1, \ldots, \mathcal{D}_n$ be consistent. $\mathcal{E}$ is a semantic consequence of $\mathcal{D}_1, \ldots, \mathcal{D}_n$ ($\mathcal{D}_1, \ldots, \mathcal{D}_n \models \mathcal{E}$) if and only if $\mathcal{E}$ is provable from $\mathcal{D}_1, \ldots, \mathcal{D}_n$ ($\mathcal{D}_1, \ldots, \mathcal{D}_n \vdash \mathcal{E}$) in GDS.

Note that the above completeness is obtained by regarding diagrams as "syntactic objects", corresponding to first-order formulas, and by defining appropriate set-theoretical semantics. We next investigate a view of Euler diagrams as counterparts of set-theoretical models.

3 Euler diagrams as models

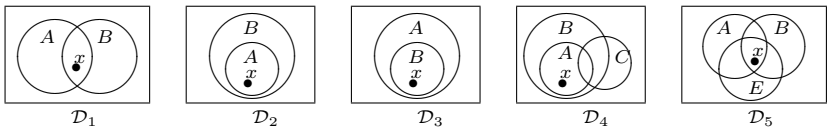
By regarding our diagrams as models, we define the truth condition for our sentences.

▶ **Definition 4.** Constants are interpreted as named points, predicates are circles. Then, for a diagram $\mathcal{D}$:

- a is B holds on $\mathcal{D}$ iff named point a is located inside circle B;
- a is not B holds on $\mathcal{D}$ iff named point a is located outside circle B;
- All A are B holds on $\mathcal{D}$ iff $A \sqsubset B$ holds on $\mathcal{D}$;
- No A are B holds on $\mathcal{D}$ iff $A \dashv B$ holds on $\mathcal{D}$;
- Some A are B holds on $\mathcal{D}$ iff there exists an existential point x in the intersection of A and B;
- Some A are not B holds on $\mathcal{D}$ iff there exists an existential point x inside A and outside B.

We write $\mathcal{D} \Vdash S$ when sentence S holds on diagram $\mathcal{D}$.

▶ **Example 5.** Some A are B holds on each of the following diagrams, for example.



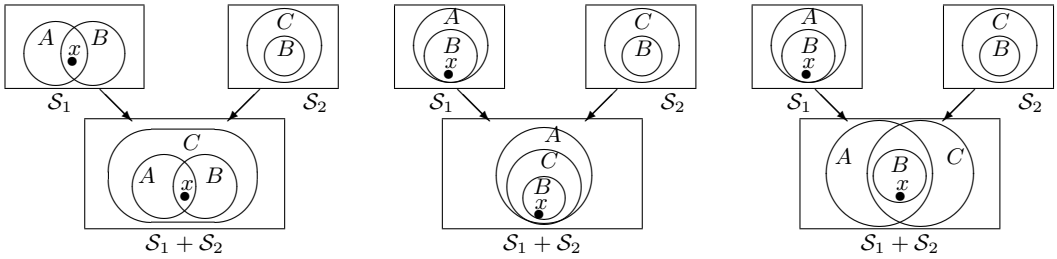
We define the notion of **Euler diagrammatic validity** in the same way as model-theoretic validity.

▶ **Definition 6.** For $n \neq 0$, $S_1, \ldots, S_n \Vdash S$ if S holds on any diagram $\mathcal{D}$ on which $S_1, \ldots, S_n$ hold (i.e., $\forall \mathcal{D}. \mathcal{D} \Vdash \bigwedge S_i \Rightarrow \mathcal{D} \Vdash S$).

In our Unification of GDS, as illustrated in Example 3, when the relationship between circles cannot be determined as $\sqsubset$ or $\vdash\!\dashv$ from the given premises, we assign this to be a $\bowtie$-relation without any implication. However, if we extend the notion of "unification" to allow additional implications (and hence, such that this "unification" is invalid) as long as all relations on premises are preserved, then we can describe model construction in terms of the extended "unification" as follows.

▶ **Example 7.** Some A are B, All B are $C \Vdash$ Some A are C

Various diagrams can represent the given premises, and there are various ways to "unify" these diagrams (using Deletion if necessary). The following describes three possible diagrams, and the conclusion Some A are C holds in each of them. Cf. Example 3, where the representations of sentences are fixed canonically in a one-to-one correspondence depending on each sentence, and the way of unification is also fixed uniquely.



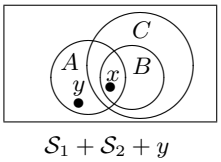
It is shown that the canonical form of the diagram for every sentence (e.g., $\mathcal{D}_1$ in Example 5 for Some A are B) is sufficient to consider the usual model-theoretic validity. Then, through the completeness of GDS, it is shown that Euler diagrammatic validity of Definition 6 is equivalent to the usual model-theoretic validity.

4 Euler diagrams for counter-models

Models can be practically applied to disprove given inferences. Let us consider how to disprove a given inference using Euler diagrams.

▶ **Example 8.** Some A are B, All B are $C \not\models$ All A are C

These premises are the same as in Example 3. To construct a diagram to falsify the given conclusion All A are C, we add a fresh existential point y to the unified diagram $\mathcal{S}_1 + \mathcal{S}_2$ of Example 3 as follows.



The existential point y denotes that "there exists something that is A but not C," i.e., the negation of All A are C. Note that even after the addition of y, all of the $\sqsubset, \vdash$ relations that hold on the premises ($x \sqsubset A$ and $x \sqsubset B$ on $\mathcal{S}_1$, and $B \sqsubset C$ on $\mathcal{S}_2$) are maintained. Hence, the above $\mathcal{S}_1 + \mathcal{S}_2 + y$ neatly illustrates a counter-model of the given inference in which all premises are true but the conclusion is false.

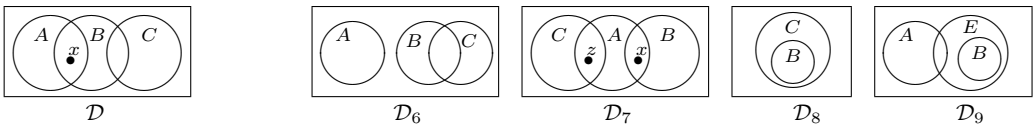
Let us define our counter-diagrams.

▶ **Definition 9.** A diagram $\mathcal{E}$ is a **counter-diagram** of $\mathcal{D}$, and vice versa, when one of the following holds between $\mathcal{D}$ and $\mathcal{E}$:

- $a \sqsubset B$ holds on $\mathcal{D}$, and $a \vdash B$ holds on $\mathcal{E}$;
- $A \sqsubset B$ holds on $\mathcal{D}$, and, for some x, $x \sqsubset A$ and $x \vdash B$ hold on $\mathcal{E}$;
- $A \vdash B$ holds on $\mathcal{D}$, and, for some x, $x \sqsubset A$ and $x \sqsubset B$ hold on $\mathcal{E}$.

Note that our diagram is abstractly a conjunction of relations, and the above definition lists the so-called "counter-relation" for every relation. As illustrated in Example 10, there may be a number of counter-diagrams for a given diagram $\mathcal{D}$, because a counter-diagram of $\mathcal{D}$ may contain circles and points that are irrelevant to $\mathcal{D}$. Thus, in general, the notions of a counter-diagram of $\mathcal{D}$ and the negation of $\mathcal{D}$, which should be uniquely determined, are different.

▶ **Example 10.** Let $\mathcal{D}$ be the following diagram on the left. Then, all of $\mathcal{D}_6, \mathcal{D}_7, \mathcal{D}_8$, and $\mathcal{D}_9$ are counter-diagrams of $\mathcal{D}$. Thus, a diagram and its counter-diagram do not necessarily share the same circles and points, and it is sufficient that a relation that holds on a counter-diagram falsifies a relation of the given diagram.



We can list all of the operations to construct counter-diagrams, such as the addition of a new point illustrated in Example 8. However, let us consider another way to construct counter-diagrams. Theoretically speaking, $S_1, \ldots, S_n \nVdash T$ means that there exists a $\mathcal{D}$ such that $\mathcal{D} \Vdash S_1 \wedge \cdots \wedge S_n \wedge \neg T$. Based on this fact, we construct a diagram that falsifies the given conclusion, and, by unifying the diagram with all premises, we construct a counter-diagram for the given inference. We formalize our counter-diagram construction according to this idea. Let us reconsider the inference in Example 8.

▶ **Example 11.** Some A are B, All B are $C \not\models$ All A are C
Let $\mathcal{E}'$ be a diagram such that $\mathsf{rel}(\mathcal{E}') = \{y \sqsubset A, y \vdash C, A \bowtie C\}$, which represents *Some A are not C*, and which falsifies the given conclusion All A are C. We unify this $\mathcal{E}'$ with $\mathcal{S}_1$ and $\mathcal{S}_2$, as illustrated in Fig. 1. Thus, the resulting diagram $\mathcal{S}_1 + \mathcal{S}_2 + \mathcal{E}'$ is the same as $\mathcal{S}_1 + \mathcal{S}_2 + y$ in Example 8, and disproves the given inference.

Let us examine another example.

▶ **Example 12.** No A are B, There is something $C \not\models$ No A are C
Let $\mathcal{E}'$ be a diagram such that $\mathsf{rel}(\mathcal{E}') = \{y \sqsubset A, y \sqsubset C, A \bowtie C\}$, which represents Some A are C, and which falsifies the given conclusion No A are C. We then unify this $\mathcal{E}'$ with the given premise $\mathcal{S}_3$, representing No A are B, as illustrated in Fig. 2. When we further attempt to unify diagram $\mathcal{S}_4$, in which $\{x \sqsubset C\}$ holds, representing There is something C, we find

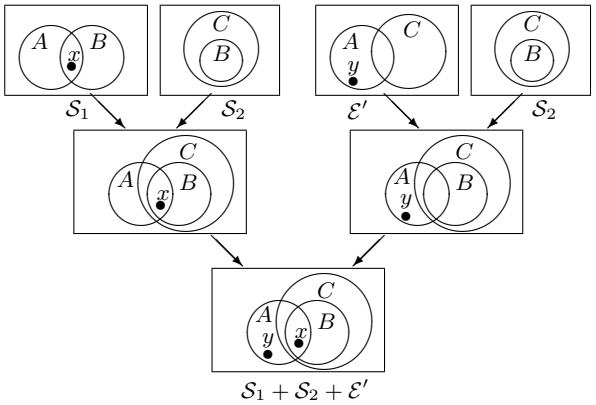


Figure 1 Construction of a counter-diagram (Example 11)

three possibilities for the position of x, that is, x is indeterminate with respect to circles A and B, and we cannot unify this from our valid Unification rules. However, whichever of the three possibilities gives the position of x, we obtain a counter-diagram to disprove the given inference. For example, $\mathcal{E}''$ in Fig. 2 is a required counter-diagram.

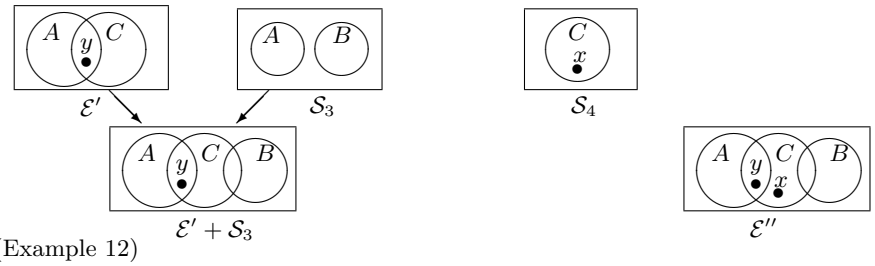


Figure 2 (Example 12)

Because we cannot obtain the above $\mathcal{E}''$ from $\mathcal{E}' + \mathcal{S}_3$ and $\mathcal{S}_4$ using our valid Unification rules of GDS, we construct counter-diagrams by introducing another "invalid" rule: invalid Point Insertion (iPI). This arbitrarily fixes the position of a point from several possibilities. Then, we obtain $\mathcal{E}''$ from $\mathcal{E}' + \mathcal{S}_3$ and $\mathcal{S}_4$.

In this way, a given inference is shown to be invalid by: (1) constructing a diagram that falsifies the given conclusion; and (2) unifying the diagram of (1) with all premises. In the resulting diagram, the relation falsifies the given conclusion and all of the $\sqsubset$ and $\vdash$ relations of the given premises hold. Hence, it is a counter-diagram for the given inference.

▶ Remark. In our system, we postulate the so-called *existential import* in the literature of syllogisms. With this postulate, for example, we have All A are B, No B are C $\models$ Some A are not C, even though this is not valid from the usual logical viewpoint without the existential import. Without this postulate, two diagrams, say $\mathcal{D}$ in which $A \sqsubset B$ holds and $\mathcal{E}$ in which $A \vdash B$ holds, are consistent when A denotes the empty set. However, it is difficult to express $\mathcal{D}$ and $\mathcal{E}$ in a single diagram, as our system lacks a device to express the emptiness of circles. Even if we introduce another device such as "shading", which expresses the emptiness of the corresponding region (cf. [7]), there remains a question as to whether we draw the shaded circle A inside or outside B (whichever is legal in an abstract sense). Thus, it seems that the assumption of the existential import is natural for our Euler diagrammatic system.

Let us define our counter-diagrammatic proofs.

▶ **Definition 13.** A **counter-diagrammatic proof**, or **counter-d-proof** for short, of $\mathcal{E}$ under $\mathcal{D}_1, \ldots, \mathcal{D}_n$ is a tree structure:

- consisting of valid Unification rules and invalid iPI;
- whose premises are $\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{E}'$, where $\mathcal{E}'$ is a counter-diagram of $\mathcal{E}$ such that no existential point of $\mathcal{E}'$ appears on $\mathcal{D}_1, \ldots, \mathcal{D}_n$;
- in whose conclusion, all $\sqsubset, \vdash$-relations of $\mathcal{D}_1, \ldots, \mathcal{D}_n, \mathcal{E}'$ hold.

We write $\mathcal{D}_1, \ldots, \mathcal{D}_n \dashrightarrow \mathcal{E}$ when there exists a counter-d-proof of $\mathcal{E}$ under $\mathcal{D}_1, \ldots, \mathcal{D}_n$.

Note that we may not use the invalid iPI as illustrated in Example 11. Additionally, we can freely choose a counter-diagram $\mathcal{E}'$ to add to the premises of a counter-d-proof. Hence, it may be the case that $\mathcal{E}'$ is itself a required conclusion of a counter-d-proof. In our proof of Proposition 2 below, we present a canonical method to choose $\mathcal{E}'$ and construct counter-d-proofs. See [11] for further details.

Although space limitations prevent us from going into detail, there exists an invalid inference for which we cannot construct a counter-d-proof in our framework of inclusion, exclusion, and crossing relations. Thus, we need to restrict the position of existential points appearing in a given conclusion. We call a diagram "relational" if the position of every existential point in the diagram is determined by relations with at most two circles. See [11] for a detailed discussion. Thus, we have shown that our counter-d-proofs sufficiently characterize the notion of invalidity.

▶ Proposition 2. Let $\mathcal{D}_1, \ldots, \mathcal{D}_n$ be consistent, and $\mathcal{E}$ be relational. $\mathcal{E}$ is not a valid conclusion of $\mathcal{D}_1, \ldots, \mathcal{D}_n$ (i.e., $\mathcal{D}_1, \ldots, \mathcal{D}_n \not\models \mathcal{E}$) if and only if there exists a counter-d-proof of $\mathcal{E}$ under $\mathcal{D}_1, \ldots, \mathcal{D}_n$ (i.e., $\mathcal{D}_1, \ldots, \mathcal{D}_n \dashrightarrow \mathcal{E}$).

There is no need to worry about relational diagrams when we consider the usual syllogisms or transitive inference, where a conclusion diagram consists of only two circles, and it is already relational. We require the restriction when we consider a more general diagram as a conclusion, in which only a particular existential point makes the given inference invalid. See [11].

References

1 Jon Barwise and John Etchemendy, *Tarski's world*. Stanford, CSLI, 1993.
2 Jon Barwise and John Etchemendy, Logic, Proof, and Reasoning, manuscript.
3 Anthony G. Cohn and Shyamanta M. Hazarika, Qualitative Spatial Representation and Reasoning: An Overview, *Fundamenta Informaticae*, 46 (1-2), 1-29, 2001.
4 Anthony G. Cohn and Jochen Renz, Qualitative Spatial Representation and Reasoning, in: F. van Hermelen, V. Lifschitz, B. Porter, eds., *Handbook of Knowledge Representation*, Elsevier, 551-596, 2008.
5 John Howse, Gem Stapleton, and John Taylor, Spider Diagrams, *LMS Journal of Computation and Mathematics*, Volume 8, 145-194, London Mathematical Society, 2005.
6 Koji Mineshima, Mitsuhiro Okada, and Ryo Takemura, A Diagrammatic Inference System with Euler Circles, *Journal of Logic, Language and Information*, 21, 3, 365-391, 2012.
7 Sun-Joo Shin, *The Logical Status of Diagrams*, Cambridge University Press, 1994.
8 David A. Randell, Zhan Cui, Anthony G. Cohn, A spatial logic based on regions and connection, *Proceedings of 3rd International Conference on Knowledge Representation and Reasoning*, Morgan Kaufmann, San Mateo, 165-176, 1992.
9 Ryo Takemura, Completeness of an Euler Diagrammatic System with Constant and Existential Points, *Journal of Humanities and Sciences Nihon University*, College of commerce Nihon University, Vol.19, No. 1-2, 23-40, 2013.

10 Ryo Takemura, Proof theory for reasoning with Euler diagrams: a logic translation and normalization, *Studia Logica*, 101, 1, 157-191, 2013.

11 Ryo Takemura, Counter-Example Construction with Euler Diagrams, *Studia Logica*, to appear. A preliminary version is available at:
http://abelard.flet.keio.ac.jp/person/takemura/index.html

TryLogic tutorial: An approach to learning Logic by proving and refuting

Patrick Terrematte[1] **and João Marcos**[2]

1 **Federal University of Rio Grande do Norte (UFRN)**
Department of Informatics and Applied Mathematics (DIMAp)
Group for Logic, Language, Information, Theory and Applications (LoLITA)
Natal – RN – Brazil
jmarcos@dimap.ufrn.br

2 **Federal University of Rio Grande do Norte (UFRN)**
Digital Metropolis Institute (IMD)
Group for Logic, Language, Information, Theory and Applications (LoLITA)
Natal – RN – Brazil
patrickt@imd.ufrn.br

Abstract

Aiming to offer a framework for blended learning to the teaching of proof theory, the present paper describes an interactive tutorial, called TryLogic, teaching how to solve logical conjectures either by proofs or refutations. The paper also describes the integration of our infrastructure with the Virtual Learning Environment `Moodle` through the IMS Learning Tools Interoperability specification, and evaluates the tool we have developed.

1998 ACM Subject Classification K.3.1 Computer Uses in Education; F.4.1 Mathematical Logic

Keywords and phrases Teaching Logic; Didactic Software; Proof Theory; Refutations.

1 Introduction

> *"(...) Knowledge never hurts — what hurts is helplessness, the futility of banging your head against a brick wall without finding either proof or disproof. I have often spent weeks trying to prove a false statement — and when I learned that it is false, I felt victorious. Progress was made, knowledge was acquired, one more step toward the truth was taken."*
> — Paul R. Halmos
> *I Want to be a Mathematician: An Automathography* (1985) p. 91.

Logic permeates computing and provides essential tools for dealing with data structures. The curricula of discrete mathematics and logic courses emphasize a kind of verificationist approach through the teaching of techniques that focus on proving, instead of approaches that encourage also disproving, or refuting by way of counter-examples. The main afteraffect of such approach is a common misunderstanding by the students of the boundaries between a deductive framework and a semantic refutation.

