Yoda: a simple tool for natural deduction

Benjamín Machín and Luis Sierra

Instituto de Computación, Fac. de Ingeniería, Universidad de la República, Uruguay
{bmachin,sierra}@fing.edu.uy

Abstract. The course of logic offered by our Engineering School introduces the notion of formal proof following the Gentzen tree representation method. In this paper we present Yoda, a JavaScript application that ensures the correct construction and display of these proofs in web browsers.

1 Introduction

Logic is a core course for the Computer Engineering career in Universidad de la República, Uruguay, whose syllabus covers the topics required for the DS/BasicLogic outlined in [oCC01,For08]. The notion of formal proof is introduced following the Gentzen tree representation method ([vD94]). Since 2009 we offer an experimental course of Logic; while the standard course has five hundred students, the experimental one counts with forty students and is considered a place where innovations can be tested in a controlled way.

We are interested in some computer-based software that allow students to check autonomically their assignments. After an exploratory use of different tools and approaches we used ProofWeb ([HKvRW10]) in the experimental course of 2009. Taking into account this experience, we implemented and used our own tool Yoda in both courses in 2010.

In the rest of the paper we present Yoda[1]. We start describing in Section 2 those features that we look for in a tool. Next, in Section 3 we present Yoda by means of a running example. In Section 4 we comment briefly about our experience with this tool. Finally, we give an outlook on future work.

2 The context

For many years we have been interested in introducing tools to improve teaching. We believe there are different ways in how to measure this improvement; in our case, a greater involvement with the course and an (expected) decrease in the drop-out rate[2] would be highly satisfactory.

In this section we present some simple criteria to classify tools, as well as the requirements we consider important for a tool to fulfil in our educational context.

[1] www.fing.edu.uy/inco/cursos/lyc/soft/Yoda/indexProp.html
www.fing.edu.uy/inco/cursos/lyc/soft/Yoda/indexPred.html

[2] Last year 38 percent of students drop out of this course.

2.1 Classification criteria

During the last thirty years many tools have been implemented and used in the teaching of logic at university. In the following lines we propose some criteria to classify them ([Sie08]).

The first criterion refers to the relationship between the course and the tools used. Sometimes both are closely related (Tarski's World [BPBE99], Winke[3] [DE00]), or there is a set of tools that cover a wide spectrum of needs (Logic Daemon[4] [AH00]). In other cases, the tool focuses in a small problem that may be considered in its own (Jape [5]).

Tools and students interact in two different ways. In general, the student indicates each action of the tool, advancing step by step in solving the problem. This approach is followed by ETPS, ProofWeb and Papuq ([ABP$^+$04,HKvRW10,SC07]). On the other hand, some tools provide automatic mechanisms to complete the task.

A last criterion distinguishes the complexity of the interface implemented. In our case, the interface is the one given by a browser. This decision allows us to work minimally in this problem, but counting with the usual features of zooming and scrolling.

2.2 Requirements for the tool

We believe that a tool used in our course requires certain conditions in order to be successful. The first condition is that the tool should be very simple to execute. Our course has a teacher for every hundred students, and we cannot manage the problems that arise when training users.

Another condition is that the tool should be broadly accessible. This condition rules out those tools that are tied to a particular operating system, even when emulators or virtual machines may enable its portability.

The third constraint is that the tool should use Gentzen trees. We are not interested in building a course around a tool, but in using a simple tool inside an ongoing course that uses this representation. Francis J.Pelletier observes that Gentzen trees' method is not used much in elementary logic books ([Pel00]). This observation also holds when looking for software.

Yoda is a simple software focused in natural deduction without automatic features that satisfies our needs.

3 The tool

Yoda is a set of web based proof assistants developed as single-page applications, implemented in Javascript, CSS and jQuery[6], making simplicity one of

[3] `http://staff.science.uva.nl/~ulle/WinKE/`

[4] `http://logic.tamu.edu/`

[5] `http://users.comlab.ox.ac.uk/bernard.sufrin/jape.html`

[6] The library jQuery is used only to retrieve the browser dependent length of some texts.

its most remarkable features. There are no system requirements other than a standard-compliant web browser, and no user training needed as well, given the straightforward usability of the tool. We have implemented two assistants, one for building proofs in propositional logic, and the other for first order predicate logic.

