Teaching natural deduction in the right order with Natural Deduction Planner

Declan Thompson
The University of Auckland, New Zealand
declanthompsonnz@gmail.com

Jeremy Seligman
The University of Auckland, New Zealand
j.seligman@auckland.ac.nz

Abstract

We describe a strategy-based approach to teaching natural deduction using a notation that emphasises the order in which deductions are constructed, together with a LaTeX package and Java app to aid in the production of teaching resources and classroom demonstrations. Our approach is aimed at students with little exposure to mathematical method and has been developed while teaching undergraduate classes for philosophy students over the last ten years.

Keywords: Natural Deduction, Strategy, Proof Assistant.

1 Natural Deduction as a Creative Process

Teaching modern logic to students with little background in mathematics is notoriously hard. The philosophy student, adept at reading complex prose and composing artful essays is usually not well prepared for manipulating symbols and constructing rigorous proofs of theorems. Acquisition of at least the following three skills are needed.

The first is using the language of propositional and predicate logic to represent one's thoughts in formal notation and understand what has been written by others. This is usually achieved by learning to translate to and from natural language. Many resources are available.

The second is manipulating the symbols of formal notation according to precise rules. This is a basic skill necessary for almost all of logic, from applying mechanical

methods of argument evaluation to acquiring an appreciation of the autonomy of the syntactic realm, without which the major theoretical results of logical theory cannot be understood. It can be acquired by learning how to produce truth tables, truth trees, and in many other ways. Again many resources are available.

The third is reading and writing rigorous arguments, of the kind used in mathematics. This is a much more difficult skill to acquire, requiring mastery of the first two skills, and in addition, a level of mathematical maturity that is attained by mathematics students only after years of practice with algebra, geometry, analysis, etc. Consequently, this side of logic education is often neglected by philosophy undergraduate programmes. Although many introductory textbooks include some discussion of logical theory, such as soundness and completeness, the emphasis is on understanding the theorems rather than developing the skills to prove them. Few are aimed directly at acquiring the skill of creating proofs from scratch.[1]

One solution is to require logic students to take a substantial number of courses in mathematics, so that they acquire the necessary skills in the same way as mathematics students. In the long run, a broad experience with mathematical methods is certainly useful for research in logic, if not absolutely essential. But the huge gap that must be filled is daunting and dispiriting for most philosophy students, most of whom decide that it is just too big to breach.

Is there another solution? The obvious candidate is to teach students the skill of rigorous argumentation using the very formalisms that they have already learned: propositional and predicate logic. From a theoretical perspective, we know that our various systems of deduction can duplicate all that a mathematics student learns by a much more indirect and less explicit route. Why then is it so difficult for a philosophy student who has learned a formal system of deduction to transfer her skills to the production of informally rigorous arguments of the kind needed for progress in her subject?

It is generally recognised that axiomatic systems, while elegant and theoretically parsimonious, are wholly inappropriate for learning deduction. Instead, most logic programmes for philosophy students include some system of natural deduction, in which axioms are replaced by rules which mirror patterns of reasoning used in natural language argumentation. In the classic approach of Irving Copi, numerous rules are added, so as to capture as many such patterns as possible.[2] Yet there is often an insufficient level of attention to any systematic discussion of the process of *creating*

[1] A notable exception is "How to Prove It: A Structured Approach" [1, 2]. There are also countless introductions to mathematical method, e.g. "The Nuts and Bolts of Mathematical Proof" [3], "How to Read and Do Proofs: An Introduction to Mathematical Thought Processes" [4].

[2] "Introduction to Logic" [5] is now in its 14th edition. Many other textbooks on natural deduction employ a similarly lengthy list of rules.

deductions. Typically, students are given an introduction to the rules, motivated by their natural language correlates, a few examples of complete deductions, and are then left to fend for themselves on a large number of exercises, with the hope that they will develop their own strategies by trial and error.

An alternative is to teach the *strategies* of creation explicitly. As well as helping students to learn formal deduction, these are the strategies that will prepare the student for the harder task of creating rigorous informal arguments of the kind needed to do postgraduate work in logic, and which mathematics students learn implicitly through their application to a wide range of mathematical topics. Teachers of natural deduction in the traditional style may be fully aware of this point, but the effective learning of explicit strategy is made almost impossibly hard by several factors.

The first is simply the number of rules used by logic textbooks aimed at mirroring patterns in informal reasoning, which include both proof by cases (Disjunction Elimination, $\vee\mathbf{E}$), and Disjunctive Syllogism, if not also Constructive Dilemma. While each of these is relatively easy to explain in isolation, the more rules, the harder it is to master their strategic interactions, which the student must consider when creating her own deductions.

A second, related factor is the lack of structure to the set of rules. From the perspective of teaching strategy, one would prefer a simpler set of rules, organised in a way that corresponds to patterns of use in the creation of deductions, and exactly this is provided by Gentzen's original system, which uses the idea of introduction and elimination to expose the structure and symmetry of proof. More details will be given in Section 2, but for now a brief summary of the main points will suffice. Firstly, the fact that the intuitionistic fragment of the system has only a pair of rules for each logical operator allows one to develop general strategies: one for Introduction rules and one for Elimination rules, concerning the management of resources and simplification of goals. Moreover, an orthogonal classification of rules allows us to distinguish between cases in which a choice is required (e.g., $\vee\mathbf{I}$ and $\exists\mathbf{I}$) and those that are 'automatic', in the sense that they can be applied without the need for further choice. Even the symmetry-breaking oddity of the non-intuitionistic rule $\neg\neg\mathbf{E}$, which can be applied to any conclusion, raises an important strategic question: how to manage the creative steps of deduction? And this leads to an explicit discussion of back-tracking in problem solving and the need to recognise dead-ends. While such matters of strategy are implicit in more complicated systems, they are highlighted in systems in which the number of rules is small and well-balanced.

Standard presentations of Gentzen-style natural deduction, such as those of Fitch or Lemon, still have an important deficiency. Designed for reading rather than

writing, the argument is displayed with the premises at the top, the conclusion at the bottom and with each line justified by lines higher up on the page, according to a formal rule. This makes the process of checking the correctness of the deduction relatively easy, but the process of generating the deduction itself unnecessarily hard.

