An interactive tool for manipulating logical formulae

Josje Lodder

Johan Jeuring

Harrie Passier

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-040

www.cs.uu.nl

ISSN: 0924-3275

An interactive tool for manipulating logical formulae

Josje Lodder, Johan Jeuring, Harrie Passier
Open University
Heerlen, the Netherlands

July 31, 2006

1 Introduction

Logic is constructive in nature, and in a course on logic a student learns how to manipulate logical formulas. For example, a student has to learn how to simplify a logical formula, how to transform a logical formula into disjunctive normal form (DNF), and how to prove equivalences of logical formulae.

Solving logical exercises is often done with pen and paper, but e-learning tools offer great possibilities. In particular for a distance learning university such as the Dutch Open University it is important to support the interactive construction of solutions to logical exercises. Currently all exercises and solutions can be found in our lecture notes for the courses that teach logic. Although all exercises are worked out in detail, it is impossible to list all the possible answers to for example normalization exercises and all the different correct ways to reach an answer. Furthermore, with all the answers available, a student may be tempted to look at an answer, instead of constructing it herself. With an interactive electronic tool the student can construct and check her own answer.

Feedback is essential for effective learning and hence crucial for an interactive e-learning tool such as an interactive tool for solving logical exercises. E-learning tools provide the capability to give semantically rich feedback to a student at each step. As far as we know only a few of the existing tools supporting the interactive construction of solutions to logical exercises give feedback to a student at each step in the solution process, but in all of these tools either the process can be made more flexible by not enforcing particular rules, or the feedback given can be much more precise.

This paper describes an interactive e-learning tool for solving logical exercises, that gives semantically rich feedback to users. The tool supports the stepwise construction of a solution to an exercise, and provides feedback at each step.

This paper is structured as follows. Section 2 introduces our tool for solving logical exercises. Section 3 briefly describes the kind of feedback given by our tool. Section 4 describes related work, and our plans for the tool.

2 An interactive tool for manipulating logical formulae

The logical exercise solver helps students learn to rewrite formulae from propositional and predicate logic using a set of standard equivalences. Some examples of exercises the student can try to solve are:

- Translate the formula $(\neg p \rightarrow q) \rightarrow (r \leftrightarrow \neg s)$ into DNF.
- Prove the equivalence: $(\phi \leftrightarrow \psi) \Leftrightarrow (\neg\phi \leftrightarrow \neg\psi)$.
- Prove the implication $\pi \wedge \theta, \pi \rightarrow \rho \Rightarrow \theta \wedge \rho$.
- Translate into a prenex normal form $\exists x.P(x) \rightarrow \neg\forall y.Q(y)$.
- Prove $\exists x.\phi \rightarrow \psi \Leftrightarrow \forall x.(\phi \rightarrow \psi)$, for x not free in ψ.

Using our tool, students can solve these exercises as they would using pen and paper. Provided a student applies a single rule at a time, the tool checks that an expression submitted by a student can be derived from the previous expression by applying one of the rules for logical expressions. Furthermore, the tool gives detailed feedback in case of an error.

Figure 1 shows a screenshot of our tool. The logical exercise solver consists of three text fields. The top text field is the working area, in which a student can edit a logical formula stepwise towards a solution. The current formula is

$$(\neg p \vee \neg q) \vee (r \leftrightarrow \neg s)$$

The second text field displays the history of equations. Apparently the previous system of equations was

$$\neg(p \vee q) \vee (r \leftrightarrow \neg s)$$

and the student replaced $\neg(p \vee q)$ by $\neg p \vee \neg q$, forgetting to change the $\vee$ into a $\wedge$. The third text field displays the feedback. In the figure it explains why the last step is incorrect.

The logical exercise solver presents a logical expression to a student, and the student has to rewrite this expression into a proper form (disjunctive normal form for the given example). The student presses the Submit button to submit an edited logical expression, and the Undo button to undo the last step (or any amount of steps: the history keeps all steps). If a student wants help, she presses the Hint button to get a suggestion about how to proceed.