Logic is essential, in particular, for verifying correctness of code. In checking that an iteration behaves as expected, for instance, one has to: (i) carefully choose an appropriate boolean condition to test, (ii) check whether the given initial conditions of an iteration imply the required postcondition, and usually also (iii) prove that the execution of the corresponding code terminates. In case the implication mentioned in step (ii) is refuted, then the iteration code does not implement the desired specifications for the postcondition. Conversely, if it is shown that the preconditions do imply the postcondition, then the iteration does satisfy the desired specification. The heuristic used in the analysis of this sort of situation involves

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 233–242


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

basically the challenge of finding out whether certain conjectures can be proved or refuted. The fact that the two latter tasks, proving and refuting, are complementary is a very practical application of the soundness and completeness metatheorems.

The aim of the present paper is to present a tool for teaching the use of logical reasoning to verify conjectures for which it has not previously been determined whether they are provable or refutable. One of the main goals of our tool is to teach how to construct a fully justified counter-example to witness the falsity of a given conjecture. This tool implements some of the teaching principles discussed in [5].

In the current state-of-the-art, an approach not unlike ours is used in Bornat's [2], where Logic (formal deductive proof, formal semantic disproof and program specification) is presented with the help of the proof assistant `J∀p∃`. Many other existing tools also combine the teaching of Proof Theory with Formal Semantics, e.g. `AproS Project`, `Panda`, `Tarski's World`, `Fitch` and `Boole`. On top of those methodologies and tools, our contribution is to track learning beyond the mere use of proof strategies. Continuing the work presented by Terrematte *et al.* [7], here we present an interactive tutorial to guide the student through the process of learning by trial and error: the TryLogic[1].

2 The TryLogic tutorial for proving and refuting

> *"(...) mathematics is not a deductive science. When you try to prove a theorem, you do not list the hypotheses, and then start to reason. What you do is trial and error, experimentation, guesswork. You want to find out what the facts are, and what you do is in that respect similar to what a laboratory technician does, but it is different in its degree of precision and information."*
> — Paul R. Halmos
> *I Want to be a Mathematician: An Automathography* (1985) p. 321.

Logic courses represent a pedagogical challenge and the recorded number of failures and discontinuities in them is often high. On that track, the main goal of TryLogic is to diminish the cognitive overload through a step-by-step tutorial, presenting different topics of logic related to the process of proving or refuting logical conjectures. Our tool TryLogic aims to:

- present a set of lessons in Propositional Logic that exemplify the task of proving in natural deduction (theory N_p) and refuting in a formal semantics (theory $\mathtt{Sem}_p$) through `Coq`;
- organize Logic in an interactive way and provide self-evaluation tasks to students;
- provide the teacher with a follow-up activity report on the lessons completed by the student at `ProofWeb`, and provide a multi-language infrastructure for human-machine interaction.

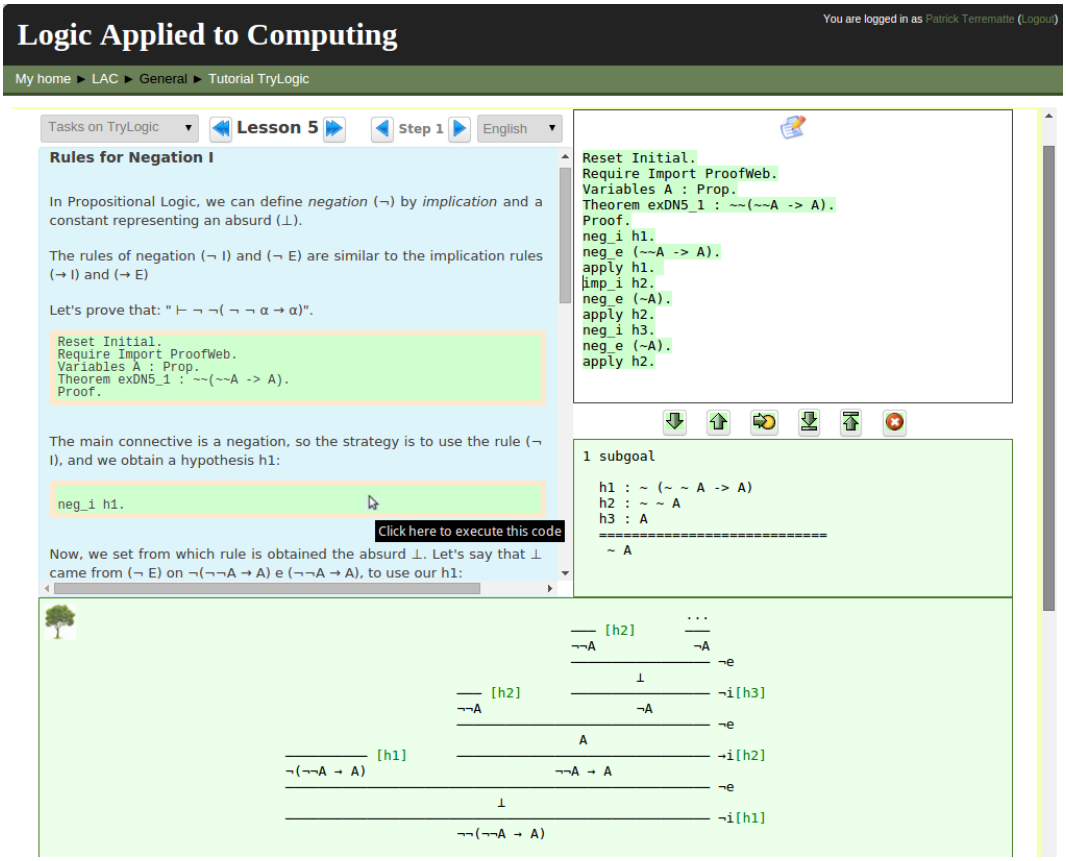
The framework is implemented by combining the following tools:

- `ProofWeb`[2]: an open source software for teaching Natural Deduction which provides interaction between some proof assistants (`Coq`[3],`Isabelle`, `Lego`) and a Web interface [4].
- `Conjectures Generator`[4]: a tool for task generation of a set of conjectural arguments (i.e. a set of premises with a formula that may follow or not from these premises). This tool was developed by our group.

[1] Available at `http://lolita.dimap.ufrn.br/trylogic`.
[2] Available at `http://prover.cs.ru.nl/`.
[3] Available at `https://coq.inria.fr/`
[4] Available at `http://lolita.dimap.ufrn.br/logicamente-ge/` and it is open source code is available at `http://github.com/terrematte/logic-propgenerator`.

- `TryOCaml`[5]: an infrastructure consisting of an interactive tutorial for teaching and interaction with the functional programming language OCaml.
- `Moodle`: a well-known Virtual Learning Environment (VLE) that helps in organizing contents and educational activities.
- `IMS Basic Learning Tools Interoperability`[6] (IMS LTI): a specification for the implementation and integration of educational tools.

Figure 1 Lessons on TryLogic integrated to `Moodle`



The TryLogic was developed as branch of the `TryOCaml` project, i.e., all lessons and interaction follow the same architecture of the latter. We implemented the $\texttt{Sem}_p$ theory in `Coq` and integrated the whole system with `ProofWeb` and `Moodle`. With a goal of centralizing the use of our tools, we have used the specification IMS LTI, which is, according to the survey [1], one of the most representative alternatives to infrastructure integration between teaching platforms. Using this specification, any collaborator who wishes to use TryLogic in any VLE can add it as an external tool. This way, it is not necessary to install TryLogic nor is it necessary to obtain special access permission for the server in which it is being installed.

[5] Available at `http://try.ocamlpro.com/`
[6] The IMS LTI was developed in 2006 by IMS Global Learning Consortium and is available at `http://www.imsglobal.org/lti/index.html`.

2.1 The Conjectures Generator

The `Conjectures Generator` for Propositional Logic was implemented through a formula generator with the SAT-solver `Limboole`[7] to evaluate propositional formulae.

The `Conjectures Generator` creates conjectures in the format of individualized tasks for `Coq`, directly in each student's area in `ProofWeb`. The students receive each task in a template for them to try and prove (showing that $\Gamma \vdash \alpha$) in the N_p theory and another one for them to try and refute (showing that $\Gamma \nvdash \alpha$) in $\texttt{Sem}_p$ theory. Of course, soundness and completeness connecting the two theories guarantee that only one of these tasks can be fulfilled.

The `Conjectures Generator` was implemented with requirements that allow one to establish some connections of relevance between the premises and the conclusion, namely: that both the conclusion and the conjunction of the premises should be contingent, and that each formula of the premises must share at least one atom with the conclusion. Other settings are available, e.g. choosing the number of conjectures, number of premises, number of distinct atoms, selecting the connectives and a range for the complexity of the formulae; also, the user may decide if in the generated conjectures all premises are necessary to prove the conclusion, and if the collection of generated conjectures are all provable, all refutable, or are evenly divided into provable and refutable, to be randomly assigned to the students.

Through the available settings, the `Conjectures Generator` is a useful tool for the teacher who wishes to evaluate propositional arguments through truth-tables, tableaux, natural deduction, resolution methods and even produce tasks concerning the evaluation of arguments.

3 Propositional logic for proving and refuting

> *"(...) Every genuine test of a theory is an attempt to falsify it, or refute it."*
> — Karl Raimund Popper
> *Conjectures and Refutations: The Growth of Scientific Knowledge* (1963) p. 36.

To **prove** a conjecture $\Gamma \vdash \alpha$ in propositional logic with Natural Deduction it is necessary to build a derivation tree to witness it. Our students were taught to do this using the rules of a theory we call N_p, which is essentially the same that is natively implemented on `ProofWeb`[8], following the usual style of natural deduction introduced by Gerhard Gentzen in 1935. As an illustration, the rules for disjunction are the following:

$$\frac{\beta}{\alpha \vee \beta}\ (\vee I_0) \qquad \frac{\alpha}{\alpha \vee \beta}\ (\vee I_1) \qquad \frac{\alpha \vee \beta \quad \begin{array}{c}\overline{\alpha}^{\,m}\\ \vdots \\ \gamma\end{array} \quad \begin{array}{c}\overline{\beta}^{\,n}\\ \vdots \\ \gamma\end{array}}{\gamma}\ (\vee E){:}m,n$$

In constrast, in an approach involving formal semantics, we build a refutation tree by using the notions of valuation and satisfaction. A valuation v maps formulae to truth-values. An argument is refutable ($\Gamma \nvdash \alpha$) if there is a valuation v that satisfies all formulae in Γ and simultaneously falsifies α ($v \Vdash \Gamma$ and $v \nVdash \alpha$). To **refute** conjectures the students need to build refutation trees on the $\texttt{Sem}_p$ theory. The rules of $\texttt{Sem}_p$ compositionally manipulate satisfaction of formulae by a valuation v, and they are the following:

---

[7] Available at `http://fmv.jku.at/limboole/`.
[8] Check the `ProofWeb`'s manual at `https://prover.cs.ru.nl/man.pdf`

$$\frac{v \nVdash \alpha}{v \Vdash \neg\alpha}\ (\neg\ T) \qquad \frac{v \Vdash \alpha}{v \nVdash \neg\alpha}\ (\neg\ F) \qquad \frac{}{v \Vdash \top}\ (\top) \qquad \frac{}{v \nVdash \bot}\ (\bot)$$

$$\frac{v \Vdash \alpha \quad v \Vdash \beta}{v \Vdash \alpha \wedge \beta}\ (\wedge\ T) \qquad \frac{v \nVdash \alpha}{v \nVdash \alpha \wedge \beta}\ (\wedge\ F1) \qquad \frac{v \nVdash \beta}{v \nVdash \alpha \wedge \beta}\ (\wedge\ F2)$$

$$\frac{v \Vdash \alpha}{v \Vdash \alpha \vee \beta}\ (\vee\ T1) \qquad \frac{v \Vdash \beta}{v \Vdash \alpha \vee \beta}\ (\vee\ T2) \qquad \frac{v \nVdash \alpha \quad v \nVdash \beta}{v \nVdash \alpha \vee \beta}\ (\vee\ F)$$

$$\frac{v \nVdash \alpha}{v \Vdash \alpha \to \beta}\ (\to\ T1) \qquad \frac{v \Vdash \beta}{v \Vdash \alpha \to \beta}\ (\to\ T2) \qquad \frac{v \Vdash \alpha \quad v \nVdash \beta}{v \nVdash \alpha \to \beta}\ (\to\ F)$$

$$\frac{v \Vdash \alpha \quad v \Vdash \beta}{v \Vdash \alpha \leftrightarrow \beta}\ (\leftrightarrow\ T1) \qquad \frac{v \nVdash \alpha \quad v \nVdash \beta}{v \Vdash \alpha \leftrightarrow \beta}\ (\leftrightarrow\ T1)$$

$$\frac{v \Vdash \alpha \quad v \nVdash \beta}{v \nVdash \alpha \leftrightarrow \beta}\ (\leftrightarrow\ F1) \qquad \frac{v \nVdash \alpha \quad v \Vdash \beta}{v \nVdash \alpha \leftrightarrow \beta}\ (\leftrightarrow\ F2)$$

With these rules, the $\mathtt{Sem}_p$ theory allows us to show that a given sentence is not a semantic consequence of a given set of premises. Here is a full example of a refutation tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{v \Vdash \mathsf{p}}}{v \Vdash \mathsf{r} \to \mathsf{p}}\ (\to T2)}{v \Vdash \mathsf{q} \vee (\mathsf{r} \to \mathsf{p})}\ (\vee T2) \qquad \dfrac{\dfrac{\dfrac{}{v \Vdash \mathsf{p}}\ (\neg F) \quad \dfrac{}{v \Vdash \mathsf{r}}\ (\neg F)}{v \nVdash \neg\mathsf{p} \vee \neg\mathsf{r}}\ (\vee F)}{v \Vdash (\neg\mathsf{p} \vee \neg\mathsf{r}) \to \mathsf{q}}\ (\to T1) \qquad \dfrac{\dfrac{\dfrac{}{v \Vdash \mathsf{p}} \quad \dfrac{}{v \nVdash \mathsf{q}}\ (\leftrightarrow F)}{v \nVdash \mathsf{p} \leftrightarrow \mathsf{q}}\ (\neg T) \qquad \dfrac{\dfrac{}{v \Vdash \mathsf{r}}\ (\neg F)}{v \nVdash \neg\mathsf{r} \wedge \mathsf{p}}\ (\wedge F1)}{v \Vdash \neg(\mathsf{p} \leftrightarrow \mathsf{q})}\ (\to F)}{v \nVdash \neg(\mathsf{p} \leftrightarrow \mathsf{q}) \to (\neg\mathsf{r} \wedge \mathsf{p})}$$

$$\mathsf{q} \vee (\mathsf{r} \to \mathsf{p}), (\neg\mathsf{p} \vee \neg\mathsf{r}) \to \mathsf{q} \not\models \neg(\mathsf{p} \leftrightarrow \mathsf{q}) \to (\neg\mathsf{r} \wedge \mathsf{p})\ (by \models)$$

In a bottom-up reading each connective in $\mathtt{Sem}_p$ has rules that provide a sufficient condition for a valuation v to satisfy (or falsify) a given sentence. On the other hand, in a top-down reading, the application of the rules represent a semantic inference. In the branches of the refutation tree one finds statements in the form $v \Vdash \alpha$ or $v \nVdash \alpha$. In the leaves, one finds statements such as $v \Vdash \mathsf{p}$ or $v \nVdash \mathsf{p}$, where p is an atomic formula. A refutation tree represents thus a fully justified counter-model to a given conjecture. Note that the rules are analytical, i.e. the premises of each rule contains statements over subformulas of the formula in the conclusion of the rule. This ultimately means that the leaves of an exhausted tree are always over atomic formulae.

The general backward strategy for building a refutation tree follows these steps:

1. Assume that the conjecture is refutable, i.e. that there is a valuation v that satisfies its premises and falsifies its purported conclusion.
2. Exhaustively explore and justify step-by-step with the $\mathtt{Sem}_p$ theory the consequences of the initial assumption that the conjecture is refutable. The exhaustive exploration means it may be necessary to backtrack to try other possible rules.
3. Check if the valuation consistently satisfies or falsifies each propositional atom involved, i.e. it cannot be that a valuation v satisfies some atom p ($v \Vdash \mathsf{p}$) and falsifies the same atom p ($v \nVdash \mathsf{p}$) in the same refutation tree.

Note that the above strategy only applies to refuting contingent or contradictory formulae, and to exhibiting witnesses to invalid arguments. If one does not manage to build a refutation tree after an exhaustive exploration of the possibilities, this means that there does not exist a valuation v such $v \Vdash \Gamma$ and $v \nVdash \alpha$. Therefore, by the relation of semantic consequence we know that $\Gamma \vDash \alpha$. Thus, from the completeness theorem ($\Gamma \vDash \alpha \Rightarrow \Gamma \vdash \alpha$), one knows that the conjecture can be proved, i.e. that it is possible to build a derivation tree in the N_p theory.

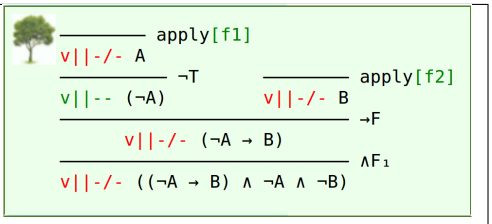
3.1 Some logical and pedagogical remarks

Each connective rule of the $\texttt{Sem}_p$ theory is implemented by tacticals in `Coq` and we extended the `ProofWeb` system to display the corresponding refutation trees, as illustrated in Fig. 2.

Figure 2 *Script* for refutation on `ProofWeb`

```
Reset Initial.
Require Import Semantics.
Parameter A B C D : Prop.
Hypothesis f1 : (v ||-/- A).
Hypothesis f2 : (v ||-/- B).
Theorem sem_ex2 :
(v ||-/- ((~A->B) /\ (~A/\~B))).
Proof.
conjF1.
impF.
negT.
apply f1.
apply f2.
```

a) *Script of refuting tacticals*

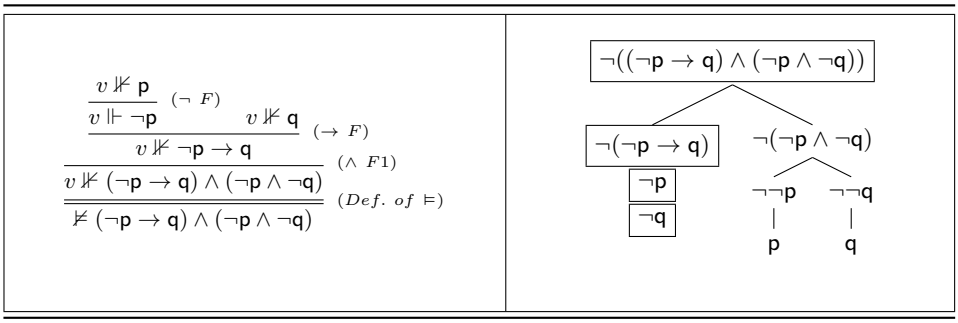


b) *Resulting tree on* TryLogic/`ProofWeb`

One of the pedagogical advantages of `ProofWeb` is its coherence with the *modality effect* of Cognitive Load Theory [6, p.129]. The idea is that two well connected sources of information reinforce the organizing process and facilitate the transfer of information to the long-term memory information store. The multiple representation is applied in Fig. 2, where we can observe that without the proof script being inserted on the left-hand side (a) it would not be possible to check the object on the right-hand side (b) for the tactical sequence represents a justification for the refutation produced. The refutations are not static (b), but in fact, correspond to the dynamic linear process of their construction on side (a). Thus, the visualization of (b) has didactic value as it is also useful to the communicability of the refutation structure in (a).

