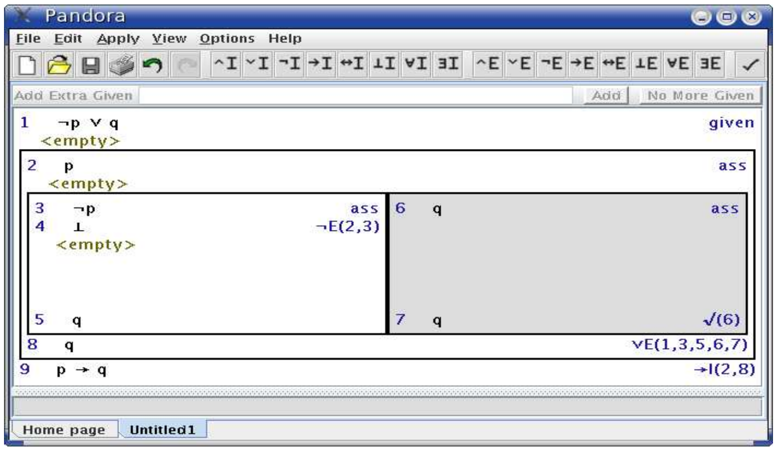
Pandora: A Reasoning Toolbox using Natural Deduction Style

KRYSIA BRODA, JIEFEI MA, GABRIELLE SINNADURAI, ALEX SUMMERS, *Department of Computing, Imperial College London, South Kensington Campus, Exhibition Road, London SW7 2AZ, UK.*
E-mails: {kb,jm103,apgs,ajs300m}@doc.ic.ac.uk

Abstract

Pandora is a tool for supporting the learning of first order natural deduction. It includes a help window, an interactive context sensitive tutorial known as the "e-tutor" and facilities to save, reload and export to latex. Every attempt to apply a natural deduction rule is met with either success or a helpful error message, providing the student with instant feedback. Detailed electronic logs of student usage are recorded for evaluation purposes. This paper describes the basic functionality, the e-tutor, our experiences of using the tool in teaching and our future plans.

Keywords: natural deduction, Fitch box proof, first order logic, predicate logic, teaching, learning, e-learning, reasoning about programs


Pandora
File Edit Apply View Options Help
∧I ∨I ¬I →I ↔I ⊥I ∀I ∃I ∧E ∨E ¬E →E ↔E ⊥E ∀E ∃E
Add Extra Given
Add
No More Given
Home page
Untitled1

```
1  ¬p ∨ q                          given
   <empty>
2  p                               ass
   <empty>
3  ¬p               ass   6  q     ass
4  ⊥             ¬E(2,3)
   <empty>
5  q                      7  q     √(6)
8  q                      ∨E(1,3,5,6,7)
9  p → q                  →I(2,8)
```

1 History and Context

Natural deduction has been taught to first year undergraduates in the Department of Computing at Imperial College since 1991. Most students find the high degree of rigour required in formal natural deduction proofs daunting. Pandora (Proof Assistant for Natural Deduction using Organised Rectangular Areas) is a learning support tool designed to guide the construction of natural deduction proofs [1]. The acronym was coined by Lloyd Kamara. It was conceived in 1996 and is based on the Fitch-style "proof box notation" for natural deduction as described in [2], which has been the recommended text for the course since it was developed from the course lecture notes. The first version of pandora was written by a final year student, Patrick Chan, using Tcl/Tk in 1996 and was available for students to use in tutorials but was not promoted in lectures. A second version was written in an early release of Java in 1999 by another project student, Dan Ehrlich. This second version was more robust and was used for demonstrations in class as well as in tutorials. It was available for students to download onto their own machines and promoted vigorously and was consequently widely used. The third and current version, also written in Java, was developed as a group project in 2002 and has since been further enhanced by several summer students, Jiefei Ma, Alex Summers and Tom Weedon. It provides the basic functionality together with a help module, a context sensitive tutorial and the facility to export proofs as latex. An applet version is available for use via the pandora web site [8] and we hope to make it available for download in the near future.

2 Basic Functionality

Pandora provides learning support to guide students in their construction of a natural deduction proof of a conclusion or goal from given premisses. It allows the user to reason "forwards", that is, from one or more given formulas to deduce another formula using one of the rules, and to reason "backwards", that is, to reduce one of the current goals to one or more subgoals from which the current goal can be deduced using one of the rules. The rules are all the usual introduction and elimination rules of first order natural deduction with equality plus a few derived rules found useful by more advanced users.

