Informatics in Education, 2005, Vol. 4, No. 1, 87–100

Point-and-Click Logic

Matti NYKÄNEN
Department of Computer Science, FIN-00014 University of Helsinki
P.O. Box 68 (Gustaf Hällströmin katu 2b), Finland
e-mail: matti.nykanen@cs.helsinki.fi

Received: January 2005

Abstract. Students of proof theory, a branch of formal logic, can benefit from computerized tools. We describe the principles behind one such tool called PROED. This tool is targeted especially at novice students, and therefore it is designed to support effortless exploratory use. We moreover argue that focusing on root-first proof construction in Sequent proof systems helps attain this effortlessness.
Key words: logic, user interfaces.

1. Introduction

Kapur (2004) recently wrote: "During the past ten years, theorem proving has been playing an increasingly important role in education. One reason is the improvement in the friendliness of theorem provers; many are now suitable for the nonspecialist [...] With computer-based educational tools profoundly changing the way instruction is imparted, automated reasoning tools are likely to have considerable impact in education and learning in the future." Such automated reasoning tools are basically computer implementations of various *proof theories*, or even meta-frameworks extensible with new proof theories. A proof theory for a logic is a formal system for constructing correct proofs for statements written in this logic, such as those presented by Negri and von Plato (2001). These proofs proceed step by step according to specific *inference rules*.

When the student encounters his first proof theory, he faces the following problem: He is given the collection of inference rules of the proof theory in question, and he should start composing proofs of logical formulæ using them. However, merely knowing the rules themselves is not enough to know how they can be made to work in unison as parts of a complete formal proof. Instead, the student must experiment with proof construction in order to see the rules in action.

Constructing a formal proof involves not only the reasoning steps themselves but also tedious bookkeeping to ensure that these steps really do connect to each other in an appropriate manner. Such bookkeeping is best left to the computer; otherwise the student's experimentation is hampered by mere notation, which distracts him from the actual subject matter.

This is why for instance the textbook by Negri and von Plato (2001) also provides the PESCA tool, a computer implementation of its proof theories. PESCA has been used in

the proof theory courses given by Negri at Helsinki University Computer Science Department. Note that it is not an introductory course on logic in general. Instead, the students should already be familiar with an informal notion of a logical argument, and this course focuses on its technical counterpart, the formal proof. This affects our perspective somewhat.

PESCA is a vast improvement over pencil and paper with respect to bookkeeping. On the other hand, its purely textual interface does not lend itself well to experiments like "What if I applied *this rule* to *that position* of my ongoing proof?": the student must first translate his experiment into a PESCA command which explicitly spells out not only the rule to apply but also the path to the desired position. Worse, if another rule has already been applied at that position, then the student must first *undo* it before the new rule can be applied in its place. Furthermore, once the student gains more confidence, his experiment can ask for a whole *series* of rule applications to be performed starting at the indicated position. In PESCA, the student must spell out the series of corresponding commands explicitly.

Our aim is to explain how the student can perform such an experiment with a *single click or drag-and-drop operation with the mouse* in a graphical user interface (GUI). We shall see how the student's operation suffices to indicate the *last* position and rule to apply. For certain kinds of proof theories, the computer can then generate the intermediate stages of the series without any further guidance from the user. This leads to a tool where the student uses mouse operations to indicate only the key choices in constructing the proof, while the computer performs the bookkeeping involved in the concomitant series of rule applications. We feel that such a tool supports effortless experimentation with different proof construction strategies.

Our presentation runs as follows. Next, Section 2 explains what kinds of proof theories suit our approach, followed by Section 3 which explains how mouse operations are translated into meaningful operations in such proof theories. Then Section 4 reports on the prototype implementation of our design. Finally, Section 5 concludes the presentation.

2. Requirements for the Logics

Let us now examine some main varieties of proof theories with respect to their suitability for student exploration in the manner described in Section 1.

Resolution (Russell and Norvig, 2003; Chapter 9.5) is taught in Artificial Intelligence (AI) courses, because a computer can find proofs in this system efficiently. On the contrary, it is unsuitable for humans for the following two reasons. First, it negates the formula to prove, and proceeds to find a contradiction. And second, this negated formula is first converted into a certain normal form which can be processed efficiently but which is very different from the original formula. After this conversion the student may no longer recognize the formula he should prove. Furthemore, he should be steering the proof toward a contradiction, and not toward any tangible result. As a result, the student may be quite lost.