1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
3.	$\neg p$	**Ass**
4.	p	$2, \wedge\mathbf{E}$
5.	$\bot$	$3, 4, \neg\mathbf{E}$
6.	$\neg q$	**Ass**
7.	q	$2, \wedge\mathbf{E}$
8.	$\bot$	$6, 7, \neg\mathbf{E}$
9.	$\bot$	$1, 3\text{-}5, 6\text{-}8, \vee\mathbf{E}$
10.	$\neg(p \wedge q)$	$2\text{-}9, \neg\mathbf{I}$

*On the left is a correct deduction using a version of Gentzen's rules. Hypothetical reasoning is indicated by marking the assumption (***Ass***) and a vertical bracket ending below the hypothetical conclusion. The symbol $\bot$ is used to mark a contradiction.*

Information about the process of *creating* the deduction is lost in this representation, which wrongly suggests that it was written from top to bottom, starting with the premises and ending in the conclusion. (One of the most common mistakes made by students is to follow this order.) There is no record of the strategies used to construct the deduction; no record even of the order in which it was constructed. The student who fails to produce her own deduction of $\neg(p \wedge q)$ from $(\neg p \vee \neg q)$ will not learn much from looking at the above solution.

If we were to display a full sequence of steps leading to the creation of this deduction, we might write the following:

1.	$(\neg p \vee \neg q)$	**Prem**
10.	$\neg(p \wedge q)$	

$\neg\mathbf{I}$ $\rightsquigarrow$

1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
9.	$\bot$	
10.	$\neg(p \wedge q)$	$2\text{-}9, \neg\mathbf{I}$

$\vee\mathbf{E}$ $\rightsquigarrow$

1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
3.	$\neg p$	**Ass**
5.	$\bot$	
6.	$\neg q$	**Ass**
8.	$\bot$	
9.	$\bot$	$1, 3\text{-}5, 6\text{-}8, \vee\mathbf{E}$
10.	$\neg(p \wedge q)$	$2\text{-}9, \neg\mathbf{I}$

$\neg\mathbf{E}, \wedge\mathbf{E}$ $\rightsquigarrow$

1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
3.	$\neg p$	**Ass**
4.	p	$2, \wedge\mathbf{E}$
5.	$\bot$	$3, 4, \neg\mathbf{E}$
6.	$\neg q$	**Ass**
8.	$\bot$	
9.	$\bot$	$1, 3\text{-}5, 6\text{-}8, \vee\mathbf{E}$
10.	$\neg(p \wedge q)$	$2\text{-}9, \neg\mathbf{I}$

$\neg\mathbf{E}, \wedge\mathbf{E}$ $\rightsquigarrow$

1.	$(\neg p \vee \neg q)$	**Prem**
2.	$(p \wedge q)$	**Ass**
3.	$\neg p$	**Ass**
4.	p	$2, \wedge\mathbf{E}$
5.	$\bot$	$3, 4, \neg\mathbf{E}$
6.	$\neg q$	**Ass**
7.	q	$2, \wedge\mathbf{E}$
8.	$\bot$	$6, 7, \neg\mathbf{E}$
9.	$\bot$	$1, 3\text{-}5, 6\text{-}8, \vee\mathbf{E}$
10.	$\neg(p \wedge q)$	$2\text{-}9, \neg\mathbf{I}$

This is much too cumbersome for practical use in textbooks, and leaves the assignment of line numbers somewhat mysterious. How to know the deduction will use ten lines? But a simple change in notation can help. Instead of numbering the lines of the deduction from top to bottom, we number them in the order they were created. The above sequence can then be represented with just one deduction, as shown:

1.	$(\neg p \vee \neg q)$	**Prem**
3.	$(p \wedge q)$	**Ass**
5.	$\neg p$	**Ass**
9.	p	$3, \wedge\mathbf{E}$
6.	$\perp$	$5, 9, \neg\mathbf{E}$
7.	$\neg q$	**Ass**
10.	q	$3, \wedge\mathbf{E}$
8.	$\perp$	$7, 10, \neg\mathbf{E}$
4.	$\perp$	$1, 5\text{-}6, 7\text{-}8, \vee\mathbf{E}$
2.	$\neg(p \wedge q)$	$3\text{-}4, \neg\mathbf{I}$

First, the premise and conclusion are written as lines 1 and 2, with a generous space between. We then apply $\neg\mathbf{I}$ to the conclusion to get a hypothetical deduction with assumption $(p \wedge q)$ on line 3 and conclusion $\perp$ on line 4. Next, we apply $\vee\mathbf{E}$ to line 1 to get two nested hypothetical deductions, from $\neg p$ on line 5 to $\perp$ on line 6, and from $\neg q$ on line 7 to $\perp$ on line 8. The first of these is completed using $\neg\mathbf{I}$ to get p on line 9 (justified by $\wedge\mathbf{E}$ from line 3). The second is completed similarly, with line q on line 10. In this way, the line numbers match the order of construction of the deduction precisely, which is thereby emphasised to students as they create it.

The discipline of numbering in the order a deduction is created helps students (and instructors) to think strategically. The goal is to provide a justification for the conclusion given the resources in the premises, and seen this way deduction is just planning how to use the resources to satisfy a goal. While this is a familiar idea in automated reasoning research, it rarely enters the classroom. By using the above system of numbering, students cannot avoid thinking in this strategic way and learning that introduction rules serve to split the goal into subgoals, whereas elimination rules deploy resources. Strategic concepts such as back-tracking, management of decision points, and an awareness of risk are brought to the fore. Certain rules, such as Disjunction Introduction are seen as “choice rules" to be used with caution and postponed as long as possible, whereas others, such as Implication Introduction are “automatic" - they can and should be used immediately with no risk of having to undo.

The use of a new notation has the disadvantage that teaching resources, especially solutions to exercises, have to be produced from scratch. And it was to aid in

this that we decided to produce both a LaTeX package for formatting our deductions easily, and a Java app to aid in the generation of both LaTeX code and various other formats for classroom demonstration.

2 Teaching Strategies, Explicitly

At any stage of creating a deduction, one has to decide which rule to apply next. One of the great advantages of using Gentzen-style systems of natural deduction is both the relative paucity of rules and the direction one has from the syntactic structure of premises and conclusion as to which rule to use. For intuitionistic natural deduction, whose rules in Fitch-style are shown in Table 1, exactly one rule applies to each formula and this fact makes it easy to teach specific strategies.