We will explain the usual use of the tool by working through a small example to derive $p \rightarrow q$ from the premiss $\neg p \vee q$. On starting a new proof pandora first requires the premisses and goal to be given. These are typed by the user into a text box. In our example there is only one premiss so we type `~p|q` then click on the "no more given" button. Pandora then requires us to give the goal. We type `p>q` into the text box. Pandora checks that these are well formed formulas and then displays the initial state of the partial proof with the premisses at the top and goal at the bottom. This is shown om the left below.

1	$\neg p \vee q$	given
	$<$ *empty* $>$	
2	$p \rightarrow q$	conclusion

1	$\neg p \vee q$	given
	$<$ *empty* $>$	
2	p	ass
	$<$ *empty* $>$	
3	q	
4	$p \rightarrow q$	$\rightarrow\mathcal{I}$(2,3)

Notice that we input the symbols for "not", "or" and "implies" as the plain text characters `~`, `|` and `>` but pandora displays them as $\neg$, $\vee$ and $\rightarrow$. We experimented with using buttons to represent the symbols but found that plain text was more convenient for user input and having to learn a second set of symbols for the connectives did not present any problem to our students. Now for the proof. First we apply the $\rightarrow$ introduction rule backwards by selecting the goal, $p \rightarrow q$, and clicking on the $\rightarrow\mathcal{I}$ button. Pandora displays the new proof state, shown on the right above. The box shows the scope of the assumption, p, which is discharged by the $\rightarrow\mathcal{I}$ rule and the conclusion, $p \rightarrow q$, now has the justification $\rightarrow\mathcal{I}(2, 3)$.

Next we would like to apply the $\vee$ elimination rule to split the premiss $\neg p \vee q$ into its two cases. The current goal is q and we want to maintain this so we must use the rule backwards by selecting the goal, q, and clicking on the $\vee\mathcal{E}$ button then clicking on the premiss $\neg p \vee q$. The new proof state is displayed.

1	$\neg p \vee q$	given			
	$< empty >$				
2	p	ass			
	$< empty >$				
3	$\neg p$	ass	5	q	ass
	$< empty >$			$< empty >$	
4	q		6	q	
7	q	$\vee\mathcal{E}$(1,3,4,5,6)			
8	$p \rightarrow q$	$\rightarrow\mathcal{I}$(2,7)			

In the right hand box we have q as an assumption and q as the goal so we can just use the "tick" rule to note that the goal is proven. So we select the goal q, then click on the ✓ button, then on the assumption q. Pandora displays the new proof state with the $<$ *empty* $>$ line removed from the right hand box which is greyed out as that part of the proof is completed.

To save space we will not show it yet, but first apply a rule in the left hand box. This will be our first application of a rule forwards. We have both p and $\neg p$ in scope so we can deduce $\perp$ by the $\neg$ elimination rule. To do this in pandora we select the $<$ *empty* $>$ line in the left box, then click the $\neg\mathcal{E}$ button, then click on the two formulas which give the contradiction. The new proof state displayed by pandora is the one in the screen shot on the first page.

Next we apply $\perp$ elimination backwards from the goal q in the left hand box by selecting the goal then clicking on the $\perp\mathcal{E}$ button. This gives $\perp$ as the new goal. As usual, pandora would display the new proof state.

Finally we use the tick rule again to note that we have $\perp$ and we need to show $\perp$. The proof is now complete. Pandora removes all the empty lines and greys out the proof and is intelligent enough to remove the last application of the tick rule which is not needed in the final proof. Note that the line numbers and references to them in the justifications were consistently updated as the proof emerged. Note also that in the completed proof every line has a justification.

1	$\neg p \vee q$	given			
2	p	ass			
3	$\neg p$	ass	6	q	ass
4	$\perp$	$\neg\mathcal{E}$(2,3)			
5	q	$\perp\mathcal{E}$(4)	7	q	✓(6)
8	q	$\vee\mathcal{E}$(1,3,5,6,7)			
9	$p \rightarrow q$	$\rightarrow\mathcal{I}$(2,8)			

If at any stage the student tries to apply a rule wrongly they are given an error message. The use of the tool to construct a proof as outlined above is augmented by a help window, interactive tutorial known as the "e-tutor" and export to latex facility. All the proof states shown above were generated using pandora's export to latex option. The student can save and reload partially constructed or complete proofs and work on several proofs at once, each in its own proof window. They can undo proof steps with the undo button allowing them to try alternative proof strategies.