Hilbert (or axiomatic) systems (Negri and von Plato, 2001; Chapter 2.5 (b)), taught in Mathematical Logic courses for their amenability to metamathematical analysis, do retain the original form of the formula to prove. However, they are unsuitable for another reason: The *axioms*, or the "obvious" starting points of proofs which require no further proof of their own, are often very complicated formulæ themselves, such as all tautologies of a certain form. The student is thus expected to master the use of these complicated formulæ right from the beginning of his study. A pathological example is the following complex proof for a simple result:

$$\dfrac{\dfrac{(A \Rightarrow ((A \Rightarrow A) \Rightarrow A)) \Rightarrow \quad A \Rightarrow ((A \Rightarrow A) \Rightarrow A)}{((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))} \qquad A \Rightarrow (A \Rightarrow A)}{A \Rightarrow A}$$

Note how the proof terminates in instances of the following tautological formula schemata given as axioms:

$$(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)) \qquad \text{and} \qquad \alpha \Rightarrow (\beta \Rightarrow \alpha).$$

Note also how tedious it is to verify that they are indeed instances of the respective schemata. Constructing such proofs is even more difficult for the beginning student, because he must also see the instances towards which he should be striving in his proof.

Hence we seek proof theories which both retain the original formulæ and have simple axioms. Two related candidates are **Natural deduction** (Negri and von Plato, 2001; Chapter 1.2) and **Sequent (or Gentzen) systems** (Negri and von Plato, 2001; Chapter 1.3). Natural deduction is taught in philosophical logic courses, because its inference rules correspond to intuition about how reasoning proceeds: An *introduction* rule brings a logical connective into the proof, while the corresponding *elimination* rule gets rid of it. For example, the introduction rule $\supset I$ for implication reads "if assuming A enabled you to prove B, then you have proved $A \supset B$".

$$\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \supset B} \supset I$$

The corresponding elimination rule

$$\dfrac{A \supset B \quad A \qquad \begin{array}{c}[B] \\ \vdots \\ C\end{array}}{C} \supset E \tag{1}$$

is engineered so that introduction immediately followed by elimination cancel each other out to yield the direct proof without the implication. Natural deduction is chosen by for example the ETPS system (Andrews *et al.*, 2004) with its emphasis in educational use.

As natural as this approach is, it too suffers from a drawback. The student is initially faced with only the conclusion C to prove. Suppose he wanted to experiment with rule (1). Where would the new assumption $A \supset B$ come from? While a seasoned logician may well somehow see it from C, the novice student may not. Coming up with just the right assumption is certainly a valuable skill to master in constructing proofs; but here we are designing a tool for learning proof construction to begin with. The situation is analogous to but easier than in Hilbert systems above: in both cases, the student must learn to anticipate the appearance of new, previously unseen formulæ into his proofs.

Another related candidate is using **analytic tableaux** as in for example the STRATUM system by Janhunen *et al.* (2004). This method is akin to solving systems of equations with the form "formula A is $\mathrm{true} \,/\, \mathrm{false}$" by repeated decomposition. However, the idea of Natural deduction and Sequent systems extends in a natural way to modelling other properties of sentences than their truth- or falsehood; an example are the Lambek calculi from formal linguistics (Carpenter, 1997; Chapters 5.1 and 5.2). That is, the latter are a more generally applicable tool then the former. In particular, Sequent systems provide decomposition rules on *proofs* instead of *truth values* as in analytic tableaux. Similarly, we argue that although **calculational reasoning** yields spectacular educational results already at the secondary school level (Peltomäki and Salakoski, 2004), applying it in, e.g., linguistics does not seem straightforward. We feel that our university-level target audience benefits from learning widely applicable tools.

We therefore opt for Sequent systems. Natural deduction drew conclusions like "formula C is true" below the horizontal line of the inference rule. Sequent calculi instead draw conclusions like "there is a proof of C from assumptions Γ"; these are denoted as $\Gamma \Rightarrow C$. Thus, sequent calculi provide an explicit notation for handling assumptions. Moreover, these assumptions are kept near the conclusion being drawn, whereas in natural deduction they may be far removed from it. On the other hand, this notation entails a lot of redundant bookkeeping, which makes sequent calculi tedious for experimentation on pencil and paper. However, we can relegate this routine bookkeeping to the computerized tool. In fact, Natural deduction systems can also be equipped with sequent-style assumption handling. Our student interaction concept, to be presented in Section 3 below, extends readily to such systems as well.