2.1 Goals and Resources

The first distinction we teach is that between "goals" and "resources". When a natural deduction problem is first written down in Fitch-style, it consists of a number of lines at the top (the premises) and a line at the bottom (the conclusion). For example:

1.	$(p \wedge q)$	**Prem**
2.	$(p \to r)$	**Prem**
	$\vdots$	
3.	$(p \to (q \wedge r))$	

Here, the *goal* is to prove line 3, $(p \to (q \wedge r))$ from *resources* on lines 1 and 2. We can write this explicitly in sequent notation as

$$(p \wedge q), (p \to r) \implies (p \to (q \wedge r))$$

Students are not taught sequent notation initially. Instead, we discuss the role of goals and resources by referring directly to Fitch-style deductions, written on the whiteboard or projected from a computer. Nonetheless, they are useful here as a way of making proof strategies explicit.

As a deduction progresses, one typically has multiple goals and resources. For example, in the deduction shown below, we have two remaining tasks:

Natural Deduction, NJ			
Conjunction Introduction $i \quad \phi$ $j \quad \psi$ $\Rightarrow \quad (\phi \wedge \psi) \quad i,j,\wedge\mathbf{I}$	Conjunction Elimination $i \quad (\phi \wedge \psi)$ $\Rightarrow \quad \phi \quad i,\wedge\mathbf{E}$ $\ldots.$ $i \quad (\phi \wedge \psi)$ $\Rightarrow \quad \psi \quad i,\wedge\mathbf{E}$	Implication Introduction $i \quad \phi \quad \mathbf{Ass}$ $\vdots$ $j \quad \psi$ $\Rightarrow \quad (\phi \to \psi) \quad i{-}j,\to\mathbf{I}$	Implication Elimination $i \quad (\phi \to \psi)$ $j \quad \phi$ $\Rightarrow \quad \psi \quad i,j,\to\mathbf{E}$
Disjunction Introduction $i \quad \phi$ $\Rightarrow \quad (\phi \vee \psi) \quad i,\vee\mathbf{I}$ $\ldots\ldots\ldots\ldots\ldots\ldots$ $i \quad \psi$ $\Rightarrow \quad (\phi \vee \psi) \quad i,\vee\mathbf{I}$	Disjunction Elimination $i \quad (\phi \vee \psi)$ $j \quad \phi \quad \mathbf{Ass}$ $\vdots$ $k \quad \theta$ $l \quad \psi \quad \mathbf{Ass}$ $\vdots$ $m \quad \theta$ $\Rightarrow \quad \theta \quad i,j{-}k,l{-}m,\vee\mathbf{E}$	Equivalence Introduction $i \quad \phi \quad \mathbf{Ass}$ $\vdots$ $j \quad \psi$ $k \quad \psi \quad \mathbf{Ass}$ $\vdots$ $l \quad \phi$ $\Rightarrow \quad (\phi \leftrightarrow \psi) \quad i{-}j,k{-}l,\leftrightarrow\mathbf{I}$	Equivalence Elimination $i \quad (\phi \leftrightarrow \psi)$ $j \quad \phi$ $\Rightarrow \quad \psi \quad i,j,\leftrightarrow\mathbf{E}$ $\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ $i \quad (\phi \leftrightarrow \psi)$ $j \quad \psi$ $\Rightarrow \quad \phi \quad i,j,\leftrightarrow\mathbf{E}$
Negation Introduction $i \quad \phi \quad \mathbf{Ass}$ $\vdots$ $j \quad \bot$ $\Rightarrow \quad \neg\phi \quad i{-}j,\neg\mathbf{I}$	Negation Elimination $i \quad \neg\phi$ $j \quad \phi$ $\Rightarrow \quad \bot \quad i,j\neg\mathbf{E}$	Falsum Introduction	Falsum Elimination $i \quad \bot$ $\Rightarrow \quad \phi \quad i,\bot\mathbf{E}$
Universal Introduction $i \quad \phi^x_a$ $\Rightarrow \quad \forall x\,\phi \quad i,\forall\mathbf{I}$ a is a parameter not in ϕ, nor any premises, nor active assumptions	Universal Elimination $i \quad \forall x\,\phi$ $\Rightarrow \quad \phi^x_t \quad i,\forall\mathbf{E}$ t is any term	Existential Introduction $i \quad \phi^x_t$ $\Rightarrow \quad \exists x\,\phi \quad i,\exists\mathbf{I}$ t is any term	Existential Elimination $i \quad \exists x\,\phi$ $j \quad \phi^x_a \quad \mathbf{Ass}$ $\vdots$ $k \quad \psi$ $\Rightarrow \quad \psi \quad i,j{-}k\exists\mathbf{E}$ a is a parameter not in ϕ, ψ, nor any premises, nor active assumptions

Table 1: The rules of intuitionistic natural deduction (NJ) in a standard presentation.

1.	$((p \wedge q) \rightarrow r)$	**Prem**
2.	$(q \vee \neg(p \wedge r))$	**Prem**
4.	p	**Ass**
6.	q	**Ass**
	$\vdots$	
7.	r	
8.	r	**Ass**
10.	q	**Ass**
11.	q	10
12.	$\neg(p \wedge r)$	**Ass**
	$\vdots$	
13.	q	
9.	q	2, 10-11, 12-13, $\vee$**E**
5.	$(q \leftrightarrow r)$	6-7,8-9, $\leftrightarrow$ **I**
3.	$(p \rightarrow (q \leftrightarrow r))$	4-5, $\rightarrow$ **I**

These can be represented explicitly by listing the resources and goal in sequent form:

From 1,2,4,6 to 7: $((p \wedge q) \rightarrow r), (q \vee \neg(p \wedge r)), p, q \implies r$

From 1,(2),4,8,12 to 13: $((p \wedge q) \rightarrow r), p, r, \neg(p \wedge r) \implies q$

Line 2 has been dropped as a resource for the second task because it has already been used in the $\vee$**E** used to justify line 9. Learning to identify the remaining goals of a deduction is relatively easy: they are simply the line which still lack a justification. Learning the available resources requires an understanding of the nesting of hypothesis lines, but is more apparent in the process of constructing the deduction than in reading it.