The heuristic procedure for refutation presented here might be replaced by other deductive formalisms in Propositional Logic, such as the sequent calculus, resolution, tableaux or even truth-tables. However, we avoid the truth-table method for its fixed exponential computational cost (2^n, where n is the number of distinct atoms in the conjecture) and its purely algorithmic character, which we judge not to have optimal pedagogical value. Tableaux, on the other hand, are often very efficient in both tasks of proving and refuting. However, they also make the procedure fully automatic. While this might be a desirable property from a computational viewpoint, from the didactic perspective we claim that tableaux create a conceptually undesirable overlap between deductive formalism and formal semantics. As a consequence of the exclusive use of tableaux, students are often led to build no appreciation at all for the distinction between Proof Theory and Formal Semantics. To clarify the meaning of our semantic heuristics, check the comparison between the refutation tree and the tableau method in Fig. 3: Note in particular that in using $\texttt{Sem}_p$ the students are forced all the time to take decisions about which tableau branch they should want to explore. It is known that in the worst-case scenarios tableaux might be much more costly than truth-tables [3, p.62]. A tableau is exhausted only when all the branches are fully

Figure 3 Refutation tree *versus* Tableaux Method



explored, and this may depend on the ratio between the complexity of the formulae and the atoms that occur in them. Therefore, if a formula has higher complexity than the number of distinct propositional atoms, then the tableau analysis may be longer than the number of rows in the truth table. In contrast, our heuristic procedure is not fully automatic and the wise choice of which path to follow may introduce exponential speed-up. Ultimately, the use of $\texttt{Sem}_p$ simply requires the production of a sequence of formulae corresponding to an open branch of a tableau tree.

Our goal is to improve the logical intuition of students. Therefore, students are told that in cases where the semantic heuristics do not allow for a refutation, they should look for a derivation tree in N_p. On the other hand, when they are having trouble in proving, they might well try to refute the selected conjecture.

3.2 Analysis of experiments with TryLogic

We performed several experiments in the years of 2012, 2013 and 2014 to evaluate TryLogic in blended learning use. The task of proving or refuting was given to the average of 15 students per semester of Computer Science in the upper undergraduate course of *Logic Applied to Computing* at the Federal University of Rio Grande do Norte (DIMAp/UFRN), of which an average of 58% have actively used the system. Through face-to-face classes we taught only using theoretical fundamentals, and our biggest challenge was teaching the computer-assisted task of proving or refuting exclusively through TryLogic. The main task given to students was to prove or refute six to eight conjectures randomly assigned. These consisted in two conjectures per each of three or four levels of difficulty. For instance at first level (easiest), the conjectures have 3 distinct propositional atoms, with 3 premises and a complexity between 2 and 4 connectives per formula. The fourth and hardest level has 6 distinct propositional atoms, with 4 premises and a complexity of 4 to 6 of connectives. The learning goals are to practice formal proof and refutation heuristics, as well as to advance the understanding of soundness and completeness metatheorems. At the end of each experiment, students answered a questionnaire about their profile, their use of the available tools, their difficulties in solving the tasks and their theoretical understanding of the tasks.

Some general conclusions about the experiments are:

- TryLogic provides the understanding of the deductive process in `Coq` to students who had brief theoretical contact with the content of Natural Deduction.
- The students consistently solved more refutable conjectures than provable ones, even if they have received in average an equal number of each kind of conjecture. For instance in Spring 2013, out of 60 solved conjectures, the students presented 43 refutations. It is

possible to draw at least the following two interpretations for this phenomenon: that the search for a refutation tree is easier than the production of a natural deduction proof, or that the lessons for proving in Natural Deduction on TryLogic need to have an improved teaching strategy. The first interpretation is coherent with our learning goals, we aim to show that refutation is natural and necessary in Logic. As for the second interpretation, we feel it important to add that some conjectures given as task are really large and difficult to prove[9] and this might explain the smaller number of produced formal proofs.

- A negative conclusion drawn from the questionnaire was that the practice involved on prove or refute, does not necessary imply the theoretical understanding of the metatheorems of completeness and soundness.
- Using the theories N_p and $\texttt{Sem}_p$, implemented in `Coq`, the student applies a heuristic for proving and refuting through justified and verified steps. This way, with TryLogic, the experimental process of 'trial and error' is taught in a guided environment.

4 Future Works and Final Remarks

> *"(...) we teach mathematics to the engineers, physicists, biologists, psychologists, economists — and mathematicians — of the future. (...) It is not enough to teach them everything that's known—they must know also how to find out what has not yet been found."*
> — Paul R. Halmos
> *I Want to be a Mathematician: An Automathography* (1985) p. 322.

This paper presents an infrastructure of integrated tools for the teaching of Logic with focus on: (i) an organized step-by-step presentation of the content of Natural Deduction and Propositional Semantics in a sequential and interactive way; (ii) providing the student with interactive self-evaluation tasks; (iii) the interaction with the `Conjectures Generator` and TryLogic with `Moodle` through IMS LTI. It is worth noting that, since the TryLogic is based on `ProofWeb` and the lessons are structured on `TryOCaml`, our infrastructure is extensible and customizable[10] to build lessons on any other formal theory implemented on `Coq` or `Isabelle`, e.g. on Modal Logic, Number Theory, Set Theory, or Hoare Logic. Our contributions to teaching Logic are part of an initiative that needs to be enhanced. Some opportunities for the extension of the project would include:

- Producing lessons in English, Spanish, French and other languages.
- Implementing in the `Conjectures Generator` metrics of difficulty for derivations given by the size of normalized proofs and the number of uses of certain rules in the latter proofs.
- Extending the `Conjectures Generator` and the theory $\texttt{Sem}_p$ to First Order Logic and producing new tasks and lessons of First Order Logic through TryLogic.

Acknowledgements The authors acknowledge partial support by the Marie Curie project PIRSES-GA-2012-318986, funded by EU-FP7, and by CNPq / Brazil. The authors also want to thank all undergraduate students of Computer Science and Computer Engineering who have contributed to the project during several semesters of the course of Logic Applied to Computing at DIMAp/UFRN. For the implementation of the `Conjectures Generator`, a special acknowledgement should go to Elias Amaral.

[9] For an example, a conjecture generated in the fourth level to be proved in Natural Deduction was this one:
$\{\neg(\mathsf{p}\vee((\mathsf{q}\to(\mathsf{u}\leftrightarrow\mathsf{r}))\wedge(\mathsf{s}\leftrightarrow\mathsf{r}))),\mathsf{t}\to(\neg(\mathsf{r}\vee(\mathsf{r}\vee(\mathsf{p}\leftrightarrow\mathsf{p})))),\neg((\neg((\mathsf{p}\leftrightarrow(\neg\mathsf{t}))\leftrightarrow\mathsf{u}))\vee\mathsf{p}),((((\mathsf{r}\wedge\mathsf{s})\wedge\mathsf{u})\to(\mathsf{s}\vee\mathsf{p}))\leftrightarrow\mathsf{u})\wedge\mathsf{s}\}\vdash(\mathsf{u}\to(\mathsf{u}\wedge(\mathsf{t}\to(\mathsf{s}\to\mathsf{q}))))\vee(\mathsf{r}\leftrightarrow\mathsf{p})$.

[10] The project can be forked from: `https://github.com/terrematte/trylogic`.

References

1 Carlos Alario-Hoyos and Scott Wilson. Comparison of the main alternatives to the integration of external tools in different platforms. In *Proceedings of the International Conference of Education, Research and Innovation, ICERI*, pages 3466–3476, 2010.

2 Richard Bornat. *Proof and Disproof in formal logic: an introduction for programmers.* Oxford University Press, 2005.

3 Marcello D'Agostino. *Investigations into the complexity of some propositional calculi.* PhD thesis, University of Oxford, 1992.

4 Maxim Hendriks, Cezary Kaliszyk, Femke Van Raamsdonk, and Freek Wiedijk. Teaching logic using a state-of-the-art proof-assistant. *Acta Didactica Napocensia*, 3(2):35–48, 2010.

5 João Marcos. Fail better: What formalized math can teach us about learning. 4th International Conference on Tools for Teaching Logic, pages 119–128.

6 John Sweller, Paul Ayres, and S. Kalyuga. *Cognitive Load Theory.* Explorations in the Learning Sciences, Instructional Systems and Performance Technologies. Springer, 2011.

7 Patrick Terrematte, Fabrício Costa, and João Marcos. Logicamente: A Virtual Learning Environment for logic based on Learning Objects. In Patrick Blackburn, Hans Ditmarsch, María Manzano, and Fernando Soler-Toscano, editors, *Third International Conference Tools for Teaching Logic.*, volume 6680 of *Lecture Notes in Artificial Intelligence*, pages 223–230. Springer-Verlag, Berlin, 2011.

To Teach Modal Logic: An Opinionated Survey *

Luis A. Urtubey

Universidad Nacional de Córdoba
Córdoba, Argentina
urtubey@ffyh.unc.edu.ar

Abstract

I aim to promote an alternative agenda for teaching modal logic chiefly inspired by the relationships between modal logic and philosophy. The guiding idea for this proposal is a reappraisal of the interest of modal logic in philosophy, which do not stem mainly from mathematical issues, but which is motivated by central problems of philosophy and language. I will point out some themes to start elaborating a guide for a more comprehensive approach to teach modal logic, and consider the contributions of dual-process theories in cognitive science, in order to explore a pedagogical framework for the proposed point of view.

Keywords and phrases Modal Logic – Philosophy – Teaching Modal Reasoning –

1 Introduction

In his recent book *Logic and how it gets that way*, Dale Jaquette [5] quotes a few lines of Nicholas Rescher's *Autobiography*, where he talks about his conception of logic. Rescher's words read as follows:

> I never looked a logic as an accomplished fact –a field fully formed and merely in need of systematization and exposition, but saw it as an unfinished and malleable discipline to be reworked and structured for the sake of its applications. I always gravitated towards those logical issues and areas that were of potential utility for the elucidation and treatment of philosophical issues, and viewed logic not just as a self-contained discipline, but as a body of machinery for the accomplishment of extra-logical work. [16]

I was struck by this quote of Rescher. Admittedly, it contains some insightful observations. Firstly, Rescher points out that logic can be seen as an unfinished discipline, which can be structured according to its applications. A second interesting point comes from the observation that philosopher's interest in logic is triggered by those issues which have at least potential utility for philosophical problems. Last but not least, logic is chiefly viewed by Rescher as a tool for accomplishing extra-logical work. This last remark is relevant with respect to teaching logic, if we include it among those extra-logical activities.

In this short paper, I will be concerned with these remarks from the perspective of modal logic in particular. It is clear that modal logic has been articulated by considering its applications, as it is witnessed by the alternative interpretations of the modal operators of necessity and possibility. Thus, we are accustomed to talk nowadays about logical, metaphysical, deontic, and epistemic modalities, among many other options. Moreover, it is also easy to see that modal logic is central to the so-called Philosophical Logic, wherein those logical issues of philosophical concern are collected. Viewing modal logic as a tool for

* This work was partially supported by the project "Intentional Hybrid Logic" (MICINN FFI2013-47126-P); Department of Philosophy, Logic and Aesthetics; University of Salamanca (Spain). It has received also partial support from SeCyT (UNC-Argentina).

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 243–252

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the accomplishment of extra-logical work is less frequent, but it is not totally unusual, as it is witnessed, for example, by certain applications in artificial intelligence. Our primary concern will be, in this respect the potential use of modal logic to improve people's reasoning abilities, by furnishing them with formal tools and normative standards. It has been said, however, that formal logic failed in its attempt to accomplish this objective. Consequently, modal logic could not reach this honour either. Contrariwise, I will argue that assuming and elaborating certain perspective with respect to cognition and reasoning, modal logic can still be fruitfully applied to improve people's reasoning abilities. Notwithstanding, it should not be misinterpreted. I will not offer here any specific details about the concrete organization of a modal logic course following this approach. I am only giving an overview, which can be exploited and implemented in different ways.

2 Modal Logic: Philosophical Issues

I am going to take a look now at some themes which populated the relationships between modal logic and philosophy during the twentieth century. I think that these examples can serve to articulate a bunch of topics for teaching modal logic. Admittedly, this is not an exhaustive enumeration, but it will suffice to give an idea of the approach we have in mind. I will articulate these issues in a pedagogical approach towards the end of the paper. A very useful summary to start with is given by the comprehensive article by Linsdström and Segerber in *The Handbook of Modal Logic* [9], where they point out some themes concerning the relationship between modal logic and philosophy that are also relevant from our point of view. They enumerates the followings:

1. The back-and-forth between philosophy and modal logic: good examples are Carnap and Prior among many others, who tried to use modal logic to throw light on old philosophical questions.
2. The interpretation problem. The problem of providing a certain modal logic with an intuitive interpretation, which should not be conflated with the problem of providing a formal system with a model-theoretic semantics.
3. The fact that one may compare this situation with that in probability theory, where definitions of concepts like 'outcome space' and 'random variable' are orthogonal to questions about interpretations of the concept of probability.
4. The value of formalization. Modal logic sets standards of precision, which are a challenge to —-and sometimes a model for-— philosophy. Classical philosophical questions can be sharpened and seen from a new perspective when formulated in the framework of modal logic. On the other hand, representing old questions in a formal garb has its dangers, such as simplification and distortion.
5. Why modal logic rather than classical (first or higher order) logic? The idioms of modal logic seem better to correspond to human ways of thinking than ordinary extensional logic. One can think, for example about counterfactual reasoning and its central place in human thinking.

Let's go a little bit more along this path.

2.1 The Readings of Modality

As it was said above, Carnap and Prior constitute two important philosophical landmarks in the development of modal logic. Thence, it is worthwhile to isolate the philosophical approach to modalities of these authors and to introduce thereby other themes which are related with their work. Lindström and Segerberg observe that Carnap's project was not

only to develop a semantics (in the sense of Tarski) for intensional languages, but also to use metalinguistic notions from formal semantics to throw light on the modal ones. In "Modalities and quantification" from 1946, he writes:

> It seems to me ... that it is not possible to construct a satisfactory system before the meaning of the modalities are sufficiently clarified. I further believe that this clarification can best be achieved by correlating each of the modal concepts with a corresponding semantical concept (for example, necessity with L-truth).

Concerning Prior, years ago, in a remarkably thoughtful article [10], Christopher Menzel analyzed Prior's approach to modal logic and his deep philosophical concerns. Menzel points out that, for Arthur Prior, "the construction of a logic was a supremely philosophical task". Being a clever logician, Prior joined the virtue of appreciating "a finely crafted formal system for its own sake", apart of its "meaning" or philosophical significance, as well as the power of a genuine logic "to lays bare the nature of those concepts that it purports to be a logic of". This last task can be accomplished only by way of deep reflection and insightful philosophical analysis. Menzel remarks that Prior's search for the "true" modal logic is where this sort of reflection and analysis is more evident[1]. Menzel's paper in its turn is a nice guide "to trace the course of Prior's own struggles with the concepts and phenomena of modality, and the reasoning that led him to his own rather peculiar modal logic Q". It is also an advisable starting point to appreciate Prior's intuitions and the arguments that rest upon them.

Let us also consider Prior's work under a new light. In a recent article about Arthur Prior's work on hybrid logic [2], Patrick Blackburn has considered extensively those aspects of Prior's philosophy which motivated his interest on hybrid logic. This kind of logic departs from traditional logical categories, allowing that formulas can be used as terms. Blackburn's article is also an excellent starting point to stimulate the reflection on philosophical questions, which have motivated the development of modal languages. Blackburn points out that Prior's development of hybrid logic was fuelled by "his most fundamental convictions concerning time and tense, namely that the modal perspective, and in particular the internal view offered by modal logic, was the way to capture genuinely temporal logic". In the early years of the last Century, the Hegelian philosopher John McTaggard posed a tricky argument concerning the unreality of time. McTaggard's argument is based on the postulation of two alternative series, which respectively constitute two ways of conceptualizing time: the A series and B series conceptions. In his work *Past, Present and Future*, Prior discusses McTaggard argument. The A series considers the flow of time from past, through present, to future. It is an internal and situated conception of time, which reflects the experience of temporality of human beings.

Under the B series, time is a collection of instants ordered by a relation of temporal precedence, i.e. positions are ordered from earlier-than to later-than relations. Particularly, Prior favours A series talk over B series talk, because he thought that the external perspective underlying B series did not reflect the way in which human beings experience time. This motivate, in its turn the development of hybrid logic, since Prior believed that hybrid logic was the A series logic underpinning everyday tensed logic. There are too many connections between Prior's temporal logic and philosophical problems raised by Blackburn's article. I cannot survey all of them here. Nonetheless, the following quotations from [2] will be useful to illustrate an important point concerning the changes experimented by the relationship between logic and philosophy through the years:

[1] Particularly Menzel considers system Q from Prior's "Tense Logic for Non-Permanent Existents"[11]

> It is important to realize that Prior's willingness, so clearly in evidence here, to base the core of a philosophical position at least on the differing properties of formal logical systems, contrasts sharply with the attitude of many, perhaps most, contemporary modal logicians. Nowadays logicians are rather more guarded about the direct relevance of the properties of particular formal systems to philosophy. (...) Much of this contemporary attitude stems from the model-theoretic perspective which underlies current practice.

Nonetheless, Patrick Blackburn also notices that

> (...) this does not mean that the model-theoretic conception rules out the possibility of drawing philosophical conclusions based on logical analysis.

On the contrary:

> (...) The most direct way that model-theoretically oriented logic helps philosophy is by demonstrating that there are more ontological possibilities than is apparent at first sight, or that certain possibilities can't be made to fit together coherently. The model-theoretic perspective helps induce philosophical modesty.

This also holds for Prior's work on tense and temporal logic.

Admittedly, unlike the case of mathematical modal logic, for teaching modal logic in philosophy or broadly speaking in the humanities, it seems to be crucial to count with a substantial reading of the box or the diamond. Following ideas of an unpublished paper by John P. Burgess [3], the point to be observed is that we have to admit throughout the whole history two alternative readings of the box as "it is analytic that...", which seems the official explanation of I. Lewis, and reading the box as "it could not have failed to be the case that..." (a reading which doubtless influenced the "intuitions" of some). It must also be admitted that these were two different and incompatible readings. We'll go back on this below.

2.2 Quine's Interpretational Challenge

For the second item above mentioned, it is inevitable to think of Quine's criticism of quantified modal logic[2]. According to the article of Lindström and Segerberg, this criticism comes in different strands. First, there is the simple observation that classical quantification theory with identity cannot be applied to a language in which substitutivity of identicals for singular terms fails. The so-called Morning Star Paradox shows that we must abandon either universal specification (and existential generalisation) or indiscernibility of identicals. This premise goes to the seemingly uncontroversial conclusion that classical quantification theory (with identity and individual constants) cannot be mixed up with non-extensional operators (those for which substitutivity of identicals for singular terms fail) without being modified in some way. This weak claim already gives rise to the challenge of extending quantification theory in a consistent way to languages with non-extensional operators.[3] Perspicuously Lindström and Segerberg differentiate a much stronger claim in addition to this former claim, which sometimes can be found in Quine's early works, relative to the fact that objectual quantification into non-extensional ("opaque") constructions simply does not make sense. The cause is that occurrences of variables inside of opaque constructions do not serve simply

[2] The most known references at this respect are [13], [12], [14], [15]
[3] A very comprehensive treatment is given by James Garson in [4]

to refer to their objects, and cannot therefore be bound by quantifiers outside of the opaque construction. Consequently, quantifying into opaque contexts would be like trying to quantify into quotations. As Lindström and Segerberg observe: "This claim is hardly credible in the face of the multitude of quantified intensional logics that have been developed since then, and it can be taken to be refuted by the work of Kaplan and Fine among others. I think that Lindström and Segerberg's distinction about Quine's very different thesis is a fact that must be appreciated and incorporated when teaching modal logic in order to exploit this frequent confusions.