3 The e-tutor

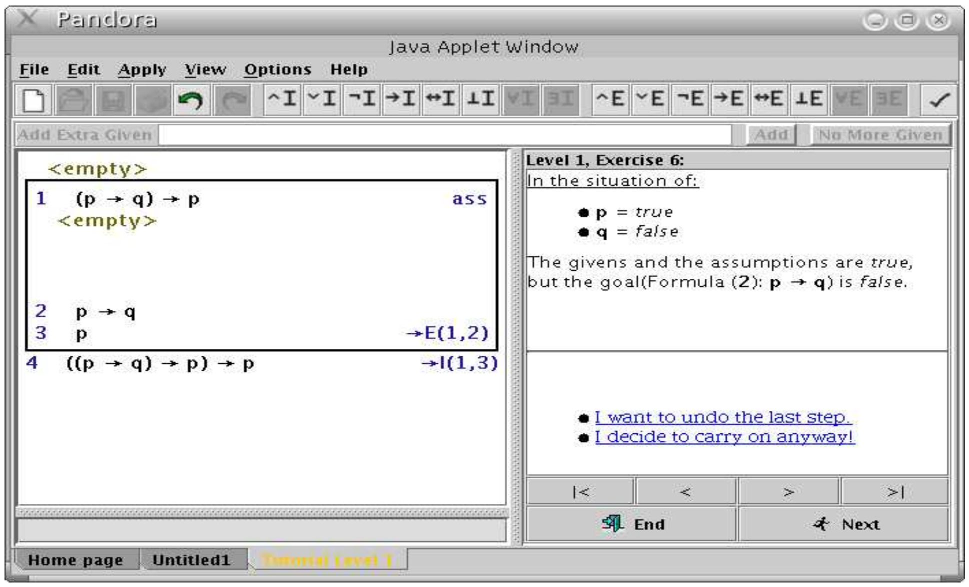
On loading pandora, the user is presented with the pandora home page, which offers various options, including loading a previously stored proof, starting a new proof or starting an e-tutorial. When the e-tutorial is started, four tutorials of propositional exercises are available. The first consists of fixed exercises and is useful in teaching when we demonstrate the tool in class or run a laboratory session in which all the students are required to attempt the same exercise. The other three, known as "easy", "medium" and "hard", consist of sets of five exercises which are randomly selected at run time by the e-tutor.

In normal mode, pandora just responds to the student's actions by either applying the rule they request or giving appropriate error messages. The student's strategy is not checked and if a student applies a rule correctly but unwisely, turning an achievable goal into an unprovable goal they are not warned. When an exercise is attempted in the e-tutor the level of support is greatly enhanced. Hints and explanations are provided, a warning and counter-model are given if the student applies a rule which creates an unprovable goal, and the friendly e-tutor will simply do a step for the student if asked.

We will explain the use of the e-tutor by following a hypothetical student's attempt to derive Pierce's law $((p \to q) \to p) \to p$ which is one of the fixed tutorial exercises. In the e-tutor the view is split into a left window in which the proof is constructed and a right window in which the e-tutor offers advice. Initially the proof window just contains the goal $((p \to q) \to p) \to p$ and the advice window just contains the remark "I am always here to help you" and a clickable link "I need some advice". Suppose our student clicks the link. The tutor suggests "Arrow introduction rule" and the suggestion is itself a clickable link to an explanation of the rule. On clicking this link the student is given an explanation of the usual use of the rule and what will happen if it is applied in this particular case and another clickable link "How do I do it?". On clicking this link the student is told exactly what to click on to apply the rule and given two more clickable links. In this case the advice is to select the goal, $((p \to q) \to p) \to p$, and click on the $\to\mathcal{I}$ button. The links are "Do the steps for me!" and "Show the advice again". When our lazy student selects the former the proof window is updated as shown in the proof state below and the advice window goes back to showing the remark "I am always here to help you" and a clickable link "I need some advice".

$$\begin{array}{lll}
 & <\mathit{empty}> & \\
\begin{array}{|lll|}
\hline
1 & (p \to q) \to p & \text{ass} \\
 & <\mathit{empty}> & \\
2 & p & \\
\hline
\end{array} & & \\
3 & ((p \to q) \to p) \to p & \to\mathcal{I}(1,2)
\end{array}$$