2.2 Automatic Rules

Our second main distinction is between "automatic" application of rules and applications that involve a "choice". A paradigmatic example of an automatic rule application is any use of $\rightarrow$ **I**. Here, the goal is an implication and the deduction is transformed by adding its antecedent as a resource and making its succedent the new goal:

1.	$(p \wedge q)$	**Prem**
2.	$(p \rightarrow r)$	**Prem**
4.	p	**Ass**
	$\vdots$	
5.	$(q \wedge r)$	
3.	$(p \rightarrow (q \wedge r))$	4-5, $\rightarrow$ **I**

We can represent an application of $\rightarrow$ **I** in sequent form as:

$\rightarrow$ **I**	$\leftrightarrow$ **I**	$\neg$**I**
$\dfrac{\Gamma,\, \phi \implies \psi}{\Gamma \implies (\phi \rightarrow \psi)}$	$\dfrac{\Gamma,\, \phi \implies \psi \qquad \Gamma,\, \psi \implies \phi}{\Gamma \implies (\phi \leftrightarrow \psi)}$	$\dfrac{\Gamma,\, \phi \implies \bot}{\Gamma \implies \neg A}$
$\exists$**E** (a new)	$\vee$**E**	$\forall$**I** (a new)
$\dfrac{\Gamma,\, \phi^x_a \implies \psi}{\Gamma,\, \exists x\, A \implies \psi}$	$\dfrac{\Gamma,\, \phi \implies \theta \qquad \Gamma,\, \psi \implies \theta}{\Gamma,\, (\phi \vee \psi) \implies \theta}$	$\dfrac{\Gamma \implies \phi^x_a}{\Gamma \implies \forall x\, \phi}$
$\bot$**E**	$\wedge$**I**	$\wedge$**E**
$\dfrac{}{\Gamma,\, \bot \implies \phi}$	$\dfrac{\Gamma \implies \phi \qquad \Gamma \implies \psi}{\Gamma \implies (\phi \wedge \psi)}$	$\dfrac{\Gamma,\, \phi,\, \psi \implies \theta}{\Gamma,\, (\phi \wedge \psi) \implies \theta}$

Table 2: The automatic strategy rules of **NJ**

$$\frac{\Gamma,\, \phi \implies \psi}{\Gamma \implies (\phi \rightarrow \psi)}$$

Reading upwards, this means that the deduction task with goal $(\phi \rightarrow \psi)$ and resources Γ is replaced by the task with goal ψ and resources $\Gamma,\, \phi$. The rule application is *automatic* because of the equivalence of these two tasks: ψ is a consequence of $\Gamma,\, \phi$ if and only if $(\phi \rightarrow \psi)$ is a consequence of Γ. The application of the rule usually also results in a decrease in the complexity of the deduction, splitting up the formula $(\phi \rightarrow \psi)$ into its proper parts. Students are therefore encouraged to apply the rule of $\rightarrow$ **I** automatically, without the need to think strategically.

All the hypothetical rules are similarly automatic: $\rightarrow$ **I**, $\leftrightarrow$ **I**, $\neg$**I**, $\exists$**E** and $\vee$**E**. In addition, among the non-hypothetical rules, $\forall$**I**, $\bot E$, $\wedge$**I** and $\wedge$**E** are automatic. Sequent notation for these rules in shown in Table 2. Each of the automatic rules satisfies a similar equivalence: the sequent below the line is valid iff those above the line are all valid. Strategically, this means that nothing is lost by applying the rule. If the task below the line is completable, then it can be completed by applying the rule and moving to the task(s) above the line. There is no risk of failure.

The majority of rules in **NJ** are automatic and so students have a lot of guidance as to how to proceed. After a certain amount of practice, they can easily be trained to identify the automatic rules and apply them almost mechanically, giving them confidence in following the goal-resource methodology rather than simply forward chaining from the premises in an attempt to reach the conclusion. This breaking of

bad habits and providing a reliable framework within which more complex strategies can be developed is a large part of what we can do by teaching natural deduction "in the right order".

2.3 Choice and Procrastination

The automatic application of rules is contrasted with the application of rules that require a choice. A paradigmatic example of a choice rule is $\vee\mathbf{I}$. Faced with the goal of proving $(\phi \vee \psi)$, we have to decide which: ϕ or ψ. For example, in the problem on the left, an application of $\vee\mathbf{I}$ with choice p quickly leads to impasse (right):

1.	$((p \to r) \leftrightarrow (r \to q))$	**Prem**	1.	$((p \to r) \leftrightarrow (r \to q))$	**Prem**
2.	r	**Prem**	2.	r	**Prem**
	$\vdots$		7.	r	**Ass**
				$\vdots$	
			8.	q	
			5.	$(r \to q)$	7-8, $\to$ **I**
			6.	$(p \to r)$	1, 5, $\leftrightarrow$ **E**
				$\vdots$	
			4.	p	
3.	$(p \vee q)$		3.	$(p \vee q)$	4, $\vee$**I**

Students are presented with examples and exercises like this one to show that a real choice is made and that one must recognise when one is stuck so as to backtrack to the last choice. In this case, one can simply make the other choice, q and complete the deduction:

1.	$((p \to r) \leftrightarrow (r \to q))$	**Prem**
2.	r	**Prem**
7.	p	**Ass**
8.	r	2
5.	$(p \to r)$	7-8, $\to$ **I**
6.	$(r \to q)$	1, 5, $\leftrightarrow$ **E**
4.	q	6, 2, $\to$ **E**
3.	$(p \vee q)$	4, $\vee$**I**

In sequent notation, the choice is displayed vividly as a choice between two rules:

$$\frac{\Gamma \implies \phi}{\Gamma \implies (\phi \vee \psi)} \qquad \frac{\Gamma \implies \psi}{\Gamma \implies (\phi \vee \psi)}$$

Neither shares the equivalence of the automatic rules. It is possible for one or more of sequents above the line to be invalid (and so not provable) even if the sequent below the line is provable, by some other means. So in fact, the situation is worse than merely choosing between ϕ or ψ when trying to prove $(\phi \vee \psi)$. It may be that neither choice works:

1.	$(r \vee s)$	**Prem**
2.	$(r \to p)$	**Prem**
3.	$\neg(s \wedge q)$	**Prem**
4.	$(p \vee \neg q)$	

Here, applying $\vee$**I** at step 5 is no good no matter which disjunct we choose:

1.	$(r \vee s)$	**Prem**
2.	$(r \to p)$	**Prem**
3.	$\neg(s \wedge q)$	**Prem**
6.	r	**Ass**
7.	p	2, 6, $\to$ **E**
8.	s	**Ass**
	$\vdots$	
12.	q	
10.	$(s \wedge q)$	8, 12, $\wedge$**I**
11.	$\bot$	3, 10, $\neg$**E**
9.	p	11, $\bot$**E**
5.	p	1, 6-7, 8-9, $\vee$**E**
4.	$(p \vee \neg q)$	5, $\vee$**I**