To build a proof of $\Gamma \vdash \varphi$ with YODA, one should specify the conclusion φ and the premises set Γ via an intuitive ascii syntax[7]. Once the goal and premises are set, the building of the proof begins following a goal-driven strategy; the assistant shows the partial proof tree with the current conclusion as its root, and a drop-down list which displays the rules, premises and current hypotheses that one may apply to prove it. On choosing one of these, the assistant checks whether the action is applicable, in which case builds the corresponding subtree (prompting for additional input if necessary) or shows an informative message allowing the student to make a different choice.

We illustrate how the tool works with an example proof of $\{\forall x(P_0(x) \rightarrow P_1(x)), \exists x P_0(f(x))\} \vdash \exists x P_1(x)$. First we must specify the similarity type of our language, and then we can input the premises and goal (Fig. 1).

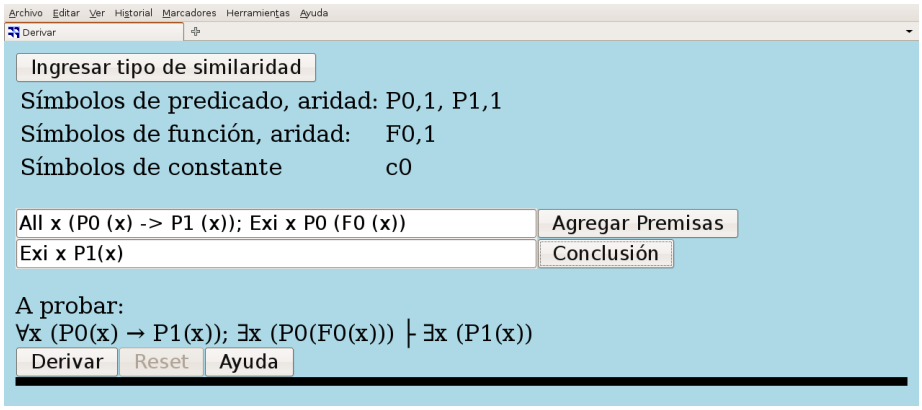

Fig. 1. Setting up the similarity type, premises and goal.

After checking the syntax of the formulas, YODA displays the partial proof tree with $\exists x P_1(x)$ as its root (Fig. 2).

We choose the $\exists$ elimination by clicking in the action list (Fig. 3), and input $\exists x P_0(x)$ when prompted.

Now we have two subgoals (Fig. 4), the leftmost is one of our premises and proving it requires the explicit selection of the assumption rule from the action list (Fig. 5). The other requires an $\exists$ introduction with $F_0(x)$ as witness, $\rightarrow$ elimination of $P_0(F_0(x)) \rightarrow P_1(F_0(x))$, hypothesis cancellation of $P_0(F_0(x))$ and

[7] If using the first order predicate logic assistant, the similarity type of the used language must be specified as well.

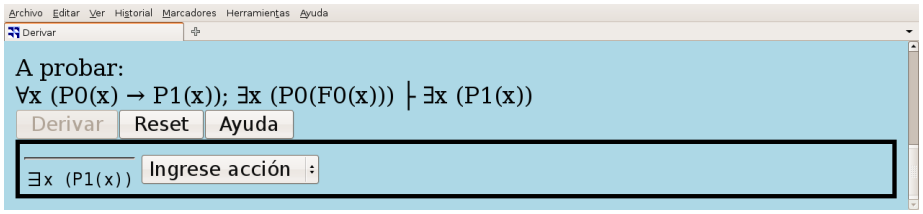

Fig. 2. Beginning the proof.

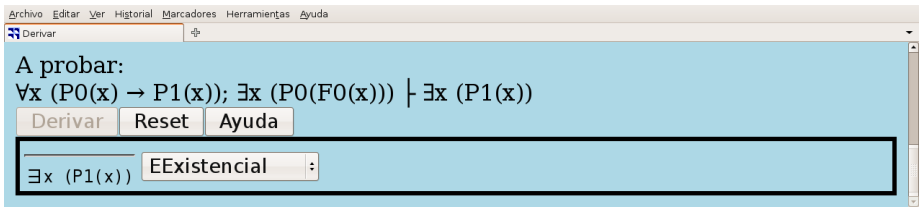

Fig. 3. Selecting the ∃ elimination rule.