2.3 The Nature of Modality

It will be worthwhile to return here to some reflections about modal logic recently introduced by John P. Burgess that we just referred to [3]. According to Burgess' approach, it is convenient to look backward to the origins of modal logic. Originally logic used to "name an enterprise aiming to prescribe the norms of deductive argumentation in philosophy and elsewhere". However, "the pursuit of what were the original aims of logic is now for many inseparable from philosophy generally, and philosophy of language especially.[3] Thus, logic progressively has gone far away from its original motivations. In fact, logic is no more restricted to the study of correct or valid reasoning. Formal systems are studied in its own right. Burgess' observation also enables to see that the original logic's aim is now shared by many disciplines other than logic. In consequence, in order to fulfil this original aim one must cross some disciplinary borders, since it is no longer in the domains of logic alone. Burgess also considers the case of modal logic in particular and thinks that we must go back to the work of Aristotle to appreciate the initial sense of modality that was getting lost as time went by. Moreover, a neglected question at this point is what is meant by "modal" here. And if there is a sense which is particular relevant to philosophy. What is characteristic of modality is the expression of irreality of one kind or another, as it is expressed by means of grammar. It is widely accepted the distinction between deontic, epistemic and dynamic modalities. In the beginnings of modal logic, H. von Wright also made an almost similar distinction. In fact, all of them were of interest in philosophy, but specially the third, which more or less corresponds to what is called "metaphysical" modality. This is the conclusion of Burgess, which is worth to consider in the setting of teaching modal logic in philosophy. The notions associated with this type of modality are the necessary, in the sense, of what is and would have been (could not failed to be) no matter what, and the possible in the sense of what is or isn't but potentially might (or could) have been. The idea of a true modal logic endorsed by Burgess and other philosophers must be interpreted as an effort to put modal logic again in the right way. In the words of Burgess [3]:

> Very little of modal logic is devoted to the attempt to determine what are the valid forms of argument involving necessity and possibility in any sense, and vanishingly little is devoted to the original aim of modal logic, that of dealing with the logic of modality in our default sense.

The starting point in this regard is a distinction, which admits two contrasting opinions about modality that have coexisted through the whole history of modal logic. Together with this initial idea of necessity, there would be an idea of necessity as analyticity, which is also unconsciously influenced by thoughts about necessity in the "metaphysical" sense. According to Burgess, the general pre-Kripkean failure to recognize both notions explicitly and to distinguish between them naturally made it impossible to appreciate that different logics may be appropriate to the different notions. As it happens with the precedent observations

in this paper, I believe that this thesis about the confusion between two alternatives ideas of necessity can be usefully incorporated in a strategy for teaching modal logic. Among other confusions that one can take into account regarding the application of modal logic, Burgess also considers "the modal distinction between the contingent and the necessary". Let us consider the following examples also from [3].

1. There has been no female U. S. president.
2. The number 29 has no nontrivial divisors.

Admittedly, though there has been no female U. S. president, there could have been, whereas the number 29 not only has no nontrivial divisors but couldn't have had any. Thence, it is allowed to make counterfactual speculations regarding the first proposition, but not with respect to the second. And the most important question is how we know about the difference between these propositions. This is a central problem for modal epistemology concerning the bridge between 'is' and 'could'. It is also an interesting problem to reflect upon when you teach modal logic. A nice place to start is the classic statement of the problem given by Immanuel Kant in the introduction to the second or B edition of his *Critique of Pure Reason* that Burgess points out:

> Experience teaches us that a thing is so, but not that cannot be otherwise.

That is to say that experience can teach us that a necessary truth is true, but it cannot teach us that it is necessary. Translated into the lexicon of possible world semantics it means that we can access through our senses to the actual world -the one we live in- whereas when we speak about what could or couldn't have been, we are making an assertion about how thinks are in other worlds.

2.4 A Modal Puzzle

Let us close this section by considering a well-known tricky puzzle about modality which can swell the list of teaching devices. The question concerning the right modal logic for "metaphysical modality" has already prompted the interest of philosophers and logicians. The most favoured systems have been S5 and S4, though there have been alternatives like McKinsey S4.1. Nathan Salmon and others have developed an argument for the system T, or rather against S4. As you know the characteristic law distinguishing S4 and T claims that if a proposition P is necessary, it is necessarily necessary (or if P is possibly possible), then it is possible. Salmon's argument has became to be known as the "modal paradox". Let us now describe the example which centres in the S4 axiom following the exposition of Burgess [3].

> The same principle [S4] would seem to apply to, say, a ship made of a thousand planks. There is an intuition X to the effect that the same ship could have been made of 999 of the same planks, arranged the same way, plus a replacement for plank 473; but at the same time there is an intuition Y to the effect that a ship made of a thousand different planks would have been a different ship. The puzzle is that these two intuitions, X and Y, seem to conflict.

In [3], the conflict referred above is described in Leibnizian-Kripkean possible worlds language as follows.

> Here in the actual world A we have a ship, let us name it the good ship Theseus, made of 1000 planks. Our first intuition X is that the same ship could have been made of 999 of these planks plus a replacement for plank 473. In possible-worlds

> terms that means there is another world B where the same good ship Theseus exists with all but one plank the same as in our world A, and only plank 473 different. (...) [A]nd by the same sort of intuition, there must be another world C where the same good ship Theseus exists with all (...) but with plank 692 different. That means for us back in world A there is another world C where the good ship Theseus exists with all but two planks the same as in our world A, but with planks 473 and 692 different (...). The same sort of considerations can then be used to argue that one could have three planks different, or four, or five, or all 1000. But that is contrary to our other intuition Y.

Salmon's solution rejects S4 axiom to avoid treating identity as vague. It is tantamount to recognize that though world B is a possibility for us in world A, and world C is a possibility for those in world B, still C is not a possibility for us in world A, but only a "possible possibility". That means that transitiveness must be banned in this context. There are reasons, however, as Burgess claims, to doubt of this analysis. Since the puzzle seems to be parallel to another famous temporal one already discussed by Plutarch and Thomas Hobbes, which involves the ship of Theseus. And in this last case, transitivity cannot be excluded. It is admissible to conjecture that this problem concerning the necessity of origins will have great significance and may ultimately be relevant to discuss the status of the law that possibly possible implies possible, when we have to teach about normal systems of modal logic.

3 Dual-Process Theories

Dual-process theory in cognitive psychology is closely related to the study of heuristics and biases in reasoning, as it was taken up for example in the research of Kahneman and Tversky [8]. From this point of view, heuristics have intrinsic limits. And it is claimed that heuristics inserted in the human mind by the process of evolution can produce errors, although it is also given to human beings to fix these failures. By means of different tasks, Stanovich together with other researchers have shown the conflict between heuristics and the so-called "analytic system". Precisely, the goal of these tasks is to bring about "a heuristically triggered response against a normative response generated by the analytic system" [18]. The idea of an "analytic system" is crucial: it is one of the most important contributions of Kahneman and Tversky that cognitive biases are not made by chance, since the human mind utilizes "innate rules of thumb", which are autonomous and automatics, and which cannot be totally banned by us when processing information. Moreover, Kahneman and Frederick also have remarked that "[t]he persistence of such systematic errors in the intuitions of experts implied that their intuitive judgements may be governed by fundamentally different processes than the slower, more deliberate computations they had been trained to execute" [7]. It is then possible to design certain tasks or contexts where these natural heuristics bring out responses which are not adequate, before being corrected by means of a second system that is slower, more analytic, and more thoughtful. Thence, it is one of the central intuitions of dual-process theories that the human mind is composed by two different systems, the so-called system1 (S1) and system2 (S2). These models are heavily discussed in the recent literature of cognitive science though. The collection of processes of S1 (the modules) are autonomous and cognitively closed; we are not conscious of their working and we cannot influence on them directly. These processes are experimented in our intuitions and our spontaneous reactions. The S2 processes, on the contrary, are voluntary and they are able to inhibit and to fix those automatic responses of S1. Most of the time, these two systems get on well with each other, but there are situations where the response elicited by S1 conflicts with

that issued by S2. Nevertheless, we can learn the rules and we can learn to apply them rightly by using S2 processes. According to the advocates of dual-process theories, for accomplishing a task it is necessary to identify first the bias of S1 and then to fix it with the appropriate tools of S2. Moreover, we have a tendency to utilize only S1, which in most cases gives us right answers, but the processes of this system mislead us in other cases. Fortunately, we may learn how to utilize S2 to detect these situations, where we are cheated, in order to look for the normatively adequate answer. Stanovich [17] has stressed the importance of acquiring the cognitive tools which make it possible to attain the resolution of those problems which S1 has not answered adequately. These are the kind of tools which are usually provided by logic courses. However, as it has been emphasized, this is not enough to overcome this obstacle: one has to have "the relevant logical rules and also [needs] to recognize the applicability of these rules in particular situations" [6]. Thence it must be also detected that the automatic answer provided by S1 has to be neutralized. Consequently biases' correction is more complex and at least involves two stages. This is the most important thing to be learned from these studies concerning their pedagogical application. Particularly, the complexity of this dual process serves also to explain the shouting failure of standard logic courses with respect to improve the reasoning abilities of many students. From this point of view, Guillaume Beaulac and Serge Robert [1] have pointed out that we can rescue the normative role of logic, by considering it in the light of a pedagogical reading of dual-process theories. The idea can be applied also to modal logic. To that end efforts must be made to elaborate cases which prompted the action of S1-systems that will have as normative counterparts the current systems of modal logic. In this case, S1-systems will serve to show up several drawbacks of modal reasoning and other puzzling issues which stem from reasoning tasks performed in this context. I think that the philosophical problems that have been shortly addressed in this paper, can –at least partially– fulfil the requirements to confront the intuitions given by S1-systems in this case and contribute to improve teaching modal logic under these premises.

4 Conclusion

In this paper, I aimed at integrating from a pedagogical perspective, the formal treatment of modalities with those philosophical problems which appeared in the origin of modal logic and have also contributed to its development. I have considered modal logic as a tool for treating and elaborating many problems in philosophy and other connected areas, relegating the mathematical issues. Concerning teaching, the dose of mathematical modal logic to be introduced must be estimated from this point of view and it will no more be the only motive of concern. On the contrary, it turns out to be instrumental. Specifically I have not considered the mathematics of modal logic, which I presupposed in a certain way. The central tenet may be summarized by saying that things must be turned upside down. Instead of focusing on the formal and mathematical aspects of modal logic, it is convenient to return to the origins and search firstly for the philosophical motivation behind the primeval ideas.

More controversially, the strategy for teaching modal logic that I am proposing here would not be that new, but I confess that I have never seen a full-fledged development of this approach. Certainly, one can find some exemplifications or applications of modal logic to philosophical thesis scattered in different logic texts. I have surveyed in this paper some of the most relevant and more accessible ones in order to contribute to the construction of an alternative perspective for teaching modal logic in philosophy. It may sound perhaps too much oriented to philosophy ignoring other uses of modal logic. That might be right, but I think that for teaching modal logic in philosophy it must be taken into account the

central motivations of those people who have decided to study philosophy. And the challenge for all of us involved both in modal logic and philosophy is to articulate courses, which pay attention to these particular motivations.

In this regard, our position lies close to some recent trends in the development of Mathematical Philosophy, which offers a new approach to philosophical problems chiefly supported by the contribution of formal methods. The most innovative feature of this proposal is the balanced combination of formal methods with deep philosophical questions, which trigger a productive reflection. This relationship suggests also a link between this approach and the application of dual-process theories to the extra-logical activity of teaching and learning the application of formal tools.

Acknowledgements The author gratefully thank two anonymous referees of this publication for their useful comments.

References

1 Guillaume Beaulac and Serge Robert. Théories à processus duaux et théories de l'éducation : Le cas de l'enseignement de la pensée critique et de la logique. *Les ateliers de l'éthique*, 6, 2011.
2 Patrick Blackburn. *Arthur Prior* and hybrid logic. *Synthese*, 150(3), 2006.
3 John P. Burgess. Modal logic in the modal sense of modality. Online, December 2014. `http://www.princeton.edu/~jburgess`.
4 James W. Garson. *Modal Logic for Philosophers*. Cambridge University Press, New York, 2013.
5 Dale Jaquette. *Logic and how it gets that way*. Routledge, New York, 2014.
6 Daniel Kahneman and Shane Frederick. Representativeness revisited: Attribute substitution in intuitive judgment. In Gilovich et al., editor, *Heuristics and Biases. The Psychology of Intuitive Judgment*. Cambridge University Press, New York, 2002.
7 Daniel Kahneman and Shane Frederick. A model of heuristic judgment. In K.J. Holyoak and R.G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*. Cambridge University Press, New York, 2005.
8 Daniel Kahneman, Paul Slovic, and Amos Tversky. *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, New York, 1982.
9 Sten Lindström and Krister Segerberg. Modal logic and philosophy. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, chapter 21. Elsevier, New York, 2007.
10 Chrispopher Menzel. The true modal logic. *Journal of Philosophical Logic*, 20(4), 1991.
11 Arthur N. Prior. Tense logic for non-permanent existents. In *A. Prior: Papers on Time and Tense*. Clarendon Press, Oxford, 1968.
12 Willard Van Orman Quine. Notes on existence and necessity. *The Journal of Philosophy*, 40(5), 1943.
13 Willard Van Orman Quine. The problem of interpreting modal logic. *The Journal of Symbolic Logic*, 2(12), 1947.
14 Willard Van Orman Quine. Reference and modality. In *W. V. Quine: From a Logical Point of View*. Harvard University Press, Cambridge, Mass., 1953.
15 Willard Van Orman Quine. Three grades of modal involvement. In *W. V. Quine: The Ways of Paradox and Other Essays*. Harvard University Press, Cambridge, Mass., 1966.
16 Nicholas Rescher. *Autobiography*. Ontos Verlag, Frankfurt, 2007.
17 Keith E. Stanovich. *What Intelligence Tests Miss: The Psychology of Rational Thought*. Yale University Press, New Haven, 2009.

18 Keith E. Stanovich, Maggie Toplak, and Richard F. West. The development of rational thought: A taxonomy of heuristics and biases. *Advances in Child Development and Behaviour*, 36, 2008.

NaDeA: A Natural Deduction Assistant with a Formalization in Isabelle

Jørgen Villadsen, Alexander Birch Jensen, and Anders Schlichtkrull

DTU Compute - Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, Building 324, DK-2800 Kongens Lyngby, Denmark
jovi@dtu.dk

Abstract

We present a new software tool for teaching logic based on natural deduction. Its proof system is formalized in the proof assistant Isabelle such that its definition is very precise. Soundness of the formalization has been proved in Isabelle. The tool is open source software developed in TypeScript / JavaScript and can thus be used directly in a browser without any further installation. Although developed for undergraduate computer science students who are used to study and program concrete computer code in a programming language we consider the approach relevant for a broader audience and for other proof systems as well.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Natural Deduction, Formalization, Isabelle Proof Assistant, First-Order Logic, Higher-Order Logic

1 Introduction

In this paper we present the NaDeA software tool. First, we provide the motivation and a short description. We then present the natural deduction system as it is done in a popular textbook [15] and as is it done in NaDeA by looking at its formalization in Isabelle. This illustrates the differences between the two approaches. We also present the semantics of first-order logic as formalized in Isabelle, which was used to prove the proof system of NaDeA sound. Thereafter we explain how NaDeA is used to construct a natural deduction proof. Lastly, we compare NaDeA to other natural deduction assistants and consider how NaDeA could be improved.

1.1 Motivation

We have been teaching a bachelor logic course — with logic programming — for a decade using a textbook with emphasis on tableaux and resolution [1]. We have started to use the proof assistant Isabelle [2] and refutation proofs are less preferable here. The proof system of natural deduction [3, 4, 5, 15] with the introduction and elimination rules as well as a discharge mechanism seems more suitable. The natural deduction proof system is widely known, used and studied among logicians throughout the world. However, our experience shows that many of our undergraduate computer science students struggle to understand the most difficult aspects.

This also goes for other proof systems. The formal language of logic can be hard to teach our students because they do not have a strong theoretical mathematical background. Instead, most of the students have a good understanding of concrete computer code in a programming language. The syntax used in Isabelle is in many ways similar to a programming

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 253–262


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

language, and therefore a clear and explicit formalization of first-order logic and a proof system may help the students in understanding important details. Formalizations of model theory and proof theory of first-order logic are rare, for example [6, 7, 11].

1.2 The Tool

We present the natural deduction assistant NaDeA with a formalization in the proof assistant Isabelle of its proof system. It can be used directly in a browser without any further installation and is available here:

`http://nadea.compute.dtu.dk/`

NaDeA is open source software developed in TypeScript / JavaScript and stored on GitHub. The formalization of its proof system in Isabelle is available here:

`http://logic-tools.github.io/`

Once NaDeA is loaded in the browser — about 250 KB with the jQuery Core library — no internet connection is required. Therefore NaDeA can also be stored locally.

We display the natural deduction proofs in two different formats. We present the proof in an explicit code format that is equivalent to the Isabelle syntax, but with a few syntactic differences to make it easier to understand for someone trying to learn Isabelle. In this format, we present the proof in a style very similar to that of Fitch's diagram proofs. We avoid the seemingly popular Gentzen's tree style to focus less on a visually pleasing graphical representation that is presumably much more challenging to implement.

We find that the following requirements constitute the key ideals for any natural deduction assistant. It should be:

- Easy to use.
- Clear and explicit in every detail of the proof.
- Based on a formalization that can be proved at least sound, but preferably also complete.

Based on this, we saw an opportunity to develop NaDeA which offers help for new users, but also serves to present an approach that is relevant to the advanced users.

In a paper considering the tools developed for teaching logic over the last decade [14, p. 137], the following is said about assistants (not proof assistants like Isabelle but tools for learning/teaching logic):

> Assistants are characterized by a higher degree of interactivity with the user. They provide menus and dialogues to the user for interaction purposes. This kind of tool gives the students the feeling that they are being helped in building the solution. They provide error messages and hints in the guidance to the construction of the answer. Many of them usually offer construction of solution in natural deduction proofs. [...] They are usually free licensed and of open access.

We think that this characterization in many ways fits NaDeA. While NaDeA might not bring something new to the table in the form of delicate graphical features, we emphasize the fact that it has some rather unique features such as a formalization of its proof system in Isabelle.

2 Natural Deduction in a Textbook

We consider natural deduction as presented in a popular textbook on logic in computer science [15]. First, we take a look substitution, which is central to the treatment of quantifiers in natural deduction.

2.1 On Substitution

The following definition for substitution is used in [15, p. 105 top]:

> Given a variable x, a term t and a formula ϕ we define $\phi[t/x]$ to be the formula obtained by replacing each free occurrence of variable x in ϕ with t.

The usual side conditions that come with rules using this substitution seem to be omitted. In [15, p. 106 top], we are shortly after given the following definition of what it means that 't must be free for x in ϕ':

> Given a term t, a variable x and a formula ϕ, we say that t is free for x in ϕ if no free x leaf in ϕ occurs in the scope of $\forall y$ or $\exists y$ for any variable y occurring in t.

The following quote [15, p. 106 bottom] from the same book emphasizes how it seems more preferable, due to the high level of complexity, to avoid the details of these important side conditions:

> It might be helpful to compare 't is free for x in ϕ' with a precondition of calling a procedure for substitution. If you are asked to compute $\phi[t/x]$ in your exercises or exams, then that is what you should do; but any reasonable implementation of substitution used in a theorem prover would have to check whether t is free for x in ϕ and, if not, rename some variables with fresh ones to avoid the undesirable capture of variables.

We find that this way of presenting natural deduction proof systems leaves out some important notions that the students ought to learn. In our formalization such notions and their complications become easier to explain because all side conditions of the rules are very explicitly stated. We see it as one of the major advantages of presenting this formalization to students.