1.	$(r \vee s)$	**Prem**
2.	$(r \to p)$	**Prem**
3.	$\neg(s \wedge q)$	**Prem**
6.	q	**Ass**
9.	r	**Ass**
14.	p	2, 9, $\to$ **E**
	$\vdots$	
13.	s	
10.	$(s \wedge q)$	13, 6, $\wedge$**I**
11.	s	**Ass**
12.	$(s \wedge q)$	11, 6, $\wedge$**I**
8.	$(s \wedge q)$	1, 9-10, 11-12, $\vee$**E**
7.	$\bot$	3, 8, $\neg$**E**
5.	$\neg q$	6-7, $\neg$**I**
4.	$(p \vee \neg q)$	5, $\vee$**I**

The only solution is to postpone the choice (leaving the $\vee$**I** to steps 9 and 10):

1.	$(r \vee s)$	**Prem**
2.	$(r \to p)$	**Prem**
3.	$\neg(s \wedge q)$	**Prem**
5.	r	**Ass**
9.	p	2, 5, $\to$ **E**
6.	$(p \vee \neg q)$	9, $\vee$**I**
7.	s	**Ass**
11.	q	**Ass**
13.	$(s \wedge q)$	7, 11, $\wedge$**I**
12.	$\bot$	3, 13, $\neg$**E**
10.	$\neg q$	11-12, $\neg$**I**
8.	$(p \vee \neg q)$	10, $\vee$**I**
4.	$(p \vee \neg q)$	1, 5-6, 7-8, $\vee$**E**

Teaching how to manage one's choices and to realise that the best strategy is often to procrastinate is a core part of our approach to natural deduction. We strongly emphasise the distinction between automatic and choice rules, so as to highlight when choices are made and the nature of the choice. The pure choice rules are $\vee$**I** and $\exists$**I**, as shown in Table 3. The latter requires a choice of term t with which to replace the bound variable x of $\exists x\, \phi$. This should be taken from terms already occurring in the deduction, or if there is none, a new individual constant. In the

$\vee\mathbf{I_l}$	$\vee\mathbf{I_r}$	$\exists\mathbf{I}$ (t old/first)
$\dfrac{\Gamma \implies \phi}{\Gamma \implies (\phi \vee \psi)}$	$\dfrac{\Gamma \implies \phi}{\Gamma \implies (\phi \vee \psi)}$	$\dfrac{\Gamma \implies \phi^x_t}{\Gamma \implies \exists x\, \phi}$

Table 3: Pure choice rules of **NJ**

latter case, since any new constant will do, students may be tricked into thinking that no choice is involved, but that's not so. Just as in $\vee\mathbf{I}$, there are examples in which only the avoidance of any choice (procrastination) will enable a solution:

1. $\exists x\,(Fx \wedge Gx)$	**Prem**	
⋮		
2. $\exists x\, Fx$		

1. $\exists x\,(Fx \wedge Gx)$	**Prem**	
4. $(Fb \wedge Gb)$	**Ass**	
6. Fb	4, $\wedge\mathbf{E}$	
7. Gb	4, $\wedge\mathbf{E}$	
⋮		
5. Fa		
3. Fa	1, 4-5, $\exists\mathbf{E}$	
2. $\exists x\, Fx$	3, $\exists\mathbf{I}$	

1. $\exists x\,(Fx \wedge Gx)$	**Prem**	
3. $(Fa \wedge Ga)$	**Ass**	
5. Fa	3, $\wedge\mathbf{E}$	
4. $\exists x\, Fx$	5, $\exists\mathbf{I}$	
2. $\exists x\, Fx$	1, 3-4, $\exists\mathbf{E}$	

The initial problem, shown on the left, is not solved by $\exists\mathbf{I}$, shown in the middle, despite there being no choice of instantiating term; only by procrastination can the deducting be completed (right).

2.4 Rules of Deduction vs Rules of Strategy

While the distinction between automatic application of rules and those that require choice management lines up with the natural deduction rules in all the cases considered above, the three remaining rules, of $\rightarrow\mathbf{E}$, $\leftrightarrow\mathbf{E}$ and $\neg\mathbf{E}$ have varying strategic properties depending on the context. Applications of $\rightarrow\mathbf{E}$, for example, can be split into three cases, indicated below:

1.	$(\phi \rightarrow \psi)$	
2.	ϕ	
	⋮	
3.	θ	

1.	$(\phi \rightarrow \psi)$	
	⋮	
2.	ψ	

1.	$(\phi \rightarrow \psi)$	
	⋮	
2.	θ	

1.	$(\phi \rightarrow \psi)$	
2.	ϕ	
4.	ψ	$1, 2, \rightarrow\mathbf{E}$
	⋮	
3.	θ	

1.	$(\phi \rightarrow \psi)$	
	⋮	
3.	ϕ	
2.	ψ	$1, 3, \rightarrow\mathbf{E}$

1.	$(\phi \rightarrow \psi)$	
	⋮	
3.	ϕ	
4.	ψ	$1, 3, \rightarrow\mathbf{E}$
	⋮	
2.	θ	

Top left is an instance of *modus ponens*, in which both $(\phi \to \psi)$ and ϕ are available resources and the inference to ψ (shown below, bottom left) is automatic. Top middle is an instance of *reverse modus ponens*, in which $(\phi \to \psi)$ is a resource and the goal is ψ, and the replacement of this goal by ϕ (shown below, bottom middle) is a matter of choice; ϕ may not be provable even if ψ is by another route. Finally, when neither ϕ is a resource nor ψ a goal (top right), the only way of using the resource $(\phi \to \psi)$ is to take a *bold step* (bottom right), which is again a choice that may lead to impasse.