Our student can again ask for advice and this time will be offered the choice of "Law of Excluded Middle" or "Proof by Contradiction". Again these are clickable links to explanations of the rules and on selecting them further offers of links "How do I do it?" and "Do the steps for me!" would be supplied as above. But suppose our overconfident student reasons that the goal p could be proved by $\rightarrow\mathcal{E}$ using $(p \rightarrow q) \rightarrow p$ if $p \rightarrow q$ can be proved and so decides to ignore the advice and instead applies the $\rightarrow\mathcal{E}$ rule backwards by clicking on the goal p then the $\rightarrow\mathcal{E}$ button then the assumption $(p \rightarrow q) \rightarrow p$. The friendly e-tutor allows the rule to be applied in the proof window but responds with a warning in the advice window "Be careful, one or more of your goals are not provable!" and offers the links "Why?", "Undo last step" and "I want to carry on anyway". If the student asks why, the tutor provides a counter-example truth value assignment as shown in the screenshot below. If the student decides to carry on anyway the tutor reports that "No advice is available at the moment because the goal is not provable".



```
  <empty>
1   (p → q) → p          ass
    <empty>

2   p → q
3   p                    →E(1,2)
4   ((p → q) → p) → p    →I(1,3)
```

Level 1, Exercise 6:

In the situation of:

- p = *true*
- q = *false*

The givens and the assumptions are *true*, but the goal(Formula (2): **p → q**) is *false*.

- I want to undo the last step.
- I decide to carry on anyway!

Our student can undo the incorrect step and apply Excluded Middle (EM) as advised instead and then continue either with or without the help of the tutor. The e-tutor just sits quietly presenting the link "I need some advice" unless the student either clicks that link or makes a move which results in an unprovable goal.

4 Using Pandora in Teaching

We have two different cohorts of students learning logic in their first year. One class (JMC) is small (20-25 students) and studying for a joint Mathematics and Computer Science degree, whereas the other class (COMP) is large (over 100 students) and studying for a Computing degree. The treatment in respect of using pandora was different for the two cohorts.

The JMC group is taught by an enthusiastic lecturer, has enthusiastic tutors, and uses pandora in an integrated way to teach natural deduction which constitutes about

25% of the logic module. Initially, the propositional natural deduction rules are presented and hand written examples of proofs are given. Only after they have seen several proofs and tried a few on paper themselves is pandora introduced. Pandora is demonstrated using several of the same examples so the students can focus on how to drive pandora rather than on how to prove the theorems. Over the next few weeks, in the lab and tutorials, we give the students many exercises, some assessed, and introduce them to the first order and equality rules. The course finishes with a "driving test" consisting of ten problems which the students have one hour to attempt under exam conditions. Last year the test problems were:

1. $\vdash (p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$
2. $\neg p \vee q \vdash (p \rightarrow q)$
3. $\neg(p \rightarrow q) \vdash p \wedge \neg q$
4. $p \rightarrow (q \rightarrow (r \vee s)), \neg(r \vee t), (s \wedge q) \rightarrow t \vdash q \rightarrow \neg p$
5. $\forall x \neg p(x) \vdash \neg \exists x p(x)$
6. $\neg \exists x p(x) \vdash \forall x \neg p(x)$
7. $\neg \forall x p(x) \vdash \exists x \neg p(x)$
8. $\neg \exists x \exists y (\neg x = y) \vdash \forall x \forall y (p(x, y) \rightarrow p(y, x))$
9. $\forall x (x = a \vee x = b), \neg(a = b), g(a) = b, \forall x \forall y (g(x) = g(y) \rightarrow x = y) \vdash g(b) = a$
10. $\forall x \forall y \forall z (r(x, y) \wedge r(y, z) \rightarrow r(x, z)), \forall x (r(x, a) \vee r(x, b)), r(a, b) \vdash \exists y \forall x r(x, y)$

Pandora has been enhanced with various features to facilitate automatic marking. For assessed coursework we provide the students with exercise "skeletons". These include the given premisses and conclusion together with a "magic number" which encodes any restrictions we care to impose on the rules they are allowed to use. For example, this feature allowes us to disable certain derived rules (eg the PC rule) or to disable all the predicate rules which may be a distraction when they are proving a propositional assertion. The magic number also encodes the current state of the proof so the students cannot take the number from one proof and put it on another. The students download the skeletons from our web based continuous assessment tracking system and the first time they save a proof (usually when they have completed it!) their identity is also coded into the magic number. Whenever the proof is saved subsequently, the coded identity is unchanged so the first saver can be checked against the submitter of the proof. We are pleased to say that for the two cohorts we have checked there was no evidence of "work sharing" at all even though the students did not know about this feature. The electronically gathered proofs from both the driving test and assessed coursework are checked for correctness and converted to latex by "text to text" command line programs included in the pandora package. We can thus produce a report for each student and a summary of results for their tutors with minimal human intervention.