Consider as our example the proof theory depicted as Table 1. Introduction rules become right rules operating on a formula on the right-hand side of a sequent, while elimination rules become the corresponding left rules insted. Thus the left rule $L\supset$ corresponds to elimination rule $\supset E$ above. Comparing these two rules shows the effect: the student is faced with a sequent $\Delta \Rightarrow C$ to prove. However, now his choices are explicitly listed within it: he can manipulate either its right side C with the applicable right rule(s) or any member of its left side Δ with the applicable left rule. Moreover, examining the rules in Table 1 reveals that every formula above the horizontal line is present already below the line, either as a part A or B of the formula operated on by the rule or as a member of Γ, the other assumptions that are retained as is by the rule. This *subformula property* remedies the drawback of natural deduction noted above.

The proof theory in Table 1 is the *intuitionistic* propositional one presented by Negri and von Plato (2001). They also present a classical counterpart titled **G3cp** (Negri and

Table 1

The sequent system **G3ip** (Negri and von Plato, 2001; Chapter 2.2).

The axioms are

$$P, \Gamma \Rightarrow P \tag{2}$$

where the formula P is *atomic*, or contains none of the connectives '$\&$' (conjunction), '$\vee$' (disjunction), '$\supset$' (implication) or '$\bot$' (falsum). Each sequent is of the form $\Delta \Rightarrow D$, where its *right side* D is a formula, and its *left side* Δ is a multiset of formulæ. That is, Δ may contain multiple distinct copies of the same element. The shorthand A, Γ denotes such a multiset from which one such copy of its element A has been singled out while Γ consists of the other elements of the multiset. The inference rules are as follows:

	left rule	right rule(s)
$\&$	$\dfrac{A, B, \Gamma \Rightarrow C}{A\&B, \Gamma \Rightarrow C}\ L\&$	$\dfrac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A\&B}\ R\&$
$\vee$	$\dfrac{A, \Gamma \Rightarrow C \quad B, \Gamma \Rightarrow C}{A \vee B, \Gamma \Rightarrow C}\ L\vee$	$\dfrac{\Gamma \Rightarrow A}{\Gamma \Rightarrow A \vee B}\ R\vee_1$ $\dfrac{\Gamma \Rightarrow B}{\Gamma \Rightarrow A \vee B}\ R\vee_2$
$\supset$	$\dfrac{A \supset B, \Gamma \Rightarrow A \quad B, \Gamma \Rightarrow C}{A \supset B, \Gamma \Rightarrow C}\ L\supset$	$\dfrac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ R\supset$
$\bot$	$\dfrac{}{\bot, \Gamma \Rightarrow C}\ L\bot$	no such rule

von Plato, 2001; Chapter 3.1 (a)) whose major difference is that also the right-hand side of '$\Rightarrow$' is a multiset of formulæ. Our approach covers **G3cp** as well. However, let us concentrate here on the proof theory in Table 1 for definiteness.

The aforementioned subformula property suggests a particular proof construction mode: the student starts with the sequent $\Delta \Rightarrow D$ to prove. He selects from it a particular formula, either D or some member of Δ. The computerized tool can in turn extract all the information needed for displaying the new sequent(s) that should appear immediately above the starting sequent $\Delta \Rightarrow D$ (separated by the horizontal line). Then the user can continue with these new sequent(s) in the same manner, until he encounters axioms. Note that the axioms (2) are now easy to recognize, as they only require that the same formula P of the simplest possible kind appears on both sides of the sequent; this in turn remedies the defect of Hilbert systems noted above. This mode of proof construction is called *root-first:* The proof is in fact a tree of sequents, and this mode grows the tree from the ground up. (It is also called *top-down* in accordance with the computer scientists' habit of drawing their trees in an "antipodean" way with the root at the top.)

Although this root-first proof construction mode yields some proof for any theorem, the student may well plan his reasoning steps in some other mode as well. A well-known example is the *Modus Ponens* rule

$$\frac{A \supset B \quad A}{B}\ MP$$

which lies at the heart of Natural deduction systems and is a special case $C = B$ of our example elimination rule (1) (Negri and von Plato, 2001; Chapter 1.2). The natural mode to use this rule is *forward* reasoning: from "A implies B" and "A" infer also "B". On the other hand, the corresponding step in our root-first mode is

$$\frac{A \supset B, A, \Delta \Rightarrow A \quad B, A, \Delta \Rightarrow B}{A \supset B, A, \Delta \Rightarrow B}\ L\supset$$

where the reasoning proceeds *backwards* instead: the formula $A \supset B$ is first analyzed into its constituents A and B, which are then analyzed further into axioms. Systems like ETPS (Andrews *et al.*, 2004) and JAPE (Bornat and Sufrin, 1997) allow the student to proceed in either mode, and even mix these modes within the same proof. However, this complicates the student's interaction with the tool: he must indicate both the formulæ to operate on and the mode of operation. This in turn means that he must be aware of these two modes as well. We believe that a single-mode tool is a more appropriate one to begin with.