2.2 Natural Deduction Rules

We now present the natural deduction rules as described in the literature, again using [15]. The first 9 are rules for classical propositional logic and the last 4 are for first-order logic. Intuitionistic logic can be obtained by omitting the rule *PBC* (proof by contradiction, called "Boole" later) and adding the $\bot$-elimination rule (also known as the rule of explosion) [16]. The rules are as follows:

$$\frac{\begin{array}{c}[\neg\phi]\\ \vdots\\ \bot\end{array}}{\phi}\ PBC \qquad \frac{\phi \qquad \phi \to \psi}{\psi} \to E \qquad \frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \to \psi} \to I$$

$$\frac{\phi \vee \psi \qquad \begin{array}{c}[\phi]\\ \vdots\\ \chi\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots\\ \chi\end{array}}{\chi} \vee E \qquad \frac{\phi}{\phi \vee \psi} \vee I_1 \qquad \frac{\psi}{\phi \vee \psi} \vee I_2$$

$$\frac{\phi \wedge \psi}{\phi}\ \wedge E_1 \qquad \frac{\phi \wedge \psi}{\psi}\ \wedge E_2 \qquad \frac{\phi \quad \psi}{\phi \wedge \psi}\ \wedge I$$

$$\frac{\exists x\, \phi \qquad \boxed{\begin{array}{ll} x_0 & \phi\,[x_0/x] \\ & \vdots \\ & \chi \end{array}}}{\chi}\ \exists E \qquad \frac{\phi\,[t/x]}{\exists x\, \phi}\ \exists I$$

$$\frac{\forall x\, \phi}{\phi\,[t/x]}\ \forall E \qquad \frac{\boxed{\begin{array}{l} x_0 \\ \vdots \\ \phi\,[x_0/x] \end{array}}}{\forall x\, \phi}\ \forall I$$

Side conditions to rules for quantifiers:

$\exists E$: x_0 cannot occur outside its box (and therefore not in χ).
$\exists I$: t must be free for x in ϕ.
$\forall E$: t must be free for x in ϕ.
$\forall I$: x_0 is a new variable which does not occur outside its box.

In addition there is a special copy rule [15, p. 20]:

> A final rule is required in order to allow us to conclude a box with a formula which has already appeared earlier in the proof. [...] The rule 'copy' allows us to repeat something that we know already. We need to do this in this example, because the rule $\rightarrow I$ requires that we end the inner box with p. The copy rule entitles us to copy formulas that appeared before, unless they depend on temporary assumptions whose box has already been closed. Though a little inelegant, this additional rule is a small price to pay for the freedom of being able to use premises, or any other 'visible' formulas, more than once.

The copy rule is not needed in our formalization due to the way it manages assumptions.

As it can be seen, there are no rules for truth, negation or biimplication, but the following equivalences can be used:

$$\begin{array}{rcl} \top & \equiv & \bot \rightarrow \bot \\ \neg A & \equiv & A \rightarrow \bot \\ A \leftrightarrow B & \equiv & (A \rightarrow B) \wedge (B \rightarrow A) \end{array}$$

The symbols A and B are arbitrary formulas.

3 Natural Deduction in NaDeA

One of the unique features of NaDeA is that it comes with a formalization in Isabelle of its proof system.

3.1 Syntax for Terms and Formulas

The terms and formulas of the first-order logic language are defined as the data types term and formula (later abbreviated tm and fm, respectively). The type identifier represents predicate and function symbols (later abbreviated id).

```
identifier := string
term := Var nat | Fun identifier [term, ..., term]
formula := Falsity | Pre identifier [term, ..., term] | Imp formula formula |
           Dis formula formula | Con formula formula | Exi formula | Uni formula
```

Truth, negation and biimplication are abbreviations. In the syntax of our formalization, we refer to variables by use of the de Bruijn indices. That is, instead of identifying a variable by use of a name, usually x, y, z etc., each variable has an index that determines its scope. The use of de Bruijn indices instead of named variables allows for a simple definition of substitution. Furthermore, it also serves the purpose of teaching the students about de Bruijn indices. Note that we are not advocating that de Bruijn indices replace the standard treatment of variables in general. It arguably makes complex formulas harder to read, but the pedagogical advance is that the notion of scope is exercised.

3.2 Natural Deduction Rules

Provability in NaDeA is defined inductively as follows:

$$\frac{\text{member p a}}{\text{OK p a}}\ \text{Assume} \qquad \frac{\text{OK Falsity ((Imp p Falsity) \# a)}}{\text{OK p a}}\ \text{Boole}$$

$$\frac{\text{OK (Imp p q) a} \qquad \text{OK p a}}{\text{OK q a}}\ \text{Imp\_E} \qquad \frac{\text{OK q (p \# a)}}{\text{OK (Imp p q) a}}\ \text{Imp\_I}$$

$$\frac{\text{OK (Dis p q) a} \qquad \text{OK r (p \# a)} \qquad \text{OK r (q \# a)}}{\text{OK r a}}\ \text{Dis\_E}$$

$$\frac{\text{OK p a}}{\text{OK (Dis p q) a}}\ \text{Dis\_I1} \qquad \frac{\text{OK q a}}{\text{OK (Dis p q) a}}\ \text{Dis\_I2}$$

$$\frac{\text{OK (Con p q) a}}{\text{OK p a}}\ \text{Con\_E1} \qquad \frac{\text{OK (Con p q) a}}{\text{OK q a}}\ \text{Con\_E2} \qquad \frac{\text{OK p a} \qquad \text{OK q a}}{\text{OK (Con p q) a}}\ \text{Con\_I}$$

$$\frac{\text{OK (Exi p) a} \qquad \text{OK q ((sub 0 (Fun c []) p) \# a)} \qquad \text{news c (p\#q\#a)}}{\text{OK q a}}\ \text{Exi\_E}$$

$$\frac{\text{OK (sub 0 t p) a}}{\text{OK (Exi p) a}}\ \text{Exi\_I}$$

$$\frac{\text{OK (Uni p)}}{\text{OK (sub 0 t p) a}}\ \text{Uni\_E} \qquad \frac{\text{OK (sub 0 (Fun c []) p) a} \qquad \text{news c (p \# a))}}{\text{OK (Uni p) a}}\ \text{Uni\_I}$$

OK p a means that the formula p follows from the list of assumptions a and member p a means that p is a member of a. The operator # is between the head and the tail of a list. news c l checks if the identifier c does not occur in the any of the formulas in the list l and sub n t p returns the formula p where the term t has been substituted for the variable with the de Bruijn index n. Instead of writing OK p a we could also use the syntax a $\vdash$ p, even in Isabelle, but we prefer a more programming-like approach. In the types we use $\Rightarrow$ for function spaces. The definitions of member, news and sub are as follow:

```
member :: fm ⇒ fm list ⇒ bool
member p [] = False
member p (q # a) = (if p = q then True else member p a)

new_term :: id ⇒ tm ⇒ bool
new_term c (Var v) = True
new_term c (Fun i l) = (if i = c then False else new_list c l)

new_list :: id ⇒ tm list ⇒ bool
new_list c [] = True
new_list c (t # l) = (if new_term c t then new_list c l else False)

new :: id ⇒ fm ⇒ bool
new c Falsity = True
new c (Pre i l) = new_list c l
new c (Imp p q) = (if new c p then new c q else False)
new c (Dis  p q) = (if new c p then new c q else False)
new c (Con p q) = (if new c p then new c q else False)
new c (Exi  p) = new c p
new c (Uni p) = new c p

news :: id ⇒ fm list ⇒ bool
news c [] = True
news c (p # a) = (if new c p then news c a else False)

inc_term :: tm ⇒ tm
inc_term (Var v) = Var (v + 1)
inc_term (Fun i l) = Fun i (inc_list l)

inc_list :: tm list ⇒ tm list
inc_list [] = []
inc_list (t # l) = inc_term t # inc_list l

sub_term :: nat ⇒ tm ⇒ tm ⇒ tm
sub_term n s (Var v) = (if v = n then s else if v > n then Var (v − 1) else Var v)
sub_term n s (Fun i l) = Fun i (sub_list n s l)

sub_list :: nat ⇒ tm ⇒ tm list ⇒ tm list
sub_list n s [] = []
sub_list n s (t # l) = sub_term n s t # sub_list n s l

sub :: nat ⇒ tm ⇒ fm ⇒ fm
sub n s Falsity = Falsity
sub n s (Pre i l) = Pre i (sub_list n s l)
sub n s (Imp p q) = Imp (sub n s p) (sub n s q)
sub n s (Dis  p q) = Dis  (sub n s p) (sub n s q)
sub n s (Con p q) = Con (sub n s p) (sub n s q)
sub n s (Exi  p) = Exi  (sub (n + 1) (inc_term s) p)
sub n s (Uni p) = Uni (sub (n + 1) (inc_term s) p)
```

3.3 Semantics for Terms and Formulas

To give meaning to formulas and to prove NaDeA sound we need a semantics of the first-order logic language. This semantics is defined in the formalization in Isabelle, and it is thus not part of the tool itself. We present the semantics below. e is the environment, i.e. a mapping of variables to elements. f maps function symbols to the maps they represent. These maps are from lists of elements of the universe to elements of the universe. Likewise, g maps predicate symbols to the maps they represent. 'u is a type variable that represents the universe. In can be instantiated with any type. For instance, it can be instantiated with the natural numbers, the real number or strings.

```
semantics_term :: (nat ⇒ 'u) ⇒ (id ⇒ 'u list ⇒ 'u) ⇒ tm ⇒ 'u
semantics_term e f (Var v) = e v
semantics_term e f (Fun i l) = f i (semantics_list e f l)

semantics_list :: (nat ⇒ 'u) ⇒ (id ⇒ 'u list ⇒ 'u) ⇒ tm list ⇒ 'u list
semantics_list e f [] = []
semantics_list e f (t # l) = semantics_term e f t # semantics_list e f l

semantics :: (nat ⇒ 'u) ⇒ (id ⇒ 'u list ⇒ 'u) ⇒ (id ⇒ 'u list ⇒ bool) ⇒ fm ⇒ bool
semantics e f g Falsity = False
semantics e f g (Pre i l) = g i (semantics_list e f l)
semantics e f g (Imp p q) = (if semantics e f g p then semantics e f g q else True)
semantics e f g (Dis  p q) = (if semantics e f g p then True else semantics e f g q)
semantics e f g (Con p q) = (if semantics e f g p then semantics e f g q else False)
semantics e f g (Exi  p) = (? x. semantics (% n. if n = 0 then x else e (n − 1)) f g p)
semantics e f g (Uni p) = (! x. semantics (% n. if n = 0 then x else e (n − 1)) f g p)
```

Most of the cases of semantics should be self-explanatory, but the Uni case is complicated. The details are not important here, but in the case for Uni it uses the universal quantifier (!) of Isabelle's higher-order logic to consider all values of the universe. It also uses the lambda abstraction operator (%) to keep track of the indices of the variables. Likewise, the case for Exi uses the existential quantifier (?) of Isabelle's higher-order logic.

We have proved soundness of the formalization in Isabelle (shown here as a derived rule):

$$\frac{\text{OK p []}}{\text{semantics e f g p}}\ \text{Soundness}$$

This result makes NaDeA interesting to a broader audience since it gives confidence in the formulas proved using the tool.

4 Construction of a Proof

We now describe the core features of NaDeA from the perspective of the user. That is, we uncover how to use NaDeA to conduct and edit a proof as well as how proofs are presented.

In order to start a proof, you have to start by specifying the goal formula, that is, the formula you wish to prove. To do so, you must enable editing mode by clicking the Edit button in the top menu bar. This will show the underlying proof code and you can build formulas by clicking the red ¤ symbol. Alternatively, you can load a number of tests by clicking the Load button.

At all times, once you have fully specified the conclusion of any given rule, you can continue the proof by selecting the next rule to apply. Again you can do this by clicking the

the red ¤ symbol. Furthermore, NaDeA allows for undoing and redoing editing steps with no limits.

All proofs are conducted in backward-chaining mode. That is, you must start by specifying the formula that you wish to prove. You then apply the rules inductively until you reach a proof — if you can find one. The proof is finished by automatic application of the Assume rule once the conclusion of a rule is found in the list of assumptions.

To start over on a new proof, you can load the blank proof by using the Load button, or you can refresh the page. Please note that any unsaved work will then be gone.

In NaDeA we present any given natural deduction proof (or an attempt at one) in two different types of syntax. One syntax follows the rules as defined in section 3.2 and is closely related to the formalization in Isabelle, but with a redefined and more simple syntax in terms of learning. The proof is not built as most often seen in the literature about natural deduction. Usually, for each rule the premises are placed above its conclusion separated by a line. We instead follow the procedure of placing each premise of the rule on separate lines below its conclusion with an additional level of indentation.

Natural Deduction Assistant

1	Imp_I	[ ] P∧(P→Q)→Q
2	Imp_E	[P∧(P→Q)] Q
3	Con_E2	[P∧(P→Q)] P→Q
4	Assume	[P∧(P→Q)] P∧(P→Q)
5	Con_E1	[P∧(P→Q)] P
6	Assume	[P∧(P→Q)] P∧(P→Q)

$$\dfrac{\dfrac{\dfrac{}{p \wedge (p \to q)}\,(1)}{p \to q} \qquad \dfrac{\dfrac{}{p \wedge (p \to q)}\,(1)}{p}}{\dfrac{q}{p \wedge (p \to q) \to q}\,(1)}$$

The above proof also can be written in terms of the OK syntax as follows:

```
1  OK (Imp (Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" []))) (Pre "Q" [])) []   Imp_I
2    OK (Pre "Q" []) [(Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))]       Imp_E
3      OK (Imp (Pre "P" []) (Pre "Q" []))
           [(Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))]                 Con_E2
4        OK (Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))
           [(Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))]                 Assume
5      OK (Pre "P" []) [Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" []))]       Con_E1
6        OK (Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))
           [(Con (Pre "P" []) (Imp (Pre "P" []) (Pre "Q" [])))]                 Assume
```

5 Related Work

Throughout the development of NaDeA we have considered some of the natural deduction assistants currently available. Several of the tools available share some common flaws. They can be hard to get started with, or depend on a specific platform. However, there are also many tools that each bring something useful and unique to the table. One of the most prominent is Panda, described in [13]. Panda includes a lot of graphical features that make it fast for the experienced user to conduct proofs, and it helps the beginners to tread safely. Another characteristic of Panda is the possibility to edit proofs partially before combining them into a whole. It definitely serves well to reduce the confusion and complexity involved in conducting large proofs. However, we still believe that the way of presenting the proof

can be more explicit. In NaDeA, every detail is clearly stated as part of the proof code. In that sense, the students should become more aware of the side conditions to rules and how they work.

Another tool that deserves mention is ProofWeb [10] which is open source software for teaching natural deduction. It provides interaction between some proof assistants (Coq, Isabelle, Lego) and a web interface. The tool is highly advanced in its features and uses its own syntax. Also, it gives the user the possibility to display the proof in different formats. However, the advanced features come at the cost of being very complex for undergraduate students and require that you learn a new syntax. It serves as a great tool for anyone familiar with natural deduction that wants to conduct complex proofs that can be verified by the system. It may, on the other hand, prove less useful for teaching natural deduction to beginners since there is no easy way to get started. In NaDeA, you are free to apply any (applicable) rule to a given formula, and thus, beginners have the freedom to play around with the proof system in a safe way. Furthermore, the formalized soundness result for the proof system of NaDeA makes it relevant for a broader audience, since this gives confidence in that the formulas proved with the tool are actually valid.

6 Further Work

In NaDeA there is support for proofs in propositional logic as well as first-order logic. We would also like to extend to more complex logic languages, the most natural step being higher-order logic. This could be achieved using the CakeML approach [8]. Other branches of logic would also be interesting, and the possibilities are numerous. Apart from just extending the natural deduction proof system to support other types of logic, another option is to implement other proof systems as well.

Because the NaDeA tool has a formalization in Isabelle of its proof system, we would like to provide features that allow for a direct integration with Isabelle. For instance, we would like to allow for proofs to be exported to an Isabelle format that could verify the correctness of the proofs. A formal verification of the implementation would require much effort, but perhaps it could be reimplemented on top of Isabelle (although probably not in TypeScript / JavaScript).

We would like to extend NaDeA with more features in order to help the user in conducting proofs and in understanding logic. For example, the tool could be extended with step-by-step execution of the auxiliary primitive recursive functions used in the side conditions of the natural deduction rules.

So far only a small group of computer science students have tested NaDeA, but it will be classroom tested with around 60 bachelor students in the next semester. Currently the tool has no support for student assignments and automatic feedback and/or grading. The tool could be extended such that the students are evaluated and perhaps given a score based on the proofs they conduct. It is not obvious how this could best be implemented. We hope to find the resources for the development of such features but already now we think that the tool has the potential to be one of the main ways to teach logic in mathematics and computer science.

Acknowledgements We would like to thank Stefan Berghofer for discussions about the formalization of natural deduction in Isabelle. We would also like to thank Andreas Halkjær From and Andreas Viktor Hess for comments on the paper.

References

1 Mordechai Ben-Ari. Mathematical Logic for Computer Science. Third Edition. Springer 2012.
2 Tobias Nipkow, Lawrence C. Paulson and Markus Wenzel. Isabelle/HOL - A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science 2283, Springer 2002.
3 Dag Prawitz. Natural Deduction. A Proof-Theoretic Study. Stockholm: Almqvist & Wiksell 1965.
4 Francis Jeffry Pelletier. A Brief History of Natural Deduction. History and Philosophy of Logic, 1-31, 1999.
5 Melvin Fitting. First-Order Logic and Automated Theorem Proving. Second Edition Springer 1996.
6 John Harrison. Formalizing Basic First Order Model Theory. Lecture Notes in Computer Science 1497, 153–170, Springer 1998.
7 Stefan Berghofer. First-Order Logic According to Fitting. Formal Proof Development. Archive of Formal Proofs 2007.
8 Ramana Kumar, Rob Arthan, Magnus O. Myreen and Scott Owens. HOL with Definitions: Semantics, Soundness, and a Verified Implementation. Lecture Notes in Computer Science 8558, 308–324, Springer 2014.
9 Jørgen Villadsen, Anders Schlichtkrull and Andreas Viktor Hess. Meta-Logical Reasoning in Higher-Order Logic. Accepted at 29th International Symposium Logica, Hejnice Monastery, Czech Republic, 15-19 June 2015.
10 ProofWeb. Online `http://proofweb.cs.ru.nl/login.php` (ProofWeb is both a system for teaching logic and for using proof assistants through the web). Accessed April 2015.
11 Jasmin Christian Blanchette, Andrei Popescu and Dmitriy Traytel. Unified Classical Logic Completeness - A Coinductive Pearl. Lecture Notes in Computer Science 8562, 46–60, 2014.
12 Krysia Broda, Jiefei Ma, Gabrielle Sinnadurai and Alexander Summers. Pandora: A Reasoning Toolbox Using Natural Deduction Style. Logic Journal of IGPL, 15(4):293–304, 2007.
13 Olivier Gasquet, François Schwarzentruber and Martin Strecker. Panda: A Proof Assistant in Natural Deduction for All. A Gentzen Style Proof Assistant for Undergraduate Students. Lecture Notes in Computer Science 6680, 85–92. Springer 2011.
14 Antonia Huertas. Ten Years of Computer-Based Tutors for Teaching Logic 2000-2010: Lessons learned. Lecture Notes in Computer Science 6680, 131–140, Springer 2011.
15 Michael Huth and Mark Ryan. Logic in Computer Science: Modelling and Reasoning About Systems. Second Edition. Cambridge University Press 2004.
16 Jonathan P. Seldin. Normalization and Excluded Middle. I. Studia Logica, 48(2):193–217, 1989.

Logic Modelling*

Roger Villemaire[1]

1 **Department of Computer Science**
Université du Québec à Montréal
C.P. 8888, Succ. Centre-ville,
Montréal, Québec, H3C 3P8, Canada
villemaire.roger@uqam.ca

Abstract

This is a reflection on the author's experience in teaching logic at the graduate level in a computer science department. The main lesson is that model building and the process of modelling must be placed at the centre stage of logic teaching. Furthermore, effective use must be supported with adequate tools. Finally, logic is the methodology underlying many applications, it is hence paramount to pass on its principles, methods and concepts to computer science audiences.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.2 Automatic Programming, I.2.3 Deduction and Theorem Proving, I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases teaching, logic, verification, automatic theorem proving, knowledge representation

1 Introduction

Academically, computer science emerged from engineering and mathematics propelled by the development of computing hardware. While logic is clearly identified with the theoretical foundation of computing, it also plays an important role in many applications. Unfortunately while its method are central to many technologies, logic goes almost unnoticed for most computer scientists.