The experience of the larger class (COMP) has been variable over the last three years. The first year the current version of pandora was available its use was not encouraged, with the result that few students tried it. In the second year there was light encouragement and a demo, and more students tried it out, some using it to do their coursework. Only in the third year of its availability was pandora seriously encouraged, but there was still no driving test and no requirement to use it for assessed coursework.

For both cohorts, all exercises done throughout the term, including the driving test for the JMC, are essentially formative. The main summative assessment is an end of year written examination.

5 Evaluation

We have put considerable effort into evaluating pandora and used three methods.

Firstly, we asked the students what they think of it, both verbally and using anonymous feedback forms. The feedback is generally encouraging and students say and write that they enjoy using pandora and find it useful. The JMC classes, who learned pandora thoroughly, enjoyed using pandora more and gave more encouraging feedback than the COMP cohort. Moreover, the feedback from the COMP cohort improved as pandora was more actively encouraged. We concluded that perhaps students tell us what they think we want to hear! Students also made useful criticisms: for example the first release of the current version only had the facility to undo the last rule application but the facility to repeatedly undo steps was added by popular demand.

Secondly, we compared performance on the written exam by the two cohorts. This did not give the clearcut advantage to pandora users that we hoped for and in fact there was little difference in terms of marks between the two cohorts. We believe the reasons for this were that (i) natural deduction, being only 25% of the course, did not form a large part of the exam and (ii) the questions had to be small enough to be completed under exam conditions. We also believe that a difference between the cohorts may have shown up if the questions had been substantial. There did, however, seem to be a difference in style, namely that those who used pandora were much more at home with using rules backwards and did not make "arbitrary" assumptions which they had no hope of discharging whereas the cohort who did not use pandora mainly reasoned forwards and frequently made arbitrary assumptions. We had feared that pandora users may find it hard to adapt to writing proofs by hand but it turned out that the users were more precise syntactically in their hand written proofs than were the non-users.

For the third evaluation method we electronically recorded detailed logs of the students' use of pandora; essentially we recorded every "click" they make so that we can see in detail how they actually use it. This allowed us to see which rules were applied, whether undo, help or the tutorial were used, and so on. We computed proportions of attempted rule applications that were correct and counted error rates for various types of errors. Some of the results from analysing the logs came as a disappointment in that they showed that the help and tutorial facilities were used rather infrequently. The logs also showed a surprisingly high failure rate in students' attempts to apply the rules. A small number of students had virtually no failures but many had almost as many failures as successes. Analysis of the logs showed that many students were not selecting the $< empty >$ line or a goal line before applying a rule. Comparing the logs for the driving test with those for the previous work we were pleased to observe that, with experience, the proportion of failed rule applications decreased. The logs yielded detailed information about the common errors made by students for each rule but, perhaps surprisingly, not many general problems could be diagnosed. The one fact that really jumped out of the data was that students frequently tried to apply the $\neg\mathcal{I}$ rule backwards to a formula which was not a negation, whereas it was comparatively rare for them to try applying $\rightarrow\mathcal{I}$ backwards to a formula which was not an implication. We believe they were confusing the $\neg\mathcal{I}$ rule with the derived PC rule.

Overall the evaluation has taught us that pandora is well liked and considered useful by the students, but that to improve the learning outcomes we should modify our teaching in a number of ways. We have run the course again since the evaluation and made some improvements, notably (i) we gave an advertisement for the e-tutor and help as part of the initial demonstration, (ii) when demonstrating the individual rules we emphasised that either the $<$ *empty* $>$ or $<$ *goal* $>$ line needs to be selected **before** clicking the rule application button, and (iii) we explained how to avoid what the logs showed to be the common pitfalls in applying the rules. The course only finished last week and we have not yet had time to analyse the logs so we do not know whether the actions mentioned in (ii) and (iii) above made a difference. Anecdotally we believe they did. The demo of the e-tutor producing the counter-model initially illicited surprise from the students and they were very keen for me to select "I want to carry on anyway". When the e-tutor said "No advice is available at the moment because the goal is not provable" the students laughed because they identified with its predicament. We think the laugh indicated a warm feeling towards the e-tutor as well as an understanding of the semantic checks it was making.

6 Future direction 1: Pandora and Prolog

We teach Prolog [4] in the third term of the first year by which time our students are very familiar with natural deduction and pandora. If natural deduction is applied to the subset of first order logic consisting only of Horn clauses, it is quite easy to simulate the action of a Prolog interpreter. So why not use pandora to help their understanding of backtracking and clausal reasoning?