Incidentally, this distinction between forward and backward chaining of steps arises also in automated reasoning. There, backward chaining is found to be the more goal-oriented of the two (Russell and Norvig, 2003; Chapters 9.3 and 9.4). This finding accords with our discussion above: backward reasoning proceeds by splitting more complicated formulæ into their simpler constituents, and is therefore always oriented towards the simple axioms. In contrast, a forward reasoning step does not guarantee progress by itself, but only together with other choices by the student, as shown by our discussion on rule (1).

The student may naturally also encounter a dead end: a non-axiom sequent where no rule applies. Hence the student must also be allowed to *undo* his previous choices. In the tree metaphor, this amounts to pruning away some part of the tree, so that a new branch can be grown into its place using a different combination of rules.

Next, Section 3 explains how all this information can be extracted from the student's action. However, we must first point out a technicality: the rules in Table 1 have the additional property that the unmodified part Γ is passed from the sequent below the line to every sequent above it. That is, the context Γ of the rules is *shared* among the sequents. Another way would be to somehow partition the context Γ among these sequents. However, such partition information would be difficult to extract without additional information from the student.

3. Proof by Pointing

The student's mouse click must be translated into sufficient information about the proof steps that should be taken. This can be done with the *proof-by-pointing* approach suggested by Bertot and Théry (1998, Section 4) as a user interface concept for interactive theorem provers. Let us illustrate this approach with an example.

Suppose that the student wants to prove the transitivity of implication:

$$\Rightarrow ((A \underline{\supset} B)\&(B \supset C)) \supset (A \supset C). \tag{3}$$

Suppose further that he clicks the underlined implication '$\underline{\supset}$'. The natural interpretation is that he wants to apply some inference rule to it, because the user is pointing to it with the mouse as if saying, "I want to use *that thing* in my proof next!". However, that thing to use next is nested inside other connectives:

$$\Rightarrow \boxed{(\boxed{(A \underline{\supset} B)\&}(B \supset C)) \supset}(A \supset C).$$

Since the rules apply to the outermost connective only, we must first peel off the outer boxes one by one. The main connective of the outermost box is an implication '$\supset$' on the right side of the '$\Rightarrow$' (the left side being empty). Hence the rule to use must be $R\supset$:

$$\frac{\boxed{(A \underline{\supset} B)\&}(B \supset C) \Rightarrow A \supset C}{\Rightarrow ((A \supset B)\&(B \supset C)) \supset (A \supset C)}\ R\supset$$

This leaves the next box to peel off. Its main connective is the conjunction '$\&$' which has now jumped over the '$\Rightarrow$' on its left side. This jumping behaviour shows why these boxes must be peeled off from the outside in. The corresponding rule is therefore $L\&$. Now the underlined implication '$\underline{\supset}$' is finally exposed, and it can be peeled off as well with the rule $L\supset$, leaving the following incomplete proof:

$$\dfrac{\dfrac{\dfrac{A \supset B, B \supset C \Rightarrow A \quad B, B \supset C \Rightarrow A \supset C}{A \supset B, B \supset C \Rightarrow A \supset C}\ L\supset}{(A \supset B)\&(B \supset C) \Rightarrow A \supset C}\ L\&}{\Rightarrow ((A \supset B)\&(B \supset C)) \supset (A \supset C)}\ R\supset \tag{4}$$

At this point we stop and wait for the next mouse click from the student, because we have used up all the information from the previous one.

Note that this box nesting is uniquely determined by the structure of the original sequent (3) and the location of the mouse click inside it. That is, we can compute independently what the student's "use that" mouse click means without any additional information from him. The crucial requirement for this independent computation is noting that *the rule for peeling off a box is uniquely determined* by its side in the sequent and connective. We opted for sequent systems in Section 2 precisely because they permit such independent computation in the manner described above. This uniqueness also means that the student is still in full control of constructing the proof: the tool merely calculates the effects of his chosen rule applications, it does not choose any rules on its own. Even so, the student can perform a whole series of rule applications with a single click, as demonstrated. Or if the novice student finds this confusing, he can still work step by step instead; in our example above, he then begins by clicking the topmost implication '$\supset$' in sequent (3) instead. Thus this approach serves the student not only at the very beginning but also when he has become confident enough to try several inference steps at once.