∀ elimination of the remaining premise. Notice how used premises are marked in a different way than cancelled hypothesis in the proof tree. The complete derivation is shown on Fig. 6.

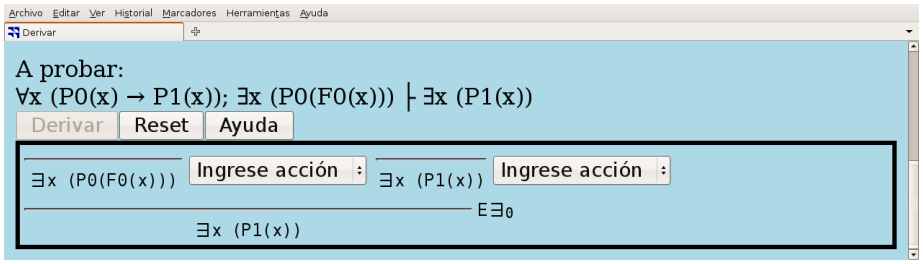

Fig. 4. New subgoals.

For larger proofs, (un)hiding of any subtree can be done by double clicking on its rule name so as to get a better view of the rest of the proof. To undo a proof or a part of it, one should double click on its conclusion.

4 Yoda in practice

A proper evaluation about the use of Yoda would require interviews and systematic monitoring of students attending the course. Unfortunately, at the moment we can only provide anecdotal information.

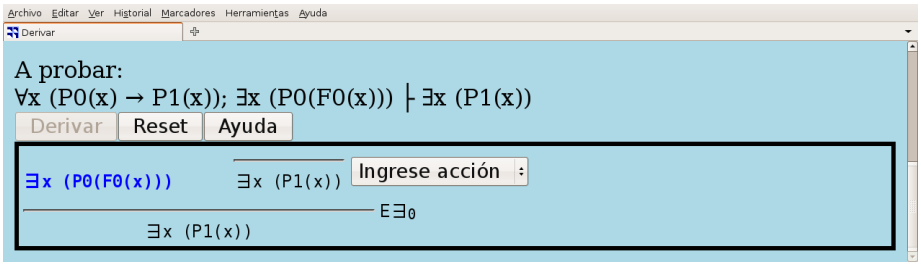

Fig. 5. Explicit cancellation rule.

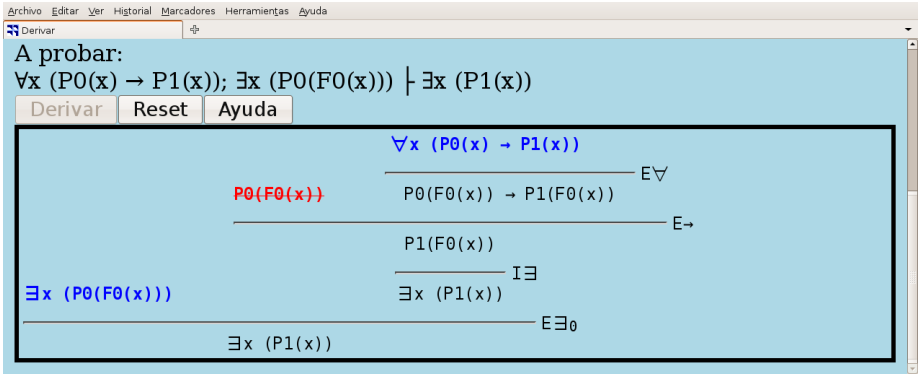

Fig. 6. Proof complete.

We implemented a first version of YODA based on comments of the students about the use of ProofWeb in 2009. Taking into account these comments, we abandoned the support of a logical framework for the benefits of a simpler interface and a faster response when using it. This version was enriched by students' proposals such as the use of libraries, a help webpage, and the design of the dialog boxes, thus becoming the definitive version for the courses of 2010. These students collaborated with the course of Logic in 2010 introducing YODA and explaining some basic exercises to the new cohort.