Prolog can be introduced without resolution by viewing Prolog derivations of query Q from program P as natural deduction proofs of a goal Q from premisses P. A Prolog program consists of definite clauses which have the general form $A \leftarrow B_1, \wedge \ldots \wedge B_n$, where A, B_1,…,B_n are atoms. In case $n = 0$ the clause has the form A and is called a fact; otherwise it is called a rule. Any variables in a clause are implicitly universally quantified. The query is a conjunction of atoms, in which variables are implicitly existentially quantified. Prolog finds all solutions for an existential query by back-tracking. This is conveniently modelled in pandora by use of UNDO.

Currently in $\forall\mathcal{E}$ forwards and $\exists\mathcal{I}$ backwards only a ground term can be substituted for the bound variable. These rules will be generalised to allow the introduction of a fresh unknown in place of the bound variable. The tick rule currently just checks that two ground formulas match. For Prolog this must be extended to include unification between atoms. Similarly backwards application of the rule $\rightarrow\mathcal{E}$ must allow for the atomic goal to unify with the head atom of the major ($\rightarrow$) premise. For convenience backwards application of $\wedge\mathcal{I}$ will be generalised for an arbitrary number of conjuncts.

Below is an illustration of the deduction of a query from a five clause program in which it is assumed that a derivation using b for x has already been found and undone and a second derivation is sought.

When a rule is applied which binds an unknown the replacement of variables by values must happen throughout the proof. This overwriting means that the unknowns are lost from the proof state. The diagram below is not itself a proof state because we have left some of the unknowns in so that you can see a shadow of earlier states. One such replacement, which was caused by the `tick` rule used to justify line 7, is applied to line 10 and indicated by $\Uparrow$.

1	$arc(a,b)$	program fact
2	$arc(a,d)$	program fact
3	$arc(b,c)$	program fact
4	$\forall u \forall v[path(u,v) \leftarrow arc(u,v)]$	program rule
5	$\forall u \forall v \forall w[path(u,w) \leftarrow arc(u,v) \wedge path(v,w)]$	program rule
6	$path(?u1,?w1) \leftarrow arc(?u1,?v1) \wedge path(?v1,?w1)$	$\forall\mathcal{E}(5)$

			8	$path(?u2,?v2) \leftarrow arc(?u2,?v2)$	$\forall\mathcal{E}(4)$
			9	$arc(b,c)$	$\surd$ (3)
			10	$path(b,c)$ (if $?u2/b, ?v2/c$)	$\rightarrow\mathcal{E}(8)$
				$\Uparrow$ caused by justification (7)	
7	$arc(?x1,?v1)$	$\surd$(1) if $?x1/a, ?v1/b$	10	$path(?v1,c)$	

11	$arc(?x1,?v1) \wedge path(?v1,c)$	$\wedge\mathcal{I}$(7,10)
12	$path(?x1,c)$ (if $?u1/?x1, ?w1/c$)	$\rightarrow\mathcal{E}(6)$
13	$\exists x[path(x,c)]$	$\exists\mathcal{I}(12)$

7 Future direction 2: Reasoning about Programs

We are also working on enhancing pandora with Hoare logic for reasoning about programs. A clear simple account of Hoare logic with its semantic justification was given by Winskel [14] and formalised in the generic theorem prover Isabelle [6] by Nipkow et al [7, 10]. We have done some work on adapting the formalism to pandora style but have not yet started the java implementation of the upgraded tool.

The Hoare rules as used in the Isabelle theory are:

skip : $\vdash \{P\}SKIP\{P\}$

assg : $\vdash \{\lambda s.P(s[x \mapsto as])\}x = a\{P\}$

semi : $[\![\vdash \{P\}c\{Q\}; \vdash \{Q\}d\{R\}]\!] \Longrightarrow \vdash \{P\}c;d\{R\}$

if : $[\![\vdash \{\lambda s.Ps\&bs\}c\{Q\}; \vdash \{\lambda s.Ps\&\neg bs\}d\{Q\}]\!] \Longrightarrow \vdash \{P\}IF\ b\ THEN\ c\ ELSE\ d\{Q\}$

while : $\vdash \{\lambda s.Ps\&bs\}c\{P\} \Longrightarrow \vdash \{P\}WHILE\ b\ DO\ c\{\lambda s.Ps\&\neg bs\}$

where s varies over states, that is, functions from memory locations to values, and arithmetic expressions and boolean expressions are represented as functions from states to natural numbers and states to booleans respectively.