In our example above, the final action of the independent computation was to apply the rule to the connective that the student clicked. However, this final rule application is

not necessarily unique. Such an ambiguous case is clicking a disjunction on the right side of the sequent; this click can mean either rule $R\vee_1$ or $R\vee_2$ of Table 1. In such a case, the computation proceeds until the clicked connective is no longer enclosed in any box, and stops there to wait whether the student chooses subformula A or B. This choice then disambiguates between the competing rules. This disambiguation is naturally automatic for all the rule applications preceding the last one.

The reason our example rules in Table 1 require such disambiguation is that they are intuitionistic: the proof of a disjunction requires a proof of either disjunct, because using the rule of excluded middle ("A or not A") is not permitted. Their classical counterpart **G3cp** needs no such disambiguation. However, we chose here the intuitionistic **G3ip** as our example in order to show how such disambiguation can be handled within our user interface concept when needed.

Admittedly one can envison proof systems with multiple different rules to peel off a box: for instance, the rule might depend on a combination of several connectives instead of just one as above. However, recalling our discussion in Section 2, this would somehow mean that these different combinations possess different introduction or elimination rules in the corresponding Natural deduction system. This would in turn mean that the intended meanings of these seemingly separate connectives would in some sense be intertwined after all. We feel that studying proof theory should start with other than such delicate systems.

The student can subsequently continue his proof (4) by clicking similarly inside either of its uppermost sequents $A \supset B, B \supset C \Rightarrow A$ or $B, B \supset C \Rightarrow A \supset C$. However, he soon finds out that the former leads into an infinite regress whereby rule $L\supset$ is repeatedly applied to the formula $A \supset B$. Thus this attempt at proving the original sequent (3) must be abandoned. The student can do this simply by clicking some of the sequents inside the proof (4). For instance, clicking on the implication '$\supset$' on the right of the third line in proof (4) substitutes it with

$$\dfrac{\dfrac{A, (A \supset B)\&(B \supset C) \Rightarrow C}{(A \supset B)\&(B \supset C) \Rightarrow A \supset C}\, R\supset}{\Rightarrow ((A \supset B)\&(B \supset C)) \supset (A \supset C)}\, R\supset \tag{5}$$

and the student is back on his way towards a completed proof.

In their study, Aitken *et al.* (1998, Sections 5.1.4 and 5.4) did not find significant benefits in this kind of user interaction with a theorem-proving program. However, their audience were mostly seasoned users of the program, not students of proof theory in general. Hence the benefits they expected differ from ours.

In fact, the ease of use outlined above may even become a hindrance in an educational setting: the student may accidentally arrive at a proof simply by random mouse clicks without truly understanding the proof. We currently have neither empirical evidence on whether this really happens or not, nor a solution if it turns out to be the case.

Table 2

The additional inference rules for the sequent system **G3i** Chapter 4.1 (c)).

We assume familiarity with the standard definition for the syntax of first-order predicate logic: variable symbols $x, y, z, \ldots$, terms $t, u, v, \ldots$ built out of these variable symbols and function symbols, and atomic formulæ built out of these terms and predicate symbols. In addition, $A[t/x]$ denotes the formula obtained by substituting the term t for every free occurrence of the variable symbol x in the original formula A.
Then we add to the connectives and rules of Table 1 the following quantifiers and rules:

	left rule		right rule	
$\forall$	$\dfrac{A[t/x], \forall x.A, \Gamma \Rightarrow C}{\forall x.A, \Gamma \Rightarrow C}$	$L\forall$	$\dfrac{\Gamma \Rightarrow A[y/x]}{\Gamma \Rightarrow \forall x.A}$	$R\forall$
$\exists$	$\dfrac{A[y/x], \Gamma \Rightarrow C}{\exists x.A, \Gamma \Rightarrow C}$	$L\exists$	$\dfrac{\Gamma \Rightarrow A[t/x]}{\Gamma \Rightarrow \exists x.A}$	$R\exists$

However, these new rules $R\forall$ and $L\exists$ have an additional *restriction* (also known as proviso or side condition) on their use: The variable symbol y substituted for x must not occur free in the conclusion below the line.