Still, there are logic courses in computer science departments, at least at my university. But the question emerges as to what to teach and how. I present in this paper three graduates logic courses that I developed over the years. Each has different objectives and aim at a different audience. Nevertheless, all three have in common to present applications and to develop the students' abilities to effectively use the material. They also illustrate how modelling is at the heart of logic, as it is used in computing.

In order to present my approach, I will briefly describe the courses' contents in Section 2 before presenting in Section 3 a unifying view for applications of logic to computer science and showing that my courses are indeed structured that way. I will also describe surprises and difficulties that the students usually encounter. Finally, in Section 4, I will conclude on my teaching experience.

Logic offers a quite general methodology and its ideas, methods and algorithms underlie many important applications. But this goes almost unnoticed, holding up the development of new applications. This paper presents my approach to reaching out to graduate computer science students.

* The author gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 263–272

Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

2 Courses presentation

This section briefly presents the three courses considered in this paper. All are graduate courses, but with different contents and quite different audiences. All courses are 15 weeks long, with the class meeting once a week for a 3 hours lecture. Between a third and the half of these 45 contact hours are spent on exercises and effective use of the tools. While each student is encouraged to come up with his own solutions, groups are usually small (between 4 and 12 students) and there is therefore a lot of interaction between students and with the lecturer.

A word on the students' background and their previous exposure to logic and the fundamentals of computer science is in order. For the computer science and software engineering classes presented in subsections 2.1 and 2.2, while students previously attended usually many different undergraduate institutions, they all have a basic training in discrete mathematics and algorithmic. As for logic itself, they have usually learned some propositional and first-order logic, but mainly as a formalism to express properties with no reference to inference and deduction systems. As for the course presented in subsection 2.3, the audience's background is broad, ranging from computer science to the humanities. While the students with a computer science background have a training similar to the students of the two previous courses, this is not the case for students from the humanities. Nevertheless, this last group of students have usually been exposed to some form of formal systems. This is particularly true for students with a background in linguistic, psychology and philosophy, which form the major part of the humanities students in this class.

2.1 Modelling and Verification

This course covers symbolic model-checking [6] and is an optional course in the master's and PhD in computer science, both being traditional research oriented degrees.

The first half of the course is devoted to modelling and the second half to covering the algorithms and data structures allowing efficient verification.

Modelling was done up to the last term with the NuSMV tool [5, 4] and now with its follow-up NuXMV [2]. Without going into the details of the tool's language, a NuSMV/NuXMV model defines variables that range over finite domains and describes variables' evolution. Initial values are assigned to variables and the (discrete) evolution of the system is given by specifying *next* values of variables. The underlying model is hence a finite state machine, where a *state* is determined by the value of variables at some moment and the *next state* is determined by the next values of the variables. The tool supports non-deterministic behaviour by allowing more than one initial and next values for variables. The tool determines the validity of *Linear Temporal Logic* (LTL) and *Computation Tree Logic* (CTL) formulas on a given model. The reader is referred to [7] for an introduction to temporal logic.

The algorithmic methods presented in the course (and supported by the tool) are *Bounded Model-Checking* (BMC) [1], which is based on efficient SAT-solvers, and BDD-based model-checking [3]. Detailed lecture notes (in French) have been produced [9] and are freely available under a Creative Common license.

2.2 Formal Methods

This course covers formal methods in software engineering and is an optional course in the master's degree in software engineering, a professional degree with mostly part-time students.

Software modelling is done in the *Unified Modeling Language* (UML) a standard of the *Object Management Group* (OMG)[1]. Since this is the standard in the profession, the students usually already master this language and have at least basic modelling skills. The course introduces logic as a tool to refine and make models more detailed and precise. Two applications are presented. First test generation and then software verification. Finally, sequent calculus is introduced as this is the method used by the software verification tool (see Section 3.2.2). This course follows a "hands-on" approach, where each class combines concepts and notions that are immediately put in application.

The logics presented are the *Object Constraint Language* (OCL), a part of the UML standard, and *Java Modeling Language* (JML), that is specifically designed for Java programs. Modelling in done with the *Eclipse Modeling Tools*[2]. An Eclipse plug-in, *Dresden OCL*[3], is used to help writing OCL expressions and to generate tests. Finally, the *KeY*[4] *Automatic Theorem Prover* (ATP) is used to verify JML properties on Java code. Slides, examples, and course material are available on the course's Web site (in French)[5].

2.3 Logic, Informatics and Cognitive Sciences

This course covers logic as a tool for human cognition and knowledge representation. It is an optional course for students in the *Cognitive Informatics Doctorate*, a joint program between the computer science department and the faculty of humanities. Following the tradition in this program, the course is given by two lecturers, namely, my colleague Serge Robert, a logician from the philosophy department, and myself.

After the first lecture where we present the course, its objectives, and our approach, and we ask the students about their expectations, we each take care of half of the lectures. Serge covers topics such as classical, non-monotonic, many-valued, and probabilistic logics in relation to human cognition. For my part, I concentrate on knowledge representation with modal and description logics. I introduce tableaux as an inference method and allow time in each class to let the students represent some knowledge and use a tool to experiment with inference.

For modal logic, I use LoTREC [8], in fact a slight modified version[6]. This tool allows me to attain two different, but fundamental, objectives. First, the student can graphically visualise tableau proofs and hence verify that she/he correctly masters the algorithm. This is particularly useful for students of the humanities that can be quite intimidated by algorithmic contents. Secondly, the tool allows me to put tableaux at work to infer new knowledge. Students can hence experiment and determine shortcomings of their tentative representations. This is particularly important when faced with the choice of the appropriate modal logic, such as K4, S4, or S5.

For description logic, students are introduced to devising ontologies in Protégé[7]. Within this tool, Hermit[8] is used to derive the inferred hierarchy. This allows the student to test whether her/his ontology is free of contradictions and, more importantly, to see in action how tableaux can infer derived knowledge.

---

[1] www.omg.org
[2] www.eclipse.org/modeling/
[3] www.dresden-ocl.org/
[4] www.key-project.org
[5] moka.labunix.uqam.ca/~villemaire_r/7160.html
[6] LoTREC 2.1 moka.labunix.uqam.ca/~villemaire_r/LoTREC/
[7] protege.stanford.edu
[8] hermit-reasoner.com

Slides, examples, and course material are available on the course's Web site (in French)[9].

3 Logic: Information, Representation and Processing

What is logic and how can it be presented to a computer science audience? I present in this section a view of logic around information processing, describe the material from my courses within this unifying perspective, and, finally, discuss the typical difficulties and surprises students encounter during my lectures.

3.1 Information

Logic is a tool to represent and process information. While computing, a.k.a. informatics, is the science of information, it should be stressed that information goes much further than what is usually considered as data. Indeed, logic offers quite a large range of abstraction levels. One can, as in databases, speak of a specific *person P having the person M as mother*, but one can also consider the *motherhood* relation and state that *every person has a mother*. Information has content and meaning, and it is used for some purpose. Obviously, quantity and relevance also matters: one cannot conclude if there is not enough information or if it is not relevant. Information must be stated in some way, this is what logic is about.

3.2 Models and Properties

Logic offers formalisms to represent information[10]. Being formal means that there is a precise syntax and semantics, much less ambiguous than for natural languages and more easily machine processed. Logic is hence a general approach to knowledge representation, allowing many levels of abstraction. Essentially, any knowledge can be represented, but the exact formalism and formalisation must be chosen with care, keeping in mind the intended goal. Logical formalisms hence make precise, concepts, ideas, relations, and data.

Aiming at applications to computing, a logic description can be presented as a pair formed of a *model* and a *property*. Technically, this distinction is artificial, since all pieces of information could be defined either all within the model or all within the property. But this distinction is convenient and natural.

The distinction between model and property lies in fact in the intention. The model is intended as a (partial) description of the system. We borrow the notion of *system* directly from the practice of computer science. It is (the formalisation of) the entity one wants to interact with. One can think of a database, a software module, a communication protocol, the description of some process, a knowledge base, etc. A model is some, possibly partial, view of some system of interest. By contrast, a property is a constraint (on the model) that one wants to check or verify. For their part, properties are naturally partial; we are not describing everything that should hold, but something that should hold.

Model and property are closely bound to one another. The model is meaningful for a specific goal, (essentially) formalised by the property. Conversely, the property is checked on the model. It is hence a constraint on the model's elements or *attributes* as they are called in software engineering. As modelling is a creative endeavour, there is no single right model or property; model and property must be appropriate for the intended goal.

---

[9] `moka.labunix.uqam.ca/~villemaire_r/9305.html`

[10] We limit ourselves to *formal* logic in this paper.

To sum up, from the perspective of a computer scientist making effective use of logic, the model describes a system whose significance transcends the verification task, while the property is confined at this task's heart, as it expresses what will be verified.

This distinction, natural from a computing point-of-view, applies nicely to the material presented in the three courses.

3.2.1 Modelling and Verification

In this class, the model is represented by a *finite state machine*. One defines (finitely many) variables ranging over finite data types and assigns initial and next values to variables. As the model's evolution is described in discrete time, the property is stated in LTL or CTL.

This modelling process can be naturally introduced to computer scientists, since at the graduate level they are already quite experienced in software development. A piece a code is indeed a formal model, but of an extremely detailed form. As for the property, which seems at first less natural from a computing point-of-view, students have encountered assertions and testing in software development and checking a property is hardly something new.

While this analogy helps introducing the concepts and get the student engaged with the material, it only goes so far. It is important to emphasise that the objective in formal verification is to check that the model fulfil the property and not to implement and debug a functional system. This comes at some surprise to mainly development-oriented students and necessitates a throughout discussion supported by hands-on modelling and verification experience during classes. Furthermore, contrary to testing, the objective it to verify the property for *all behaviours* of the model. This will offer some difficulties.

The students' natural tendency is to develop overly detailed models. A main objective of the course is to make the students aware that this is neither needed nor appropriate. Indeed, the goal is to verify a property which usually depends only on a limited number of details. Furthermore, a simpler model will usually mean a more efficient verification. Finally, a simpler model will be easier to develop (or generate) and maintain, a consideration natural for computer science students.

A nice simplification example is offered by a scheduler that guarantees, at each moment, the execution of a unique component. There may be some "real" deterministic scheduler implementing some complex algorithm. But if the objective is not to show the scheduler's correctness, but rather that of illustrating a more complex system using a scheduler, non-determinism can largely simplify the model. Indeed, in this case the model can simply specify that the scheduler always chooses a unique component without having to detail how this choice is done.

In developing models during the class, students are also led to realise that models can be of quite different nature. For instance, in modelling a network one can either models packet flow or else abstract away and concentrates on the dependencies between the network equipments' configurations. The real question being always that the model should be adequate to the verification task.

As surprising as it may seem, students have not major difficulties writing LTL or CTL formulas. This may come from the fact that in model checking the model is usually complex and the formula simple. The difficulties usually come from the verification process. Indeed, a slight mismatch between the model and the formula will often lead the tool to return an unexpected counterexample. Often this counter-example hints in fact to the subtle difference between the intended and the actual formalisation. As the students can be rapidly overwhelmed by the subtlety uncovered by the counter-example, it is paramount to teach a progressive and systematic way to analyse counter-examples.

3.2.2 Formal Methods

This is a course of a software engineering program. Accordingly, the models considered are cornerstones of the profession, namely, UML class diagrams and Java code. Simply speaking, a UML class diagram describes classes (set of objects), (binary) relations between classes, and, finally, operations, which are function names with arguments, return value, and their types (classes). A UML class diagram does not describe the system's behaviour as there is no provision to describe how operations should behave. In order to consider software behaviour, this course relies on the programming language Java.

For properties, OCL, a not so well-known part of the UML standard, and the dynamic logic of the KeY tool are used. Some explanations about these formalisms are in order.

In a class diagram, relations are binary, but can be of any kind: one-to-one, one-to-many, many-to-one, or many-to-many. Essentially, OCL allows one to constrain relation extremities and in the case of a "many" extremity, its size. For example, a `hasPassed` relation between a student class and a course class would be many-to-many, as each student can have passed many courses, and each course can have been passed by many students. A possible OCL constraint would be to state that every (graduate) student must have passed 30 (undergraduate) courses. Is some way, OCL allows one to write constraints reminiscent of (multi-)modal logic or description logic.

This course also uses the KeY tool's logic to verify Java code. This is a dynamic logic built from atomic formulas (comparisons between variables and values), propositional connectives and modal connectives $\langle a \rangle$, $[a]$, where a is a Java program. Here $\langle a \rangle \varphi$ means that a will terminate normally (no exception will be raised) and then φ will hold, while $[a]\varphi$ means that if a terminates normally, then φ holds.

In this course, students have little difficulties with models, since they are already trained in devising UML class diagrams and developing Java code. On the opposite, writing OCL constraints is quite challenging. But the benefit is that it coerces them to take UML modelling seriously. Indeed, in typical software engineering courses, software design is done by adding constraints as comments to UML diagrams. Documentation in natural languages can be pretty imprecise and its correctness tedious to assure. Students are hence not incited toward rigour in UML modelling. On the opposite, writing up OCL constraints forces students to get committed and make their ideas more precise. It also offers, as students appreciate, machine support.

Indeed, OCL being formal, the tool, in our case Dresden OCL, detects such errors as type mismatch. As simple as this may seem, it has a considerable impact. Devising an OCL constraint imposes to navigate the class diagram to access the data. Something as simple as incorrect type detection can already come a long way to notice incorrect navigation. Finally, Dresden OCL allows one to generate Java tests from OCL constraints, showing again the benefit of machine processing of formal languages.

For code verification, the course introduces JML, which allows one to state class invariants and contracts in the form of pre/post conditions on operations. Pre and post conditions are quite natural for students, since they are already largely used to document code. Invariants are more challenging, since they define what a correct instance of a class must be, a modelling task nevertheless completely in line with the course.

The KeY tool is an automatic theorem prover for JML contracts (under the fact that invariants hold). The tool implements a sequent deduction system for its dynamic logic. Students learn sequent proofs quite rapidly, since it is a rule based system. The greatest difficulty is coping with inconclusive proofs. In such a case, I engage students to be attentive to proof branching and unjustified sequents (in particular their meanings), in the search for

a missing piece of information. For instance, if an open sequent premise contains $x = null$, one should consider whether $x \neq null$ shouldn't be added to preconditions. Unfortunately, proof analysis is challenging since one must master the formalised Java execution model, in particular the quite complex memory model.

Students find this last part quite challenging and they are right that it considerably increases the burden of code development. But, it must be said that this material is presented as an advanced method appropriate when expectation is high, such as for safety critical code.

3.2.3 Logic, Informatics and Cognitive Sciences

In this course both the model and the property are formalised within a single logic. The model is intended as the representation of some knowledge, while the property is a query on this knowledge. One hence wants to determine whether the property is a consequence of the knowledge or not. The logics covered are modal logic and its extension, description logic.

Modal logic extends propositional logic with two unary connectors $\Box$ and $\Diamond$. Under Kripke semantics, the connectors $\Box$ and $\Diamond$ are interpreted via a binary relation with the intuitive meaning "for all neighbours" and "there is a neighbour", respectively. Modal logic is used to represent facts about necessity/possibility, time, and knowledge. Description logic is an extension of modal logic allowing multiple binary relations and is used in the context of the semantic web to describe and process ontologies, which are, broadly speaking, structured terminologies.

Students in this course come partly from computer science and partly from the humanities. Emphasis is not on theoretical and algorithmic treatments, but rather on the meaning that we want to convey through modelling. Semantics is paramount and the meanings of various typical formulas are analysed, compared, and discussed. Much time is spent on representing knowledge, exploring and comparing alternatives. But such a course could not be complete without explaining how inference can be mechanised. Tableaux are introduced as a common inference framework, first for propositional logic, then for modal logic and finally, in outline, for description logic.

An "information flow" approach to modelling is put forward in this course. Reflecting on the model and property, it is emphasised that the model should formalise enough knowledge to be able to draw a conclusion on the property. It is furthermore emphasised that many conclusions can be inferred from the model, but no more than follow from its intuitive meaning. The completeness of tableaux systems is proved, bridging the gap between the intended semantics and algorithmic procedures. It often comes quite as a surprise, for students, that meaning can be processed in such a general way. The significance of completeness is furthermore stressed in concrete situations by modelling significant knowledge. It is worth noting that as tableaux form a deduction system operating on the formulas structures, it falls well into this "information flow" approach.

Students are largely stunned by logic. Many students of the humanities are at first disconcerted that a pretty general formalism, such as modal logic, can model possibility/necessity, time, and knowledge. They would rather expect a specific formalism for each situation. It is hence relevant to put logic forward as a universal approach to knowledge representation, similarly to arithmetic that is an adequate formalism for counting things of all nature.

Computer science students have also their own surprises. While they are typically exposed as undergraduates to propositional and first-order logic as representation systems, they have rarely encountered deduction and hence the power of inference in logic. This is particularly significant with description logic. Many have been introduced to the semantic web, but, as usual in that field, mostly as a set of technologies with emphasis on encodings and standards.

It is hence a surprise for them that description logic in not only a knowledge representation framework, put also an inference framework allowing to draw up new facts.

4 Conclusion

When I started to teach logic to computer science audiences, almost twenty-five years ago, my now retired colleague Lorne Bouchard advised me to find the proper tools, if I really wanted to engage the student in the subject. I hence always try to find a tool that is efficient, but also usable in practice to handle significant problems. I some sort of way, I don't look for a tool for teaching logic, but more to the opposite, for a tool that can be used in practice and that uses some form of logic.

This means that I build a course around effective modelling and the use of the tool to process a model toward some useful goal. But, particularly at the graduate level, the students must be exposed to the conceptual machinery that allows such tools in the first place. I hence always introduce the concepts and methods of logic, with a level of justification appropriate for the audience.

There are more and more efficient computing tools powered by ideas, methods, and algorithms from logic. This opens up the way to make computer scientists more aware not only of the power of logic, but also of its methods. There is a logic methodology, based on modelling and powered by efficient engines, that should be much better known in computing. Ironically, there are more and more applications of logic to computer science, but unfortunately still inadequate exposure to our field. I hope to see, in the future, many more applications, but also the return of logic to the central stage of computer science teaching. Logic has a message, something particular to convey, that should properly reach computer science students.

Acknowledgments I would like to thank the anonymous reviewers for numerous comments and suggestions that have improved the final version of this paper.

References

1 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, London, UK, 1999. Springer-Verlag.

2 Marco Bozzano, Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. *nuXmv 1.0 User Manual*. FBK, Via Sommarive 18, 38055 Povo (Trento) Italy, 2014.

3 Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

4 Roberto Cavada, Alessandro Cimatti, Charles Arthur Jochim, Gavin Keighren, Emanuele Olivetti, Marco Pistore, Marco Roveri, and Andrei Tchaltsev. *NuSMV 2.4 User Manual*. ITC-irst, Via Sommarive 18, 38055 Povo (Trento) Italy, 1998-2005.

5 Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: An open-source tool for symbolic model checking. In *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, pages 359–364, London, UK, 2002. Springer-Verlag.

6 Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, Cambridge, MA, 2000.

7 E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. Elsevier and MIT Press, 1990.

8 Luis Fariñas del Cerro, Olivier Gasquet, Andreas Herzig, and Mohamed Saade. LoTREC: An environment for experiencing Kripke Semantics. In Maria Manzano, Belén Pérez Lancho, and Ana Gil, editors, *International Congress on Tools for Teaching Logic (ICTTL), Salamanca, Spain, 26/09/2006-30/09/2006*, pages 41–44, www.usal.es, septembre 2006. University of Salamanca. ISBN: 84-690-0348-8.