1	$sx = a \wedge sy = b$	pre
	$<$ *empty* $>$	
`P1`	`x = x + y; y = x − y; x = x − y`	
2	$sx = b \wedge sy = a$	$<$ *goal* $>$

We present a partial correctness proof of a simple swap program to give an indication of what we are trying to do. We start with the precondition, the program and the

$< goal >$ or hoped for postcondition as shown above. We assume that `;` associates to the right so that the outer connective is the second `;`. We can apply the rule `semi` by clicking on the program line `P1` then the rule `semi` to get the proof state below.

1	$sx = a \wedge sy = b$	pre
	$< empty >$	
2	$?P$	$< goal >$
P1	$\mathtt{x = x + y; y = x - y; x = x - y}$	
3	$?P$	ass
	$< empty >$	
P2	$\mathtt{x = x + y; y = x - y}$	
4	$?MID$	$< goal >$
5	$?MID$	ass
	$< empty >$	
P3	$\mathtt{x = x - y}$	
6	$?Q$	$< goal >$
7	$?Q$	semi P1 (3,4)(5,6)
	$< empty >$	
8	$sx = b \wedge sy = a$	$< goal >$

Here the $?P$ $?Q$ and $?MID$ are unknown formulas which can be unified with any formula using the `tick` rule. At this point rule `semi` could be applied again to line `P2` or we can use the `tick` rule to unify the formula at line 8 with $?Q$ and remove the redundant line. If we do the latter we can then use the `assg` rule on `P3` to bring the $< goal >$ at line 6 above the code line `P3` to get the proof state below.

1	$sx = a \wedge sy = b$	pre
	$< empty >$	
2	$?P$	$< goal >$
P1	$\mathtt{x = x + y; y = x - y; x = x - y}$	
3	$?P$	ass
	$< empty >$	
P2	$\mathtt{x = x + y; y = x - y}$	
4	$?MID$	$< goal >$
5	$?MID$	ass
	$< empty >$	
6	$sx - sy = b \wedge sy = a$	$< goal >$
P3	$\mathtt{x = x - y}$	
7	$sx = b \wedge sy = a$	assg P3 (6)
8	$sx = b \wedge sy = a$	semi P1 (3,4)(5,7)

Now we use the `tick` rule to unify the formula at line 6 with $?MID$. This causes the redundant line to be removed and the bottom box which is now complete to be greyed out. Next we must use the `semi` rule on `P2` and then we can proceed as above using `tick` to unify and `assg` to bring the goal above the code until we get up to the precondition and complete the proof. The resulting proof is shown below.

1	$sx = a \wedge sy = b$	pre
2	$sy = b \wedge sx = a$	TML(1)
`P1`	$\mathtt{x = x + y; y = x - y; x = x - y}$	
3	$sy = b \wedge sx = a$	ass
`P2`	$\mathtt{x = x + y; y = x - y}$	
4	$sy = b \wedge sx = a$	ass
5	$sy = b \wedge (sx + sy) - sy = a$	TMA(4)
`P3`	$\mathtt{x = x + y}$	
6	$sy = b \wedge (sx - sy) = a$	assg P3 (5)
7	$sy = b \wedge (sx - sy) = a$	ass
8	$sx - (sx - sy) = b \wedge (sx - sy) = a$	TMA(7)
`P4`	$\mathtt{y = x - y}$	
9	$sx - sy = b \wedge sy = a$	assg P4 (8)
10	$sx - sy = b \wedge sy = a$	semi P1 (4,6)(7,9)
11	$sx - sy = b \wedge sy = a$	ass
`P5`	$\mathtt{x = x - y}$	
12	$sx = b \wedge sy = a$	assg P5 (11)
13	$sx = b \wedge sy = a$	semi P1 (3,10)(11,12)

Note that when the rule `assg` was applied backwards at line 9 over program line `P4` and backwards at line 6 over program line `P3` the resulting expressions at lines 8 and 5 needed simplifying to get the lines 7 and 4 respectively. In the long term we hope to be able to add an arithmetic module to pandora to enable it to simplfy, or at least check the correctness of simplifications, itself. In order to get started we will initially build in a "trust me arithmetic" (TMA) rule. Applications of this rule will be accepted by the tool if some simple semantic plausibiliy checks are passed. Note also that in going backwards to the precondition from line 2, which like line 3 was obtained by unification with line 4, we use the `TML` rule which stands for "trust me logic" and will be used for simple logical checking.

For these ideas to scale up for use with larger programs we need the ability to collapse and expand parts of the proof so they take up less screen space when they are not being actively worked on. Also, undo/redo needs to be made smarter so that a small change required in a loop invariant does not cause all the work since the invariant was first given to be lost.