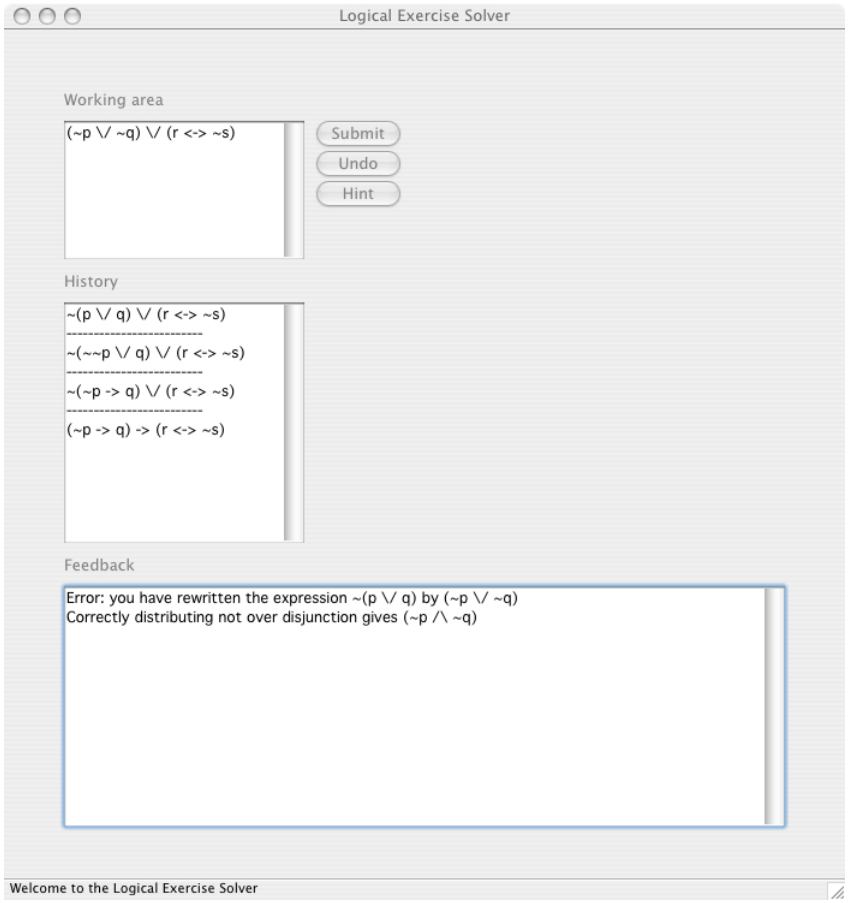

Figure 1: The interactive logical exercise solver

3 Feedback

The logical exercises solver helps a student in practicing how to:

- choose a rule that is applicable to the current formula.
- choose a rule that is a step towards an answer.
- correctly apply a rule.

A student can make different kinds of mistakes when solving an exercise:

- syntactic mistakes, such as for example a missing parenthesis.
- semantic mistakes, such as:
  - applying a "buggy rule" [5], such as for example distributing a universal quantifier over a disjunction.
  - applying a rule incorrectly such as for example forgetting to change a disjunction in a conjunction when applying De Morgans rule.
  - applying a non-applicable rule, such as for example using Modus Ponens when the premise of the implication doesn't match.

Our tool gives feedback about each of these kinds of mistakes. Feedback on the level of rewriting steps is the distinguishing feature of our tool. Feedback is not specified with an exercise, but instead derived by the tool for an arbitrary exercise in the domain of logical expressions. The tool consists of a description of the domain of logical expressions, a set of rewrite rules for transforming logical expressions, a set of buggy rules for logical expressions, one or more goals for the exercises, and a solver which checks the correctness of a submitted expression. These components are used to produce feedback.

- The tool has an error-correcting parser for logical expressions which detects as many syntactic errors as possible, and suggests corrections for these errors.
- The solver checks the correctness of a submitted expression. If the expression is correct, the tool tries to detect the applied rewrite rule and informs the student about this. If the expression is incorrect, the tool tries to find the cause using the sets of rewrite rules and buggy rules.
- The buggy rules represent common mistakes. The experience of teachers is used to collect this set of rules. If a student turns a correct expression into an incorrect expression, the tool checks whether or not the incorrect expression can be obtained from the correct expression by means of such a common mistake.
- If a mistake cannot be explained by a common mistake, it determines the closest correct expression to the expression entered by the student, under an edit-distance measure, using the set of rewrite rules for transforming logical expressions.

We have yet to experiment with our tool, but experience from a similar tool that supports solving systems of linear equations [9] makes us believe that our approach will explain most of the errors made by students correctly. Of course, it is impossible to explain all errors correctly; some students are surprisingly creative when it comes to making mistakes.

The main components of our tool and ideas are based on our tool for solving systems of linear equations [9]. As far as we know, deriving feedback from domain rules for arbitrary exercises in a particular domain has only been done in the domain of mathematical calculus [2] and in logic [10], and deriving feedback on the level of a single rewrite step without having to specify which rule is applied [9] is new. Both our tool for solving systems of linear equations and the tool discussed in this paper are derived from the same generic description of domain-specific exercise assistants that provide semantically rich feedback [8].

4 Related and future work

Related work. Many tools for teaching logic have been developed in the last two decades [3, 11]. Most of these tools concern the construction of proofs in a formal logic using for example natural deduction or semantic tableaus. In practice students also need other logical skills such as simplifying a condition for querying a database or rewriting a formula into normal form. For these purposes we teach our students to rewrite formulae given a set of equivalences like commutativity, absorption, distribution, double negation and De Morgan's rule. Some tools, like the Logic Workbench [4], perform these simplifications of formulae automatically. As far as we know there are only a couple of systems which have similar functionality to our tool: MLT-PC [7], LogicCoach 9X [10] and the Logic Tutor [1, 13]. We will describe these tools in more detail.

MLT-PC is a program in Catalan to teach students propositional logic. It covers most of the aspects of propositional logic, from formalization to actually proving with natural deduction. It contains modules for teaching students Boolean algebra and rewriting formulae in normal form. MLT-PC dictates which rule has to be applied: a user can not pick her own rule. Giving feedback is rather simple in this situation.