The introduction of this terminology ("modus ponens", "reverse modus ponens" and "bold step") helps students distinguish between rules of strategy and the actual rules of logic as defined in the system of deduction. A complete list, in sequent notation is given in Table 4. To repeat: the sequent style is not used in the classroom at this level. We teach all of these strategies through explicit examples. NDP aids significantly in this process, since the mechanics of applying the rules is performed by the software, allowing the student (or the instructor using the software) to focus on strategic matters. For example, the strategic rule of $\perp$ *bold step* is illustrated by an example like this:

1.	$(p \vee q)$	**Prem**
2.	$\neg(p \wedge r)$	**Prem**
3.	$(q \to s)$	**Prem**
5.	r	**Ass**
7.	p	**Ass**
	$\vdots$	
8.	s	
9.	q	**Ass**
10.	s	3, 9, $\to$ **E**
6.	s	1, 7-8, 9-10, $\vee$**E**
4.	$(r \to s)$	5-6, $\to$ **I**

The remaining task is $\neg(p \wedge r)$, $(q \to s)$, r, $p \implies s$. We have two complex resources available: $\neg(p \wedge r)$ and $(q \to s)$. Using $(q \to s)$ would involve reverse modus ponens, which turns out not to work. The only hope is to use $\neg(p \wedge r)$ which requires $\neg$**E**. But we have neither $(p \wedge r)$ as a resource (EFQ) nor $\perp$ as a goal ($\perp$ intro) so we have to perform a $\perp$ bold step:

Automatic	Choice	
$\rightarrow$ **E** (MP)	$\rightarrow$ **E** (reverse MP)	$\rightarrow$ **E** (bold step)
$\dfrac{\Gamma,\ \phi,\ \psi \implies \theta}{\Gamma,\ (\phi \rightarrow \psi),\ \phi \implies \theta}$	$\dfrac{\Gamma,\ (\phi \rightarrow \psi) \implies \phi}{\Gamma,\ (\phi \rightarrow \psi) \implies \psi}$	$\dfrac{\Gamma,\ (\phi \rightarrow \psi) \implies \phi \qquad \Gamma,\ \psi \implies \theta}{\Gamma,\ (\phi \rightarrow \psi) \implies \theta}$
$\leftrightarrow$ **E** (L-R MP)	$\leftrightarrow$ **E** (L-R reverse MP)	$\leftrightarrow$ **E** (L-R bold step)
$\dfrac{\Gamma,\ \phi,\ \psi \implies \theta}{\Gamma,\ (\phi \leftrightarrow \psi),\ \phi \implies \theta}$	$\dfrac{\Gamma,\ (\phi \leftrightarrow \psi) \implies \phi}{\Gamma,\ (\phi \leftrightarrow \psi) \implies \psi}$	$\dfrac{\Gamma,\ (\phi \leftrightarrow \psi) \implies \phi \qquad \Gamma,\ \psi \implies \theta}{\Gamma,\ (\phi \leftrightarrow \psi) \implies \theta}$
$\leftrightarrow$ **E** (R-L MP)	$\leftrightarrow$ **E** (R-L reverse MP)	$\leftrightarrow$ **E** (R-L bold step)
$\dfrac{\Gamma,\ \psi,\ \phi \implies \theta}{\Gamma,\ (\phi \leftrightarrow \psi),\ \psi \implies \theta}$	$\dfrac{\Gamma,\ (\phi \leftrightarrow \psi) \implies \psi}{\Gamma,\ (\phi \leftrightarrow \psi) \implies \phi}$	$\dfrac{\Gamma,\ (\phi \leftrightarrow \psi) \implies \psi \qquad \Gamma,\ \phi \implies \theta}{\Gamma,\ (\phi \leftrightarrow \psi) \implies \theta}$
$\neg$**E** (EFQ)	$\rightarrow$ **E** ($\perp$ intro)	$\rightarrow$ **E** ($\perp$ bold step)
$\dfrac{\dfrac{}{\Gamma,\ \perp \implies \theta}}{\Gamma,\ \neg\phi,\ \phi \implies \theta}$	$\dfrac{\Gamma \implies \phi}{\Gamma,\ \neg\phi \implies \perp}$	$\dfrac{\Gamma \implies \phi \qquad \dfrac{}{\Gamma,\ \perp \implies \theta}}{\Gamma,\ \neg\phi \implies \theta}$

Table 4: Mixed rules of **NJ**

1.	$(p \vee q)$	**Prem**
2.	$\neg(p \wedge r)$	**Prem**
3.	$(q \rightarrow s)$	**Prem**
5.	r	**Ass**
7.	p	**Ass**
	$\vdots$	
11.	$(p \wedge r)$	
12.	$\perp$	2, 11, $\neg$**E**
8.	s	12, $\perp$**E**
9.	q	**Ass**
10.	s	3, 9, $\rightarrow$ **E**
6.	s	1, 7-8, 9-10, $\vee$**E**
4.	$(r \rightarrow s)$	5-6, $\rightarrow$ **I**

The deduction is then easily completed with $\wedge$**I**.

2.5 NK and beyond

This is not the place to give a complete summary of our teaching methodology for natural deduction. We aim for the students to achieve competency in **NJ** within 3 weeks after which they have a test. After that, they are introduced to Classical Natural Deduction (**NK**), with the addition of Double Negation Elimination ($\neg\neg$**E**). This is a game-changer. **NJ** still provides the everyday framework within which deductions can be created but now there is a wildcard. Whereas in **NJ** every formula, whether resource or goal, has its own rule, $\neg\neg$**E** can be applied to any goal. Well, even then, our distinction between automatic and choice provides a useful heuristic: $\neg\neg$**E** is never needed for automatic goals.

$\neg\neg$**E** provides the opportunity to introduce a new strategic concept: that of looping. Albert Einstein once defined insanity as doing the same thing and expecting a different result. We use this idea as encouragement: doing a different thing and expecting a different result is not necessarily insane. The classic example of this is the natural deduction proof of excluded middle:

3.	$\neg(p \vee \neg p)$	**Ass**
7.	p	**Ass**
	$\vdots$	
8.	$\bot$	
6.	$\neg p$	7-8, $\neg$**I**
5.	$(p \vee \neg p)$	6, $\vee$**I**
4.	$\bot$	3, 5, $\neg$**E**
2.	$\neg\neg(p \vee \neg p)$	3-4, $\neg$**I**
1.	$(p \vee \neg p)$	2, $\neg\neg$**E**

At this point in the proof you are trying to prove $\bot$, again. At step 4 your goal was also to prove $\bot$. So is this looping insanity? No, because something has changed: you now have line 7, p, as an additional resource, and a solution is only a step away. Contrast this with the following:

3.	$\neg(p \vee \neg p)$	**Ass**
	$\vdots$	
7.	$(p \vee \neg p)$	
8.	$\bot$	3, 7, $\neg$**E**
6.	p	8, $\bot$**E**
5.	$(p \vee \neg p)$	6, $\vee$**I**
4.	$\bot$	3, 5, $\neg$**E**
2.	$\neg\neg(p \vee \neg p)$	3-4, $\neg$**I**
1.	$(p \vee \neg p)$	2, $\neg\neg$**E**

Here we are trying again to prove $(p \vee \neg p)$. It's the third time this has come up: once on line 1 and again on line 5. But this time we really are looping: for the goal on line 5, our only resource was $\neg(p \vee \neg p)$ on line 3, and again for the goal on line 7, we have only this as a resource. Conclusion: insanity, backtrack.