This is a useful challenge - simple enough to provide hope of progress but substantial enough to give us some confidence that good solutions will scale up to larger proof systems.

The big advantage of pandora over text based interactive theorem provers like Isabelle is that the student can see the proof emerge graphically and non-essential details like line numbers are automatically taken care of. Unlike the final proof produced by Isabelle, which is more naturally viewed as an "assembly code" program for generating a proof rather than a proof itself, the pandora proof is human readable. Its overall structure can be viewed at a glance and the detailed steps can easily be followed.

8 Related Work

Pandora is not the only teaching tool for first order natural deduction using Fitch-style boxes. Two others of which we are aware are Jape [5] and AProS [9, 12].

Jape was originally developed about the same time as pandora by Richard Bornat and Bernard Suffrin. It has recently been enhanced and although it does not currently have an e-tutor to give hints it does have the useful ability to look for first order counter-models. It includes some unification on formulas, for example applying the or introduction rule to the formula A gives $A \vee ?B$ and the unknown can later be unified with any formula. The interface allows the user to input their own problems, import valid formulas and save and print proofs. Jape silently carries out the `tick` rule whenever it can, even when unification is involved. From a teaching point of view we believe this is a draw-back as the students are not in control of the whole proof construction. Also, when this happens the proof state "jumps" which is visually confusing.

AProS is integrated into a complete teaching programme for online learning. AProS is used for a variety of courses so there are several different interfaces appropriate to the course; for example, the user is allowed to use De Morgan's laws on some problems but not on others. It does not allow the user to input their own problems as these are built in as part of the course. We understand that the next version of AProS will allow user input and that it is currently being enhanced with an e-tutor. The AProS project has an automated reasoning dimension incorporating a second tool which generates proofs and will be a basis for the e-tutor.

This paper is an extension of [3] presented at Salamanca where also AProS was presented. In pandora $\to\mathcal{E}$ backwards is currently: if $A \to B$ is in the data and B is a goal then that goal can be reduced to the goal A. This can be generalised to "$\to\mathcal{E}$ middle-out": given the data $A \to B$ and a goal G we get two new proof obligations (i) to prove A and (ii) to prove G with the additional assumption B. If G is B this reduces to our current $\to\mathcal{E}$ backwards and if A is part of the data it reduces to the usual $\to\mathcal{E}$ forwards. We much enjoyed and appreciated conversations with Sieg about his investigations into the intercalation calculi [13] where $\to\mathcal{E}$ middle-out is called $\to\downarrow$. We will implement $\to\mathcal{E}$ middle-out as an updated $\to\mathcal{E}$ backwards in the next version of pandora. Jape already includes this generalisation.

References

[1] K. Broda, S. Eisenbach and L. Kamara. Tools for Natural Deduction, *Proceedings of First Australian Conference on Computer Science Education*, ACSE96, CACM, pp 119-126, 1996.

[2] K. Broda, S. Eisenbach, H. Khoshnevisan, S. Vickers. *Reasoned Programming*, Prentice-Hall, 1994.

[3] K. Broda, J. Ma, G. Sinnadurai and A. Summers. Pandora: Natural Deduction made Easy, in [11], pp 11-14.

[4] W. Clocksin and C. Mellish. *Programming in Prolog: Using the ISO Standard*, Springer-Verlag 2003.

[5] `http://jape.comlab.ox.ac.uk:8080/jape/`

[6] `http://www.cl.cam.ac.uk/Research/HVG/Isabelle/`

[7] `http://www.cl.cam.ac.uk/research/hvg/Isabelle/dist/library/HOL/IMP/index.html`

[8] `http://www.doc.ic.ac.uk/pandora/`

[9] `http://www.phil.cmu.edu/projects/apros/`

[10] T. Nipkow. Winskel is (almost) Right: Towards a Mechanized Semantics Textbook, *Foundations of Software Technology and Theoretical Computer Science*, ed. V. Chandru and V. Vinay, Springer LNCS 1180, pp 180-192, 1996.

[11] *Proceedings of the second international congress on tools for teaching logic*, Eds: Maria Manzano, Belen Perez Lancho and Ana Gil, Salamanca, 2006.

[12] W. Sieg. STRATEGIC THINKING: web-based and dynamically tutored, in [11], pp 199-200.

[13] W. Sieg and J. Byrnes. Normal Natural Deduction Proofs (in classical logic), *Studia Logica* 60: pp 67-106, 1998.

[14] G. Winskel. *The Formal Semantics of Programming Languages*, MIT, 1993.

Received 22 December 2006