3.1. *Quantification*

Adding the universal and existential quantifiers '$\forall$' and '$\exists$' to the proof theory in Table 1 is straightforward in theory, and is given in Table 2. However, we must also extend our "proof by pointing" principles to these new quantifier rules.

The first obstacle is the term t substituted for x in the rules $L\forall$ and $R\exists$. While the rules do permit the student to choose any term t he wants, he should choose such a term t which allows him to complete the proof later. This completion in turn requires suitable applications of axioms (2), where the form of the atomic formula P depends on the term t chosen now. Thus it would seem that choosing t needs more information than can be extracted from a single mouse click, since t does not have to appear in the sequent below the line. Moreover, it seems that the student should know how to choose a good t even before he can see what kinds of axioms (2) he will later need to apply. Do we therefore suffer from the same drawback as rule (1) after all?

Fortunately we do not, because we can incorporate *metavariables* as Paulson (1996; Chapter 10.4) does in his sequent-based theorem prover with a command-line user interface. A metavariable is simply a kind of bookmark standing for some term whose actual form does not yet concern us and can be determined later. Then the student can still just click on a quantifier '$\forall$' on the left or on '$\exists$' on the right side of a sequent: the response is to create a previously unused metavariable for t when applying the corresponding rule of Table 2.

These created metavariables receive their eventual values when the student applies axioms (2). Now the sequent is of the form $P, \Gamma \Rightarrow P'$ where the atomic formulæ P and P' do not have to be exactly alike yet because they may contain metavariable occurrences. We can now reinterpret the axioms to require that P and P' must be made exactly alike

by selecting suitable values for their metavariables. The student can then select the axiom he wants by *grabbing P by its propositional symbol with the mouse, dragging it over the '$\Rightarrow$', and dropping it onto P'* (or vice versa). Thus we refine our earlier user interface concept into drag-and-drop for axioms and point-and-click for other rules.

The standard way to make two expressions alike is *unifying* them (Paulson, 1996; Chapter 10.7), (Russell and Norvig, 2003; Chapter 9.2), and it suffices here as well. Once unification binds some metavariable χ into some value v, then v must be substituted for χ throughout the whole proof under construction. Hence quantification introduces global dependencies between different parts of the same sequent proof, whereas the effects of the propositional rules of Table 1 remained local to the sequents in each rule. On the other hand, the handling of assumptions which may be far away already introduces global dependencies into the propositional rules of Natural deduction. Conversely, the binding of v into χ must be undone if the student later abandons this axiom application in favour of some other strategy.

Let us then turn to the remaining quantifier rules $L\exists$ and $R\forall$. At first sight their restriction seems innocuous: Just pick some previously unused variable symbol as the new y. But suppose that the sequent below the line contains the metavariables $\chi_1, \chi_2, \chi_3, \ldots, \chi_k$. (More precisely, they are the metavariables χ_i created in this branch of the proof.) Then the restriction actually demands that y must not occur in the values $v_1, v_2, v_3, \ldots, v_k$ that these metavariables will eventually have. The most common suggestion to ensure this seems to be picking a previously unused *Skolem* function symbol f_y instead, and using the term $u = f_y(\chi_1, \chi_2, \chi_3, \ldots, \chi_k)$ instead of y (Paulson, 1996; Chapter 10.7). Then any unification which tries to include u in a value v_i fails, because it would make v_i an infinitely long term, and that is prohibited. However, Paulson (1996; Chapter 10.7) maintains that Skolem functions make the formulæ unreadable, and this should be avoided in an educational tool. Instead he suggests recording the induced requirements "y must not occur in χ_i" and heeding them during unification. The downside of his suggestion is that this collection of induced requirements is a new aspect in our user interaction concept, which up to this point has consisted only of the proof itself, augmented with metavariables.

Either suggestion points to a more fundamental pedagogical problem: the unification fails when the axiom requested by the student does not apply. But how should the student be told *why* it failed? If the tool silently ignores the student's request and refuses to modify the proof, then the student gets no feedback at all and might even think that the tool is broken. On the other hand, the unification algorithm proceeds differently than the student, and therefore the position at which the algorithm detects the impossibility of the request may be meaningless to the student. It remains to be investigated if and how the execution of a failing unification algorithm can be traced to find the position where human and machine reasoning started to diverge.

The problem of generating meaningful error messages when unification fails is also of independent interest, since unification is used in for example compilers of such high-level programming languages as Standard ML (Paulson, 1996). However, the criteria for meaningfulness may also differ in logic and programming. That is, humans may use dis-

tinct reasoning in the two areas, and then finding the divergence position mentioned above should also be done differently in the two areas. This also remains to be investigated.