9 Roger Villemaire. Modélisation et Vérification. `www.info2.uqam.ca/~villemaire_r/7570/ModelisationEtVerification/mev.php`, 2013. Creative Commons PPP.

Teaching Logic to Information Systems Students: Challenges and Opportunities

Anna Zamansky[1] **and Eitan Farchi**[2]

1 **Information Systems Department, University of Haifa**
Haifa, Israel
annazam@is.haifa.ac.il

2 **IBM Research**
Haifa, Israel
farchi@ibm.il

Abstract

In contrast to Computer Science, where the fundamental role of Logic is widely recognized, it plays a practically non-existent role in Information Systems curricula. In this paper we argue that instead of Logic's exclusion from the IS curriculum, a significant adaptation of the contents, as well as teaching methodologies, is required for an alignment with the needs of IS practitioners. We present our vision for such adaptation and report on concrete steps towards its implementation in the design and teaching of a course for graduate IS students at the University of Haifa. We discuss the course plan and present some data on the students' feedback on the course.

Keywords and phrases Information Systems, Logic, Education, Formal Methods

1 Introduction

The fundamental role of Logic in the Computer Science curriculum is widely recognized. The ACM CS undergraduate curriculum guidelines ([16]) explicitly state logic as a mathematical requirement which is directly relevant for the large majority of all CS undergraduates (together with elements of set theory and discrete probability). This recommendation is implemented in most of the standard CS undergraduate study programs by including in the curriculum a course in discrete structures which includes a significant amount of formal logic.

The (academic) field of Information Systems (IS) encompasses two broad areas: (i) acquisition, deployment, and management of information technology resources and services, and (ii) development and evolution of infrastructure and systems for use in organization processes. Thus, as opposed to CS, IS's primary focus is on an organization's mission and objectives and the application of information technology to further these goals. Yet both IS and CS require a common subset of technical knowledge, reflected also in the intersection of the respective study programs' curricula.

Logic, however, does not appear to be in this intersection – almost none of the IS undergraduate study programs include such course in their curriculum. The ACM IS curriculum guidelines ([18]), which mention statistics and probability as required core IS topics and discrete mathematics as an optional one, but do not refer to logic as relevant: *"Even though IS professionals do not need the same level of mathematical depth as many other computing professionals, there are, however, some core elements that are very important for IS professionals. To support in-depth analysis of data, IS professionals should have a strong background in statistics and probability. For those who are interested in building a strong skill set in algorithmic thinking, discrete mathematics is important."*

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 273–280


Université de Rennes 1
LIPICS Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)

We believe the current state of affairs is suboptimal for several reasons. First of all, all the reasons for including logic in the CS curricula still hold in the IS domain: it is widely acknowledged that studying logic directly contributes to software development skills, confirmed recently also by empirical studies ([14]). Secondly, the lack of experience with formal notation forms a major cognitive barrier to the adoption of formal methods in the industry ([21]). This is further reinforced by the fact that because many IS study programs tend to be marketed as programs "excluding the hard math"[1], the students come to see the lack of mathematical courses as a benefit, and express disappointment[2] when any formal notations are integrated in the IS core courses – thus creating a vicious circle.

Although we tend to agree that a typical IS major may need a less extensive mathematical background than a CS major, we believe that rather than excluding logic from the IS curriculum, a significant adaptation of the contents that are taught to align with IS objectives. Recently more voices are calling for a reconsideration of the traditional logic curriculum and its adaptation to the needs of future pracitioners. Several proposals on *what* to teach and *how* to teach it have been made in the context of CS ([13, 17, 19]).

In this paper we address these questions in the context of the IS curriculum. We report on our experience in designing and teaching the course "Logic and Formal Specification" to graduate students at the Information Systems (IS) department at the University of Haifa, which is one of the few to include a *mandatory* course on logic and formal methods in its graduate study program. We discuss our view of *what* should be included in the "IS logic toolbox" in order for the students to be able to carry out activities for checking (by proof, analysis or testing) that a software system meets specifications and fulfills its intended purpose. These tools should *at least* (i) by providing background on induction and propositional and first-order logic, and (ii) by providing the ability to read, write and reason about formal specifications. We share our insights on *how* the above can be achieved: (i) excluding complex mathematical intricacies, and (ii) providing simple yet software-related examples. Concerning the latter, we report on an ongoing work to develop a tool for measuring "simplicity" of Z specifications. While the above insights are yet to be empirically validated, some indications of the benefits of our approach are reflected in our students' feedback, which we briefly discuss below.

2 The IS Logic Toolbox

The main practical objective in teaching logic to IS practitioners is to give them the ability to apply formal methods in industry. Application of formal aspects is particularly important for software quality control, i.e., activities for checking (by proof, analysis or testing) that a software system meets specifications and that it fulfills its intended purpose.

---

[1] Below are some exemplary quotes from webpages of academic institutions providing both CS and IS degrees on the comparison between the two:

- "As a rule, computer science requires more mathematics and analytical skill than information systems. Also, our experience has shown that it is easier to move from being a CS major to an IS major than the other way around. Therefore, if you feel reasonably comfortable with the math requirements, then you should start out as a computer science major." (Saint Michael's College)
- "A major in CS will know a considerable amount of mathematics which will help in technological applications...An IS major needs to be aware of what information technology can contribute to an organization and how to bring that solution to fruition." (University of Missouri-St. Louis).

[2] Quoting one of our graduate students who was assigned to read a research paper on formal methods: "When I see formal definitions, I just want to cry." Notably, she is one of the best students in her class.

Due to the density of the IS curricula, currently one cannot afford to have one course on pure formal logic and then another on formal methods (this problem is also discussed in [20] in the context of CS). Therefore, one must develop some mixture of a introductory formal logic together with introduction to formal methods relevant for the IS domain. In what follows we briefly survey previous reflections on the content of logic and formal methods courses that practitioners *really* need and their integration into the curricula, and propose how to adapt the proposed ideas for the context of IS.

2.1 Previous Proposals

Recently there has been an ongoing discussion on whether the traditional logic syllabus for CS is relevant for practitioners. Since our main goal in this paper is to extend and adapt this discussion to the context of IS, we start by briefly outlining some relevant proposals (mostly in the context of CS), the ideas of which are close in spirit to the vision we present below.

In his paper "From Hilbert to a Logic Toolbox" [13], J. Makowsky questions the suitability of the standard logic syllabus to the needs of CS practitioners. He states: "The current syllabus is often justified more by the traditional narrative than by the practitioner's needs." He further notes that most classical logic textbooks follow the narrative of the rise and fall of Hilbert's program, emphasizing the following ideas: (i) Logic is needed to resolve the paradoxes of set theory; (ii) First-order logic (FOL) is The logic due to its Completeness theorem; (iii) The main theorems of FOL are the Completeness and Compactness theorems; (iv) The tautologies of FOL are not recursive; (v) One cannot prove consistency within rich enough systems. This, according to Makowsky, is *not* what a CS practitioner needs: "The proof of the Completeness Theorem is a waste of time at the expense of teaching more the important skills of understanding the manipulation and meaning of formulas." What he needs is: (i) understand the meaning and implications of modeling the environment as precise mathematical objects and relations; (ii) understand and be able to distinguish intended properties of this modeling and side-effects; (iii) be able to discern different level of abstraction, and (iv) understand what it means to prove properties of modeled objects.

In her papers [20, 19], J. Wing stresses the importance of integrating formal methods into the existing CS curriculum by teaching their common conceptual elements, including state machines, invariants, abstraction, composition, induction, specification and verification. She states discrete mathematics and mathematical logic as crucial prerequisites.

The above proposals on *what* to teach are extremely relevant for IS practitioners. On the question of *how* to teach, the paper "Integrating Formal Methods into Computer Science Curricula at a University of Applied Science" ([17]) of Tavolato and Vogt offers some useful insights. It discusses teaching formal methods at universities of applied sciences, where there are usually limiting factors which are relevant to the IS context as well: (i) students have very limited theoretical background, and (ii) they are strongly focused on the direct applicability of what they are taught. In this context the authors stress the importance of making the practical applicability of the theory understandable to students, and making use of real industry-inspired examples.
In what follows, we extend and adapt the above proposals to the context of IS, and provide our vision on aligning the teaching of logic to the needs of IS practitioners.

2.2 Our Vision: Making Logic Relevant for IS

Logic is a prerequisite for understanding and successfully using formal methods, which in their turn can significantly contribute to software quality control. We agree with [20] that the main basic formal conceptual elements that the students need to be familiar with include state machines, abstraction, composition, induction, invariants, specification and verification. While the students encounter the concepts of state machines, abstraction and composition at other IS courses (such as modeling and design), aspects related to working with formal specifications are not covered elsewhere. This leads to the following practical needs of an IS practitioner: (1) read, write and understand formal specifications, (2) be able to formalize informal specifications, (3) analyze specifications and detect sources of incompleteness, inconsistency and complexity, (4) reason about specifications, and (5) check a system against a specification.

Based on the above, adapting and extending the previous proposals to the context of IS, we arrive at the following IS logic toolbox: (a) Basic principles for reasoning about sets; (b) Induction and invariants; (c) Propositional and first-order logic and their axiomatizations; (d) Formal specification and verification.

As to *how* to teach logic to IS students, i.e., designing concrete teaching methodologies, the following considerations need to be taken into account:

- *Examples from software domains are useful.* Although it has been believed for some time that studying logic improves software development skills, this common belief has recently been empirically validated. In a three-year study in the framework of the Beseme project ([14]), empirical data on the achievements of two student populations was collected: those who studied discrete mathematics (including logic) through examples focused on reasoning about software, and those who studied the same subject illustrated with more traditional examples. An analysis of the data revealed significant differences in the programming effectiveness of these two populations in favor of the former. As pointed out by [17], software related examples are also useful for increasing the motivation of students, who can see the applications of the studied material in the domain of their interest.
- *Cognitive difficulties should not be ignored.* Empirical studies show that the use of formal methods poses objective difficulties for practitioners ([5, 9]). They are also hypothesized to be a major hindering factor for the acceptance of formal methods in industry ([21]). Although the cognitive processes of students when studying logic and formal methods are not well understood, they should not be ignored ([17]). Numerous studies in the education community addressed the gap between the students' intuition and formal thinking in mathematics (see, e.g., [7]). Implementing similar ideas in the domain of teaching logic and formal methods may help deal with these barriers.
- *Intricate complexities are not always needed.* Exposing the students to full intricate complexities of mathematical logic (such as a full proof of the completeness theorem, or dealing with variables not free for substitution) has the potential to confuse novices struggling to understand new ideas. However, most of the practitioners will not encounter them in industry. This is in line the research agenda of indirect application of formal methods ([11]), calling for hiding the intricate complexities behind automatic tools with intuitive user interface. The benefits of hiding logical complexity behind the more intuitive interface of functional programming are also mentioned in [14].

In view of the above considerations, the basic principles in the design of the course described below have been (i) use mainly examples from software domains, (ii) use comprehensible examples, and (iii) introduce the logical concepts at a basic level. The issue of

comprehensibility also led to our ongoing work of automatically measuring comprehensibility of Z specifications, which we briefly describe below.

3 Teaching Logic at the IS Department of the University of Haifa

In this section we demonstrate how the vision presented in Section 2.2 has been implemented in our design of the course "Logic and Formal Specification". The course has been taught at the IS department at the University of Haifa for several years by both of the authors[3]. The course is a mandatory course for graduate students, and its length is one semester, 4 hours per week.

3.1 Course Description

Below we provide a short description of the course's main topics, which are divided into two main parts:

Part I: Introduction to Logic

- *Informal laws of mathematical reasoning*
  Our starting point is the place where the students left off in a discrete mathematics course: with basic set-theoretical concepts. However, our primary focus is not on understanding the concepts themselves, but on *reasoning* about them by applying informal logical laws. Accordingly, the students are asked to provide proofs of basic claims, explaining which laws were used at each stage. The presentation of the informal laws and other proof tips is adapted from [12]. The informal laws become explicit at the object level when classical propositional and first-order logic are introduced to the students (E.g., the law for proving general statements can be captured by the rule inferring $\forall x\psi$ from $\psi(x)$, and the law for proving conditional statements is captured by the deduction theorem.)
- *Induction:* mathematical, structural and computational induction.
  Induction is in the heart of several formal concepts relevant for verification and validation of software: fixed point constructions, model checking, program analysis and many more. Therefore a special emphasis is put on the topic throughout the course, highlighting its various manifestations, e.g. proving syntactic properties of logical formulas, proving the deduction theorem, proving invariants with respect to a Z specification. Software-related examples are adapted from Chapter 2 of ([3]).
- *Classical Propositional and First-Order Logic:* syntax and semantics, satisfiability and validity, Hilbert-style axiomatization, formalization of natural language sentences.
  For this part of the course we mostly adapt parts of the standard presentation of most mathematical logic textbooks. We make a special emphasis on formalization of natural language specifications and induction at the expense of omitting the proofs of the Completeness and Compactness theorems (in line with the recommendation of [13]).

[3] Perhaps it is important to mention here the authors' relevant background. The first author is an associate professor at the Information Systems Department at the University of Haifa with active research interests in applied logic. The second author is the manager of the Software Performance and Quality research group at the IBM Haifa Research Laboratory, and a member of the IBM corporate Board of Software Quality. Both of the authors have several years of experience in teaching logic and formal methods to various audiences of students.

- *Survey of non-classical logics*[4]*:* temporal logic, modal logic, many-valued logic, fuzzy logic, non-monotonic logic, paraconsistent logic.

Part II: Introduction to Formal Specification

This part of the course builds up on the knowledge obtained at the previous part. The final aim is for the students to be able to understand and write formal specifications using the Z notation. For this we have adapted the material from the textbook [15], covering the basic aspects of Z: types, schemas and reasoning about Z specifications. However, staying faithful to our vision oulined above, we have developed our own set of examples, which are (i) "simple" and (ii) related to software domains. In what follows we shortly discuss what we mean by "simple" and how "simplicity" can be measured.

Measuring comprehensibility of Z specifications

The notion of simplicity (item (i) above) is not well understood. Comprehensibility (or understandability) of specifications is usually thought of as the degree to which information contained in a specification is understandable to the reader, and this is a well-studied topic in software engineering (see, e.g., [6] for a survey). However, we are aware of only a few works on Z specifications ([8, 10]), all of which on the structural dimension. We believe, however, that simplicity is a key to comprehensibility of specifications, at least at the stage of learning the topic. Our attempt to quantitatively measure "simplicity has led to our ongoing project of developing automatic tools for this purpose. Our current hypothesis is that the *nesting* of definitions is and shortening notations by *introducing additional symbols* decrease the understandability of specifications. We are currently developing a tool[5] for measuring "simplicity" of specifications. Using this tool, we plan to empirically check our hypothesis, as well as to consider other comprehensibility dimensions.

Students' Acceptance

The course has only been taught in its current form for five years, so making decisive conclusions about its effectiveness is perhaps premature. However, an important dimension in evaluating such effectiveness is the students' acceptance and reaction. To gain a better understanding of these factors, the first author has undertaken a preliminary qualitative study using a questionnaire filled by twenty three students who took the course in 2013-2014.

It should be noted that the limiting factors typical of our target audience are in many aspects similar to those described in [17]. The first is *lack of mathematical background*: the undergraduate IS study program at the University of Haifa does not include a course in logic, and the majority of students have only a background in discrete mathematics, where they are taught very basic concepts of set theory. The second limiting factor is their *lack of motivation*: the majority of the students return to graduate school several years after receiving their B.A, while working full-time. They typically expect the topics to be directly

[4] This part of the course is implemented by assigning each of the students a short presentation on a non-classical logic or its applications of his choice. While the importance of temporal logic in this context is perhaps the most obvious one due to its well-known applications in verification, also other non-classical logics have IS-relevant applications. Our goal here is to increase the awareness of the students to the immense variety of logics outside the realm of classical logics, as well as engage them more actively in the course. Several students have reported that exploring new logics on their own was the part they enjoyed the most in the course.

[5] The tool is based on the open-source Java framework Community Z tools ([2]).

relevant to their IS practice, and usually exhibit difficulty in coping with the dense and abstract material taught in the course. In light of these factors, we were expecting some of the students to claim, basically, that the course was too hard without being helpful for their future as IS practitioners. However, only one student out of 23 felt the course was not useful for his practice. A full analysis of the data obtained from students is out of the scope of this paper. However, some aspects highlighted by the data were that (i) the majority of students felt the course has improved their analytical thinking abilities[6]; (ii) the majority of students felt the part related to formal specification is relevant for their IS practice[7]; and (iii) some students felt the course has *directly* improved their daily IS practice[8].

4 Summary and Future Research

There has recently been a discourse on the relevance of traditional logic courses to future computer science practitioners. Jeannette Wing writes in [19]: "...we still face the educational challenge of teaching mathematical foundations like logic and discrete mathematics to practicing or aspiring software engineers. *We need to go beyond giving the traditional courses and think about who the target students are.*" This paper discusses these issues for the target population of IS students, for whom the lack of direct relevance of the traditional logic courses seems to have led to their exclusion from the curriculum. We believe logic is central to IS objectives, as it is the key to applying formal methods in specification, verification and validation of information systems. Ideally, we need more empirical evidence in the spirit of the Beseme project ([1]) that such courses are useful for IS practitioners. In addidion, there is a need for a wider discussion on *what* logical background is needed for Information Systems practitioners and *how* it should be taught. In a contribution to such discussion, we have reported on our insights from teaching the "Logic and Formal Specification" course to graduate IS students. Like previous authors report in the context of CS, we have seen that using software-related and comprehensible examples, as well as simplification of logical intricacies contributes to achieving the courses' objectives. From a more practical perspective, a future direction is an empirical investigation of *how* to make formal specification more understandable for students. This question is particularly interesting due to its direct relation to the more general topic of comprehensibility of specifications. In this context we plan to extend and refine our tool for automatic analysis of Z specifications and carry out an empirical evaluation. From the angle of education, strategies for an efficient integration of logic and formal methods into the IS curricula are required (along the lines of [4, 20]), as well as an investigation of the ways to bridge intuitive and analytical thinking processes in logic and formal methods (along the lines of [7]). In this context we would also like to point out that a textbook with an IS-orientation would be a welcome addition to the large existing variety of CS-oriented books.

References

1 Beseme website, http://www.cs.ou.edu/ beseme/.

[6] Examples: "I don't use logic or Z on a daily basis in my research or my work, but it improved my modelling skills", "I think it can be even more useful as an introduction to programming, because it teaches you to think systematically", "Every graduate student needs this course as a basis for study and research".

[7] Examples: "I think Z language is the most useful part, because it is applied in industry", "If I encounter any other formal notation in industry, it will take me less time to get familiar with the subject".

[8] Examples: "I started using truth tables at work to rule out impossible behaviors", "I was surprised to find out how helpful the tools we obtained are in my daily work".

2 CZT website, http://czt.sourceforge.net/manual.html.

3 Alfred V Aho and Jeffrey D Ullman. *Foundations of computer science*, volume 2. Computer Science Press New York, 1992.

4 Ian Barland, Matthias Felleisen, Kathi Fisler, Phokion Kolaitis, and Moshe Y Vardi. Integrating logic into the computer science curriculum. *Innovation and Technology in Computer Science Education*, 2000.

5 Deirdre Carew, Chris Exton, and Jim Buckley. An empirical investigation of the comprehensibility of requirements specifications. In *Empirical Software Engineering, 2005. 2005 International Symposium on*, pages 10–pp. IEEE, 2005.

6 Nelly Condori-Fernández, Maya Daneva, Klaas Sikkel, and Andrea Herrmann. Practical relevance of experiments in comprehensibility of requirements specifications. In *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on*, pages 21–28. IEEE, 2011.