LogicCoach 9X deals with predicate logic as well as propositional logic. The proof system used is a kind of natural deduction based on well-known equivalences, together with some inference-rules like modus ponens and modus tollens. It provides many exercises ranging from very simple to rather complicated and it has a quite sophisticated feedback system, together with the possibility to ask for help. However, it does not contain modules for simplifying formulae or normal forms.

The Logic Tutor is comparable to LogicCoach: it offers exercises on proofs in the same natural deduction style as the LogicCoach (but only propositional logic) and it contains an elaborate mechanism for analyzing mistakes.

Both LogicCoach 9X and the Logic Tutor give feedback on the application of a rule *if* the user explicitly specifies which rule she wants to apply. If the

user doesn't explicitly specify a rule, LogicCoach determines the rule if the rule has been applied correctly. If the rule hasn't been applied correctly, or the user input contains syntactic errors, the feedback supplied isn't very helpful (in the style of "line x does NOT follow validly from the steps above it. So, there is NO combination of rules that will prove it", or "Nothing has been typed in the step editor").

An ideal electronic tool for teaching logic acts as an electronic teacher looking over the shoulder of the student. When necessary, this teacher gives feedback on the work of the student. Most of the existing tools for teaching logic require a student to select a rule to apply, and the tool then applies the rule (the result of applying a rule is often unique). The advantage of such an approach is that a student is forced to explicitly choose a rule. However, this is not the way most students work on pen and paper. LogicCoach 9X is an exception: it asks a student to motivate each step with the name of a rule, but if such a name is omitted, it tries to infer the rule applied. Our tool doesn't require the student to specify a rule. It tries to infer rules, and if it is impossible to infer a rule, it tries to find the closest correct expression obtained by applying a rule to the previous expression.

Future work. The implementation of the logical exercises solver hasn't been finished yet, we expect to finish a first version in the Summer of 2006. In the future we would like to work on several things:

- we would like to replace the ASCII presentation of logical expressions with a more advanced presentation, as for example in Proxima [12].
- we want to experiment with different kinds of feedback: not only direct comments on mistakes but also questions a student has to answer, references to theory, fully presented examples.
- we want to investigate if we can recognize the application of more than one rewrite rule.
- we want to investigate the possibility to differentiate between beginning and advanced students.
- we want to add 'progression indicators' to the tool, which inform students about progress towards a solution.
- and there are many other good ideas for tools for teaching mathematical logic [6].

References

[1] David Abraham, Liz Crawford, Leanna Lesta, Agathe Merceron, and Kalina Yacef. The logic tutor: A multimedia presentation. *IMEJ of Computer-Enhanced Learning*, 3(2), 2001. `http://imej.wfu.edu/articles/2001/2/03/`.

[2] Arjeh Cohen, Hans Cuypers, Dorina Jibetean, and Mark Spanbroek. Interactive learning and mathematical calculus. In *Mathematical Knowledge Management*, 2005.

[3] Hans van Ditmarsch. Logic software and logic education. `http://www.cs.otago.ac.nz/staffpriv/hans/logiccourseware.html`. These pages contain a comprehensive, alphabetically ordered list of educational logic software, 2005.

[4] Gerhard Jaeger et al. The logic workbench 1.1. `http://www.lwb.unibe.ch/`, 2002.

[5] Martin Hennecke. *Online Diagnose in intelligenten mathematischen Lehr-Lern-Systemen (in German)*. PhD thesis, 1999. Fortschritt-Berichte VDI Reihe 10, Informatik / Kommunikationstechnik; 605. Dsseldorf: VDI-Verlag.

[6] Manfred Kerber. Building a tool for teaching mathematical logic. `http://www.macs.hw.ac.uk/~fairouz/mkm-symposium03/abstracts/manfred.pdf`, 2003.

[7] Antonio Moreno and Neus Budesca. Mathematical logic tutor propositional calculus. In María Manzano, editor, *Proceedings of the First International Congress on Tools for Teaching Logic*, 2000.

[8] Harrie Passier and Johan Jeuring. Ontology based feedback generation in design-oriented e-learning systems. In P. Isaias, P. Kommers, and M. McPherson, editors, *Proceedings of the IADIS International conference, e-Society*, volume II, pages 992–996, 2004.

[9] Harrie Passier and Johan Jeuring. Feedback in an interactive equation solver. In *Proceedings of the Web Advanced Learning Conference and Exhibition, WebALT 2006.*, 2006.

[10] Nelson Pole. Logiccoach 9. `http://academic.csuohio.edu/polen/`, 2006.

[11] Ruy de Queiroz. Archive logic-l. `http://www.di.ufpe.br/~ruy/TOOLS/logic-l-software`. Logic programs and teaching aids, theorem provers and languages. Incomplete.

[12] Martijn M. Schrage. *Proxima – a presentation-oriented editor for structured documents*. PhD thesis, Utrecht University, The Netherlands, Oct 2004.

[13] Kalina Yacef. Experiment and evaluation results of the logic-ita. Technical Report 542, School of information Technologies, University of Sydney, 2003.