Note finally that the quantifier treatment suggested here is compatible with the "peeling off the outermost box" computation suggested above: the student can click inside a quantified formula, and the quantifiers enclosing the clicked position can be peeled off automatically with these rules.

3.2. *Equality*

Using the equality $t = u$ of two terms t and u is so common in proofs that it warrants its own special treatment. One possibility is having special inference rules (Negri and von Plato, 2001; Chapter 6.5)

$$\frac{t = t, \Gamma \Rightarrow C}{\Gamma \Rightarrow C}\ \textit{Ref} \qquad \frac{P[u/y], t = u, P[t/y], \Gamma \Rightarrow C}{t = u, P[t/y], \Gamma \Rightarrow C}\ \textit{Repl}$$

for reasoning about equality. We extend our user interaction concept to these rules as follows.

The intuition of rule *Repl* is that the occurrence of term t at position y within the atomic formula P should be replaced with u. We can refine the drag-and-drop approach introduced in Section 3.1 by allowing the student to grab not only whole atomic formulæ but also their terms and subterms. Then the student can simply drag the subterm at position y onto the left-hand term t of the equality to use (or vice versa). Again, these terms can contain metavariables, and they are therefore unified first.

Rule *Ref* resembles the rules $L\forall$ and $R\exists$ in Table 2 in the sense that it too introduces some previously unknown term t above the line which has no counterpart below the line. The difference is that whereas rules $L\forall$ and $R\exists$ provided the quantifier for the student to click, rule *Ref* offers no such visual disambiguating element. However, we can introduce such an element, if we make a mild additional assumption on the way the student is likely to proceed: We expect that he will use the new equality $t = t$ in the next step. Then we can also expect that the term t appears already as a subterm below the line. Thus we can refine our point-and-click approach as follows: clicking on a subterm t means an application of rule *Ref*, whereas clicking on a predicate symbol or a connective refers to the rules in Table 1 as before.

Unfortunately these suggestions do not permit using these rules automatically, in contrast to the suggestions in Section 3.1.

4. Implementation

We have a prototype implementation of the proof-by-pointing approach outlined in Section 3. This prototype is called PROED for PROof EDitor. Although the presentation of Section 2 focused on **G3ip**, an intuitionistic logic, the implementation supports the related classical logic **G3cp** as well.

Fig. 1 presents a screen shot of the tool. The sequent to prove is (3). The student needs only *two* mouse clicks to get this far: The first click on the implication '$\supset$' between A and C yields the incomplete proof (5), and the second click between A and B yields the situation as shown.

Our current implementation restricts itself to propositional logic. That is, it does not support the refinements suggested in Sections 3.1 and 3.2 yet. On the other hand, the restriction to propositional logic also permits some extra features such as automatic detection of axioms. In addition, our implementation provides facilities to save an incomplete proof and continue it later, or to print it or parts of it in LaTeX.

Adding the refinements suggested in Section 3.1 would be the next step in developing our implementation further, because then it would cover all the core material of the introductory proof theory course mentioned in Section 1. The further refinements in Section 3.2 would be straightforward after that. Systems like ETPS (Andrews *et al.*, 2004) and STRATUM (Janhunen *et al.*, 2004) also offer pedagogical tools for assessing the students' progress, which our PROED currently lacks.

We have offered our implementation to the students of the course. Anecdotal evidence

The tool differs slightly from the proof system in Table 1 in the following three respects: (1) Axioms are explicitly marked with 'Ax' to provide an explicit visual clue that this branch has now been dealt with. (2) Conjunction is denoted with '$\wedge$', disjunction with '$\vee$' and implication with '$\rightarrow$'. This notation is perhaps more common in mathematics and computer science, while the former seems to be preferred by philosophers. (3) The left implication rule $L \rightarrow$ omits displaying the repeated formula $A \rightarrow B$ to save screen space. Invoking this rule again would merely lead into an infinite regress, as discussed in Section 3.

Fig. 1. A screen shot of the PROED tool.

suggests that they preferred it to the PESCA tool also mentioned in Section 1, but no formal comparisons have been performed. This preference was to be expected anyway, since users tend to generally prefer graphical to textual user interfaces (at least until they become more proficient in the task at hand). The course is elective, which means that it may not be offered regularly, and that its attendance may be small. These aspects of the course make it harder to gather more substantial evidence.