From **NK** we move on to identity, and consider the complexity of $\forall$**E** and $\exists$**I** in a language with complex terms. This requires a new approach to proof search and introduces various concepts associated with pattern matching. And finally, we move to formal arithmetic, using natural deduction also as a way of teaching mathematical induction. Strategically, this is also interesting because of the need for lemmas in arithmetic. That is another teaching moment on the limits of syntax-guided proof methods.

In summary, natural deduction taught in this way, provides both a training ground for those who want to be able to create their own proofs, but also a wonderful case study of general strategic reasoning, including such topics as the balance between automatic moves and those that require management of choices, the virtues of procrastination, attention to goals and resources, the advantages and disadvantages of too much power ($\neg\neg$**E**), avoiding insanity, and the need for genuine creativity.

3 Natural Deduction Planner

Efficiently creating large numbers of typeset sample deductions can be a daunting prospect. On pen and paper, even a challenging proof can be completed within minutes. However typesetting a proof in software such as LaTeX requires a great deal more effort. With custom packages, structural features such as the scope lines used above can be automated well, but the task of inputting formulas is still cumbersome. Where the hand can draw any symbol at much the speed of any other, typesetting special characters often requires lengthy commands. We began development of a proof assistant software application with the primary goal of overcoming these difficulties, but the result is useful in many more ways than typesetting. We call the result the Natural Deduction Planner (NDP). It generates LaTeX code for use with a custom package.

Our interface essentially replicates the pen and paper proof process, using the same layout and notation of Gentzen's system, as above. Users input sequents using a set of special characters available onscreen. No special formatting (such as prefix notation) is required - a correctly inputted sequent appears as it would on the page. A range of proof systems are available, such as **NJ**, **NK** and Peano Arithmetic. Proofs appear graphically onscreen exactly as they would be typeset.

NDP is similar to the Proof Developer tool created by Daniel Velleman.[3] The interfaces are very alike, but where Proof Developer focusses on informal proof writing, NDP is concerned with formal deductions. Both approaches use ideas of strategy, goals and resources to conduct proofs. Another similar tool is PANDA, developed at the Institut de Recherche en Informatique de Toulouse.[4] PANDA uses a proof tree style, rather than the Fitch-style calculus implemented in NDP.

3.1 Basic Use

The primary use of the Natural Deduction Planner is in the creation of deductions. Computer based deduction systems immediately present a challenge in their particular uses of special symbols. Without a usable interface, inputting formulas can become unnecessarily tedious. For NDP, we found it very important that formulas would appear as they do when handwritten (or as similar as possible). To this end, simplified and easily typed symbol sets (like using `&`, `v` and `->` for $\wedge$, $\vee$ and $\rightarrow$, respectively) are not the solution. Instead, we implemented a special symbols panel' on the sequent input dialog. This can be seen in Figure 1. Users can input the special symbols by selecting them. A number of intuitive shortcuts (such as `Alt+>` for $\rightarrow$ and `Alt+a` for $\forall$) are also available.

The new proof dialog is designed to reflect the standard style of sequents. Premises are listed in the uppermost textbox, separated by commas. In the lower textbox, the conclusion is supplied. Correct bracket matching must be used, and the system indicates when this is (and is not) the case. Users must input a conclusion in order to begin the deduction, though premises are optional.

In the new proof dialog, users are also able to choose which ruleset to use. In Figure 1, **NJ** has been selected, and so the double negation elimination rule will not be available (though it can be activated during the course of the deduction). The choice of ruleset allows for fine-grained control over how the deduction can proceed. Custom rulesets can be defined, allowing an instructor to, for example, deactivate all quantifier rules. Students can then select the appropriate ruleset for a given exercise, and the deactivated rules will not appear, helping to reduce potential confusion from unknown rules.

Once the sequent has been inputted, the incomplete deduction (premises at the top, with a space before the conclusion) will appear in the main window. At each

[3]Proof Developer is a Java web applet built to accompany Velleman's textbook "How to Prove It" [1, 2] `http://www.cs.amherst.edu/~djv/pd/pd.html`

[4]PANDA is a Java application designed for teaching computer science students in logic, developed by Olivier Gasquet, François Schwarzentruber, and Martin Strecker `http://www.irit.fr/panda`

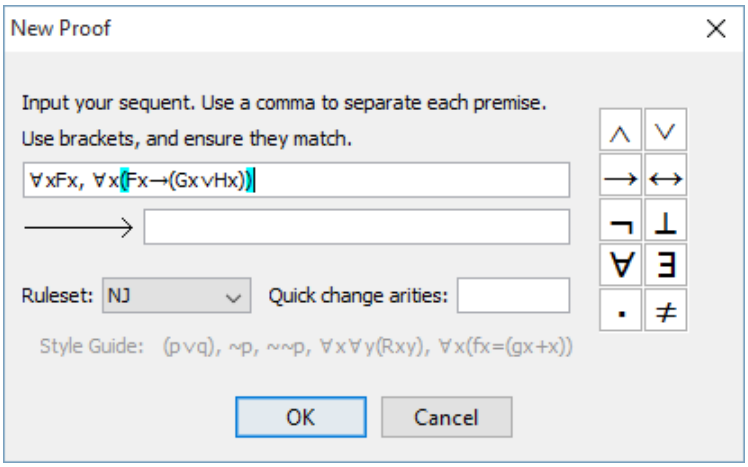

Figure 1: The input dialog. The special symbols panel is on the right and bracket match indication is shown. A conclusion is yet to be inputted.

stage of an NDP deduction, the user can apply any valid rules, and their outcome is immediately displayed. To apply a rule, a *current goal* must first be selected, which can be any unjustified line (indicated by a missing justification). This can be achieved by simply clicking on the line in question. Upon selecting a current goal, that line's introduction rules (if any) appear as button(s) next to it, and the line is highlighted. To apply one of these rules, a user can simply select that button. This takes care of introduction rules. For elimination rules, the user must also select a relevant *current resource*. Upon selection, the resource is highlighted and its elimination rules appear alongside the current goal. Again, to use a rule it is simply selected.