5 Conclusions and future work

We have developed YODA, a script that checks the correct construction of a proof. The main contribution of our tool is showing the potential of browsers to display logical proofs. This tool has been well accepted by our students; they have proposed improvements and explained and promoted its use.

The future development of the tool include the development of the following two libraries:

Library for ad-hoc compatibilities. Some browsers do not implement standard JavaScript entirely, turning important the development of a minimal

library for full portability. In Fig. 7 we can see our running example in a mobile phone; nevertheless, this proof has not been built in the mobile because of problems with the drop-down list. This sort of issues are a target of future improvements.

Library of metarules. Every natural deduction rule generates a set of JavaScript functions. We are interested in a language to express different rules and automatically generate this set of functions. Thus, YODA will be able to implement different deductive systems.

Moreover, we are interested in an improvement of the interface, as well as in the management of lemmas and incomplete proofs.

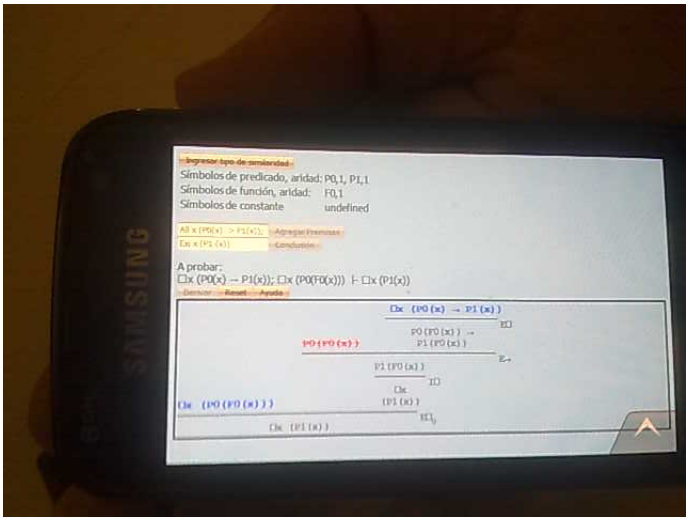

Fig. 7. Visualization of YODA in a mobile phone.

Finally, we agree with David Bostock ([Bos97]) when he states that the overall structure of a Gentzen tree proof is easily seen; in particular, for every point in the proof tree we know exactly what assumptions are used. Nevertheless, we did not have a simple and portable tool to deal with this representation. Our approach shows a simple way to obtain this tree representation leaving the main problems of visualization to the browser.

References

[ABP+04] Peter B. Andrews, Chad E. Brown, Frank Pfenning, Matthew Bishop, Sunil Issar, and Hongwei Xi. Etps: A system to help students write formal proofs. *Journal of Automated Reasoning*, 32:75–92, 2004.

[AH00] Colin Allen and Michael Hand. *Logic Primer*. MIT Press, 2000.

[Bos97] David Bostock. *Intermediate Logic*. Oxford University Press, 1997.

[BPBE99] Dave Barker-Plummer, Jon Barwise, and John Etchemendy. *Tarski's World*. CSLI Publications, 1999.

[DE00] Marcello D'Agostino and Ulrich Endriss. Winke: A proof assistant for teaching logic, June 2000.

[For08] ACM-IEEE Interim Review Task Force. Computer Science Curriculum 2008: an interim revision of CS 2001. 2008.

[HKvRW10] 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.

[oCC01] ACM-IEEE Joint Task Force on Computing Curricula. Computing Curricula 2001 - Computer Science - Final Report. 2001.

[Pel00] Francis Jeffry Pelletier. *Logical Consequence: Rival Approaches*, volume 1, chapter A History of Natural Deduction and Elementary Logic Textbooks. Hermes Science Pubs, 2000.

[SC07] Jakub Sakowicz and Jacek Chrzaszcz. Papuq: a coq assistant. In *Proceedings of PATE'07*, pages 79–96, 2007.

[Sie08] Luis Sierra. Enseñando deducción natural con coq. In *Proceedings of CIESC 2008, XVI Congreso Iberoamericano de Educación Superior en Computación*, 2008.

[vD94] Dirk van Dalen. *Logic and Structure*. Springer-Verlag, 1994.