7 Lisser Rye Ejersbo, Uri Leron, and Abraham Arcavi. Bridging intuitive and analytical thinking: Four looks at the 2-glass puzzle. *For the Learning of Mathematics*, 2014.

8 K Finney, N Fenton, and A Fedorec. Effects of structure on the comprehensibility of formal specifications. In *Software, IEE Proceedings-*, volume 146, pages 193–202. IET, 1999.

9 Kate Finney. Mathematical notation in formal specification: Too difficult for the masses? *Software Engineering, IEEE Transactions on*, 22(2):158–159, 1996.

10 Kate Finney, Keith Rennolls, and Alex Fedorec. Measuring the comprehensibility of Z specifications. *Journal of Systems and Software*, 42(1):3–15, 1998.

11 Heinrich Hussmann. Indirect use of formal methods in software engineering. In *ICSE-17 Workshop on Formal Methods Application in Software Engineering Practice, Seattle (WA), USA. Proceedings*, pages 126–133. Citeseer, 1995.

12 David Makinson. *Sets, logic and maths for computing*. Springer, 2012.

13 Johann A Makowsky. From Hilbert's program to a logic tool box. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):225–250, 2008.

14 Rex L Page. Software is discrete mathematics. In *ACM SIGPLAN Notices*, volume 38, pages 79–86. ACM, 2003.

15 Ben Potter, David Till, and Jane Sinclair. *An introduction to formal specification and Z*. Prentice Hall PTR, 1996.

16 Mehran Sahami, Mark Guzdial, Andrew McGettrick, and Steve Roach. Setting the stage for computing curricula 2013: computer science–report from the acm/ieee-cs joint task force. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 161–162. ACM, 2011.

17 Paul Tavolato and Friedrich Vogt. Integrating formal methods into computer science curricula at a university of applied sciences. In *TLA+ Workshop at the 18th International Symposium on Formal Methods, Paris, Frankreich.*, 2012.

18 Heikki Topi, Joseph S Valacich, Ryan T Wright, Kate Kaiser, Jay F Nunamaker Jr, Janice C Sipior, and Gert-Jan de Vreede. Is 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems*, 26(1):18, 2010.

19 Jeannette M. Wing. Teaching mathematics to software engineers. In *Algebraic Methodology and Software Technology, 4th International Conference, AMAST '95, Montreal, Canada, July 3-7, 1995, Proceedings*, pages 18–40, 1995.

20 Jeannette M Wing. Weaving formal methods into the undergraduate computer science curriculum. In *Algebraic Methodology and Software Technology*, pages 2–7. Springer, 2000.

21 Marc K Zimmerman, Kristina Lundqvist, and Nancy Leveson. Investigating the readability of state-based formal requirements specification languages. In *Proceedings of the 24th International Conference on Software engineering*, pages 33–43. ACM, 2002.

Using interrogative logic to teach classical logic*

Levis Zerpa[1]

1 **Department of Social Science and Innovations, Yachay Tech**
Hacienda San José y Proyecto Yachay, Urcuquí, Ecuador
lzerpa@yachaytech.edu.ec

Abstract

In the paper I discuss a tool for helping students in their symbolizations of natural language sentences using the formal language of classical first order logic (CFOL). The tool is an extension of Hintikka's concept of (Inquirer's) range of attention in the context of interrogative games. Any given text is reconstructed as the answer to a "big" or principal question obtained through the answers of a series of "small" or operative questions. The tool brings some "narrative flavor" to the symbolization and offers a convenient mold that can be used by students in many different contexts.

1998 ACM Subject Classification "F.4.1 Knowledge Representation Formalisms and Methods"

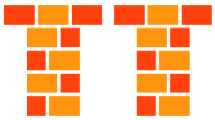
Keywords and phrases erotetic logic, interrogative logic, interrogative games, interrogative model of inquiry, classical first-order logic

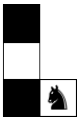
1 Introduction

Learning to symbolize natural language sentences in the formal language of classical first order logic (CFOL) is one of the main tasks of most logic courses. The concept of (Inquirer's) range of attention RA (Hintikka & Hintikka [6], Genot & Gulz [3], Zerpa [8]), from Hintikka's interrogative logic, is a major tool to carry out that task. According to Hintikka [5], RA is the set of available tautological premises of the form $S \vee \neg S$. Intuitively speaking, the set RA codifies the totality of yes-or-no questions which the Inquirer is prepared to ask. If one considers an extension of the concept of range of attention according to which wh-questions, besides yes-or-no questions, are considered, then one is able to apply the RA concept as a tool to teach students how to use the expressions of the formal language of CFOL to symbolize natural language sentences. These expressions include sentence letters, truth-functional connectives, individual constants and variables, predicates, and quantifiers. The tool can be conceived as a procedure consisting of the following steps: first, reconstruct the text as the answer to a "big" or principal question obtained through the answers of a series of "small" or operative questions. Second, obtain the presupposition of each question dropping the question mark on it. Third, make the symbolization of all the previous questions and answers using the abovementioned expressions of the formal language of CFOL in the usual way. Fourth, make all the deductive inferences required. Fifth, organize the information previously obtained in a tableau like the ones described in Hintikka [5]. Yes-or-no questions, read as "Is it the case that S or $\neg S$?", have the logical form "$S \vee \neg S$?" while which-questions, read as "Which individual x is such that $S(x)$?", have the form "$\exists x S(x)$?". The presupposition of the yes-or-no question "$S \vee \neg S$?" is the tautology $S \vee \neg S$ while the presupposition of the which-question "$\exists x S(x)$?" is the sentence $\exists x S(x)$. The text under consideration may have a

* This work is part of the project on the interrogative model of inquiry and education currently under development at Yachay Tech.

4th International Conference on Tools for Teaching Logic.
Editors: M. Antonia Huertas, João Marcos, María Manzano, Sophie Pinchinat, François Schwarzentruber; pp. 281–286





Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

specific deductive structure or not; the point is that the strategy of break down the initial principal questions in a set of several operative questions of the forms specified brings a "narrative flavor" to the symbolization and, in that sense, constitutes a convenient mold that can be used by students in many different contexts. The logical justification of the procedure is found in theorem 7.49 of Wiśniewski [7] using multiple-conclusion entailment.

In the paper I report an ongoing research project related to the application of interrogative logic and the interrogative model of inquiry and learning to the teaching of elementary classical logic in introductory university courses. The empirical basis of the project is obtained from logic courses taught at Yachay Tech (Ecuador) during 2014-2015. In spite of the fact that the project is concerned with several aspects of the interrogative approach, in this paper I focus the attention on both the interrogative approach to the teaching of the material implication and the concept of range of attention as a tool to symbolize natural language sentences in CFOL.

The paper is organized as follows: first, some general remarks are made about interrogative logic and the interrogative model of inquiry and learning. Then, the interrogative approach to the teaching of material implication are briefly considered using the experience in Yachay Tech. Finally, the largest part of the paper is focused on the RA concept through the analysis of two concrete examples, one from everyday reasoning and other from scientific (experimental) reasoning.

2 Interrogative logic and the interrogative model of inquiry and learning

Interrogative or erotetic logic is the logical theory of questions and answers. More specifically, interrogative logic studies formal systems in which interrogative as well as declarative sentences are formalized with precision. Following Frege [2], questions are understood as requests for information and these requests are studied, in a detailed way, in these formal systems. Furthermore, it is known that questioning is a fundamental activity in knowledge-acquisition; it plays a crucial role in both scientific research and the learning process. This idea leads toward the interrogative approach to inquiry and learning: both inquiry and learning are processes which can be modeled by means of sequences of questions and answers. More specifically, progress in inquiry can be represented through a study of the transformation of the successive question asked by these agents and the formulation of answers to them. According to Hintikka´s Interrogative Model of Inquiry the process of inquiry is modeled as a two-person game between an inquirer and a source of information. The goal of the game is to get a conclusive answer to the principal (or "big") question posed by the inquirer through the formulation of more specific (or "smaller") questions called operative questions. The moves of the game are of two types: interrogative and deductive. Interrogative moves consist of questions or answers while deductive moves are deductive inferences obtained from previous moves. The score of the game is kept in semantic tableaus (Hintikka [5]). The Interrogative Model of Learning is an application of the interrogative model of inquiry to the study of educational practices emphasizing the importance of the transformation of questions through the learning process (Hakkarainen & Sintonen [4]).

3 Teaching material implication through questioning

Traditionally, material implication is a perplexing topic for many students at the very beginning of their learning process. The truth table of the material implication, specially

the cases in which the antecedent are false (the FV and FF cases in which, according to material implication, the resulting conditional is true), are usually a source of confusion and perplexity for many students. After a study of negation, conjunction and disjunction, students were exposed with four examples of conditional statements of a very diverse nature, and the principal question asked to them was "when all these conditional are false?". Students provided two responses: (1) the conditional is false when VF is the case and (2) when FF is the case. On the basis of an ilustrative example, operative questions, answers to them, and further information, they discarded (2) in favor of (1). Furthermore, the symbolization of arguments in natural language containing negation, conjunction, disjunction and material implication were guided by the processing of questions. Given an argumentative text, the principal question, "which is the correct symbolic translation of this text?", was transformed into a first "intermediate" operative question: "which are the indicator words in the text?". This last questions was transformed into three "smaller" (= more specific) operative questions: "which are the conclusion indicators?", "which are the premise indicators?", "which are the phrases, in the text, related to truth-functional connectives?". In turn, this last operative question was transformed into four more specific operative questions of the form "which phrase of the text can be translated with an X?" where X stands for a truth-functional connective (the biconditional was introduced shortly after). Even specific logical laws were directly approached from an interrogative viewpoint, For example, the De Morgan theorems were presented as the answer to the question "how one can distribute the negation in a formula of the form "$\neg(A \vee B)$". My hypothesis is this: the main logical laws can be taught as the correct answers to questions about inference. One can motivate students to ask these questions so they can discover, by themselves, these laws as their answers to their own questions. A major tool in that process is the transformation of their questions. This approach is closely related to Bereiter [1] problem-solving approach to learning. A precise framework to study the transformation of questions is the abovementioned concept of range of attention. In this framework students are able to use the expressions of the formal language of CFOL to symbolize texts in natural language. Students start by consider the available questions in the RA set. Then, they organize them in such a way that the structure of the text is conceived as the answer to a principal question by means of a series of operative questions.

4 The RA set and its use in the teaching of predicate logic (1): Holmes' reasoning

Arthur Conan Doyle's story Silver Blaze can be summarized in the following way: the horse Silver Blaze has been stolen and its owner found dead. To solve the case, Holmes must find out who stole the horse. During the investigation, the following exchange takes place (Hintikka [5], Genot & Gulz [3]):

Watson: Is there any point to which you would wish to draw my attention?

Holmes: To the curious incident of the dog in the night-time.

Watson: The dog did nothing in the night-time.

Holmes: That was the curious incident.

There is a remarkable step of reasoning here, namely, the identification of the violation of a natural expectation: a dog would have barked on a stranger. But it did not. So, the thief was known by the dog. Furthermore, it is determined that the only relevant individuals are the following: the unknown thief, the dead stable-master (the owner), and his killer. Therefore, from the reduced number of individuals and the curious incident of the dog in the

night-time, Holmes correctly infers that the thief is the owner. This piece of reasoning is reconstructed by means of the concept of Inquirer's range of attention, taking Holmes as the Inquirer. According to the intuitive interpretation of the set RA, it codifies the totality of yes-or-no questions which Holmes is prepared to ask. In this case, a small proper subset RA_H of RA is enough. Following a sequential order, the two main (operative) yes-or-no questions in RA_H are the following: "Is there a dog in the stable or not?" (answer: yes) and "did the dog barked at the thief or not?" (answer: no). According to the extension of the RA concept I am proposing, in RA one is able to ask wh-questions besides yes-or-no questions. Therefore, the principal question "who is the thief?" can be included in RA_H. Consequently, if constants o and t stand for the owner and the thief, respectively, the tableau of the game starts with the principal question "who is the thief?" or "$\exists(x = t)$?" and ends with the conclusive answer to this question, namely, "the thief is the owner" or "$t = o$". Holmes' reasoning is reconstructed as an interrogative game which is played through a series of yes-no questions from RA_H, their answers, and deductive inferences from them. In general, a yes-no question of the form "$S \vee \neg S$?" has presupposition $S \vee \neg S$, so the question only presupposes the truth of that tautology. And a known theorem of interrogative logic (see Hintikka [5]) guarantees that a question like this always can be properly asked (under the appropriate conditions). The sequence of questions and answers are the following (all the interrogative moves are carried out by the inquirer, Holmes, and the answers are provided by the source of information):

(1) **Interrogative move**: *Is there a dog in the stable or not?* In symbols,

"$\exists x$[dog_in _the _stables(x)] $\vee\neg\exists x$[dog_in _the _stables(x)]?"

(2) **Answer**: *yes*; in symbols, $\exists x$[dog_in _the _stables(x)].

(3) **Deductive inference move**: dog_in _the _stables(d) by existential instantiation (d is a new constant). This step is similar to say "let's call d the dog in the story".

(4) **Interrogative move**: *Did d barked at the thief or not?* In symbols, d barked at $t \vee \neg(d$ barked at $t)$?

(5) **Answer**: *no*; in symbols, $\neg(d$ barked at $t)$.

(6) **Interrogative move (non-triviality of Holmes' reasoning)**: Holmes then asks whether the dog did not bark to a specific person or not. In other words, the dog did not bark to the thief but it would have barked to anybody else. This question rules out the possibility that the dog just did not bark to anybody (for example, if the dog was sent to sleep by somebody), so this move avoids the trivialization of Holmes' reasoning. In a semi-symbolic way, the question is this: "is the individual to whom the dog didn't bark unique or not?". In symbols, $\exists z\forall y[\neg(d$ barks at $y) \rightarrow y = z] \vee \neg\exists z\forall y[\neg(d$ barks at $y) \rightarrow y = z]$?

(7) **Answer**: *yes*; in symbols, $\exists z\forall y[\neg(d$ barks at $y) \rightarrow y = z]$.

(8) **Deductive inference move**: By existential instantiation, $(d$ barks at $t) \rightarrow t = o$.

(9) **Deductive inference move**: By modus ponens from answer (5) and deductive inference move (8), the case is solved: $t = o$, the thief is the owner.

In the corresponding tableau, the presupposition of any question must be established before that question can asked. A nice feature of Hintikka's theory is that the presupposition of any question is obtained simply by dropping the question mark on it. The traffic from side to side of the tableau also has restrictions. Some abbreviations are used: "$D(x)$" abbreviates "*dog_in_the_stables*(x)", "$B(x,t)$" abbreviates "x barked at t", and "*unique*" abbreviates "'$\exists z\forall y[\neg(d$ barks at $y) \rightarrow y = z]$". In summary, the tableau looks like this:

In this case, the basic steps of reasoning were obtained through yes-no questions. Only the principal question is a wh-question, namely, "who is the thief?". But in more complex cases, wh-questions are needed through all the game. Mendel's research on genetics can also be

#	SOURCE OF INFORMATION	INQUIRER: HOLMES	#
1	$\exists x(x = t)$	$\exists x(x = t)?$	2
3	$\exists x D(x) \vee \neg\exists x D(x)$	$\exists x D(x) \vee \neg\exists x D(x)?$	4
5	$\exists x D(x)$	$\exists x D(x)?$	6
7	$D(d)$		8
9	$B(d,t) \vee \neg B(d,t)$	$B(d,t) \vee \neg B(d,t)?$	10
11	$\neg B(d,t)$		12
13	$unique \vee \neg unique$	$unique \vee \neg unique?$	14
15	$unique$		16
17	$\neg B(d,y) \rightarrow t = o$		18
19	$t = o$ (case solved!)		20

modeled as an interrogative game. The complete tableau is more complex than the previous one; see Zerpa [8] for details. The next subsection has a small subtableau representing the results of the first round of Mendel's experiments on peas (*pisum sativum*).

5 The RA set and its use in the teaching of predicate logic (2): Mendel's experimental results

Mendel´s principal question on the subgame considered is this: "Which are the results of the first experiment?". To answer this principal question, Mendel asks to the source of information several operative questions that provide the answer to that principal question. In this case, the members of the set RA include which-questions about the number of plants crossed and the form and number of differentiating character selected. Mendel´s most innovative questions are (*) and (**). In the dialogical framework of the game, these which-questions are the following:

Mendel: *Which is the number of plants crossed?*

Source of information: *253 plants.*

Mendel: *Which is the (constantly) differentiating character selected?*

Source of information: *form of the seed.*

Mendel: *Which character is dominant?*

Source of information: *round (seed).*

Mendel: *Which character is recessive?*

Source of information: *wrinkled (seed).*

Mendel: *Which are the numbers of different forms under which the progeny appeared in the first experiment?*

[This question is transformed into two operative questions:]

Mendel: *Which is the number of plants showing the dominant character? (*)*

Source of information: *5,474 plants showed round seed.*

Mendel: *Which is the number of plants showing the recessive character? (*)*

Source of information: *1,850 plants showed wrinkled seed.*

Mendel: *Which are the ratios found between the hybrid forms in the first experiment? (**)*

Source of information: *A ratio of 2.96 : 1 between dominant and recessive characters appeared among the progeny in the first experiment.*

The entire tableau is large; to get an idea of it is convenient to consider the questions concerning the number of plants showing the dominant and recessive characters. Let's introduce the following constants and predicates: D1(x): x is a plant which shows dominant character D1 (in experiment 1); D1 = round seed, R1(x): x is a plant which shows recessive character R1 (in experiment 1); R1 = wrinkled seed N(x, y): x is the number assigned to y (after counting them.

Then, this part of the tableau looks like this:

#	SOURCE OF INFORMATION	INQUIRER: MENDEL	#
...	...	...	...
1	$\exists x \exists y[D_1(x) \wedge N(y,x)]$	$\exists x \exists y[D_1(x) \wedge N(y,x)]?(*)$	2
3	$D_1(a) \wedge N(5,474,a)$		4
5	$\exists x \exists y[R_1(x) \wedge N(y,x)]$	$\exists x \exists y[R_1(x) \wedge N(y,x)]?(*)$	6
7	$R_1(a) \wedge N(1,850,b)$		8
...	...	...	...

6 Conclusion

In conclusion, the concept of RA provides a precise framework to teach symbolization of texts in natural language through a specification of the transformation of a principal question in a series of operational questions. The resulting analysis keeps a certain "narrative flavor" that is sometimes absent from the usual teaching approaches. Furthermore, the interrogative model of learning provides an interesting way to motivates active learning through a study of the transformation of questions in logic courses.

Acknowledgements My thanks go to two anonymous referees for their helpful comments on the earlier version of this paper.

References

1 Carl Bereiter. *Problem-Centered and Referent-Centered Knowledge: Elements of Educational Epistemology*. Interchange 23(4), 337–361, 1992.
2 Glottob Frege. *The Thought: A Logical Inquiry*. Mind, 65(259), 289-311. (Original work published in German 1918-1919).
3 Emmanuel J. Genot and Agneta Gulz. *Madness in the Method: A Paradox of Inquiry Learning*. Working Paper, Lund University Knowledge and Information Quality Research Group, 2014.
4 Kai Hakkarainen and Matti Sintonen *The Interrogative Model of Inquiry and Computer-Supported Collaborative Learning*. Science & Education vol. 11, pp. 25-43, 2002.
5 Jaakko Hintikka. *Inquiry as Inquiry: A Logic of Scientific Discovery*. Dordrecht: Kluwer Academic Publishers, 1999.
6 Jaakko Hintikka and Merrill Hintikka. *The Logic of Epistemology and the Epistemology of Logic*. Kluwer, Dordrecht, Netherlands, 1989.
7 Andrzej Wiśniewski. *The Posing of Questions: Logical Foundations of Erotetic Inferences*. Dordrecht, Springer, 1995.
8 Levis Zerpa. *Mendel's interrogative game*. Forthcoming, 2015.