The current implementation is wholly written in the Java programming language, and therefore runs on multiple platforms. Bornat and Sufrin (1997) have developed JAPE which both supports similar functionality and is customizable to different proof theories. Thus JAPE is an example of a meta-framework in the sense of Section 1, one with its emphasis on the user interface. Switching from our proprietary implementation into a JAPE-based one would entail encoding the rules in Table 1 in its logic encoding language, together with enough guidance information for the rule selection machinery of Section 3 to work. Future versions of our tool may well be JAPE-based, if it turns out to suit our needs. The support for quantifier handling in JAPE would be especially valuable in implementing the refinements suggested in Sections 3.1 and 3.2. On the other hand, JAPE seems to offer no pedagogical tool support, so choosing the next implementation platform involves a tradeoff.

The implementation described here is available at
`http://www.cs.helsinki.fi/group/toto/`.

5. Conclusion

We have explained the principles behind PROED. This computerized tool is intended for novice students of propositional sequent systems of logic. The tool frees the student from tedious but necessary bookkeeping aspects, and lets him concentrate on exploring the different choices available for proof construction. Such exploration is made especially easy by providing a graphical user interface where single mouse clicks suffice for disambiguating between these choices.

Acknowledgements. The author wishes to thank Raul Hakli for fruitful discussions; he has been both the course assistant and a draft version reviewer of the textbook (Negri and von Plato, 2001). Thanks are also due to the implementors of the prototype in Section 4: team leader Marja Huovinen, and participating students Teppo Kankaanpää, Jaakko Nenonen, Aki Nyrhinen, Visa Röyskö and Juha-Matti Tapio.

References

Aitken, J., P. Gray, T. Melham, T.Thomas (1998). Interactive theorem proving: An empirical study of user activity. *Journal of Symbolic Computation*, **25**, 263–284.

Andrews, P., C. Brown, F. Pfenning, M. Bishop, S. Issar, H. Xi (2004). ETPS: A system to help students write formal proofs. *Journal of Automated Reasoning*, **32**, 75–92.

Bertot, Y., L. Théry (1998). A generic approach to building user interfaces to theorem provers. *Journal of Symbolic Computation*, **25**, 161–194.

Bornat, R., B. Sufrin (1997). Jape: A calculator for animating proof-on-paper. In *14th International Conference of Automated Deduction (CADE 14)*. Vol. 1249 of Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 412–415.

Carpenter, B. (1997). *Type-Logical Semantics*. The MIT Press.

Janhunen, T., T. Jussila, M. Järvisalo, E. Oikarinen (2004). Teaching Smullyan's analytic tableaux in a scalable learning environment. In A. Korhonen, L. Malmi (Eds.), Kolin kolistelut – Koli Calling 2004. *Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science. Education*, No. TKO-A42/04. Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science, pp. 85–96.

Kapur, D. (2004). Preface to a special issue on "automated reasoning and theorem proving in education" (arte). *Journal of Automated Reasoning*, **32**, 1–2.

Negri, S., J. von Plato (2001). *Structural Proof Theory*. Cambridge University Press.

Paulson, L. (1996). *ML for the Working Programmer*, 2nd Edition. Cambridge University Press.

Peltomäki, M., T. Salakoski (2004). Strict logical notation is not a part of the problem but a part of the solution for teaching high-school mathematics. In A. Korhonen, L. Malmi (Eds.), Kolin kolistelut – Koli Calling 2004. *Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*, No. TKO-A42/04. Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science, pp. 116–120.

Russell, S., P. Norvig (2003). *Artificial Intelligence: a Modern Approach*, 2nd Edition. Prentice-Hall.

M. Nykänen received his PhD in 1997 from the University of Helsinki, Finland. His research focus has been on theoretical aspects of computer science: algorithmics, automata theory, and applications of logic in database theory and artificial intelligence.

"Pažymėk ir spragtelk" logika

Matti NYKÄNEN

Mokant studentus įrodymų teorijos, kuri yra formaliosios logikos atšaka, gali būti pravarčios ir kompiuterinės priemonės. Straipsnyje supažindinama su viena iš tokių priemonių, kuri pavadinta "ProEd". Ši priemonė skirta pradedantiems studijuoti, todėl ji suprojektuota taip, kad atliekant tiriamąjį darbą, ją būtų galima naudoti be didesnių pastangų. Straipsnyje stengiamasi įrodyti, jog susitelkiant ties sekos įrodymų sistemų pirminėmis konstrukcijomis galima užtikrinti ypatingų pastangų nereikalaujančią sistemą.