NDP checks the scope of lines available to the current goal. If a user selects a resource out of scope of the current goal, its elimination rules appear "greyed out", and cannot be applied. In this way, NDP ensures the user is following rules correctly. This behaviour is useful in teaching students the details of scope. An extension of this is shown in Figure 2. Here lines out of scope of the goal are greyed out, to explicitly show that they cannot be used. This feature can be activated from the Options menu. It does not prevent users from selecting out-of-scope resources, but does make it very clear how scope works.

Lines are numbered in the order of creation, again reinforcing the way the deduction is constructed. This is also useful when reaching a dead-end in the proof - the user can see exactly how they reached this point, and what will happen when they retrace their steps. Highlighting is used to indicate the current goal and resource

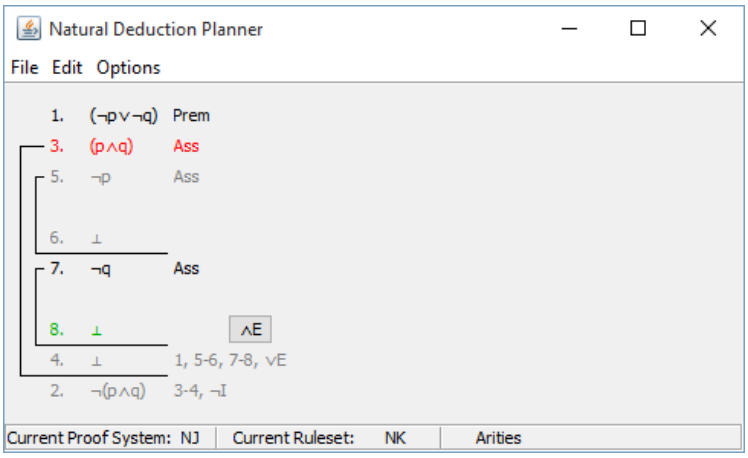

Figure 2: An incomplete proof showing current goal (green), current resource (red) and possible rules (∧**E**). All **NK** rules are available, but currently only **NJ** rules have been used.

and also serves to indicate any lines out of scope of the goal. Once all lines have been justified the proof is complete.

The requirement that users select a current goal before choosing a resource is fundamental to the operation of NDP. Even if the first step in the proof involves an elimination rule, the user must select a current goal. These explicit realisations of goal and resource help to reinforce the use of strategies in deductions. Rather than beginning with premises and working downwards, the user is encouraged to begin at the bottom of the deduction (the goal) and efficiently choose those resources which are needed. By breaking from a strictly linear approach, selecting goals encourages users to consider which available rules are automatic, and which require choice.

3.2 Types of Rules

The rules of natural deduction can be broadly assigned to one of three categories: those which are completely automatic, being applied the same way in every instance (for example, ∧**I**); those requiring some level of choice, as between two options (e.g. ∨**I**); and those which require further input, such as a choice of term (e.g. ∃**I**). Of course, further distinctions can be made between these categories, and the boundaries between them are fairly blurred (→ **E**, for example, would be in the first or second categories, depending on the case).

Knowing which rules fall into which category is very important for developing

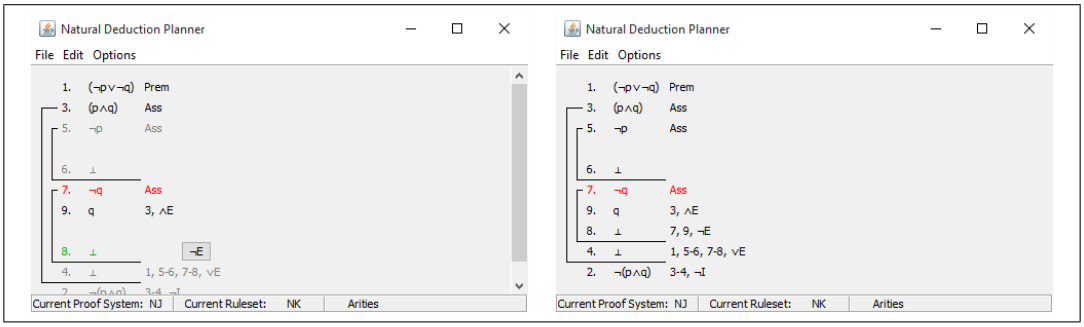

Figure 3: ¬**E** before application (left) and after (right). The user has clicked the button ¬**E** and done nothing else.

the skill of creating deductions effectively. The automatic rules should be applied quickly, and choice rules with caution. While this can be explained to students, often experience is the best way to reinforce just why we should choose one rule over another. As a direct consequence of the implementation of each rule in NDP, it becomes very obvious through use which rules are automatic and which are not. As such, practice with NDP can help students learn the best orders of rule application.

How does the behaviour of the types of rule differ in NDP? All rules are applied by selecting the appropriate button next to the current goal. They differ in what happens next.

For the automatic rules, the proof is immediately updated. Since there is no further information needed to apply the rule, no further interaction with the user is required. Figure 3 shows the application of the automatic ¬**E** rule. Negation elimination behaves differently depending on the context. Here, the goal is a contradiction ($\bot$) and the negand (p) can be found within scope. So all that needs to be given is a justification for $\bot$. In fact, this rule is *always* automatic, no matter the context. If the current goal is not $\bot$ then $\bot$ will be introduced, justified and the current goal justified by explosion ($\bot$**E**). If the negand does not already appear in the proof, it will be added as the new current goal above $\bot$.

A generally automatic rule which blurs the distinction between automation and choice is ∧**E**, which is applied to a current resource. If the current goal is one of the conjuncts, then it is automatically justified and nothing further happens. Otherwise, the user is presented with a (fairly trivial) choice: would you like to extract the first conjunct, the second conjunct or both? This is not a difficult choice - if in doubt we can choose both and get, at worst, a proof of one line longer. But this choice, presented to the user, reinforces the bigger picture. We're using strategy to move through the deduction, and it may be possible to proceed with only one of the

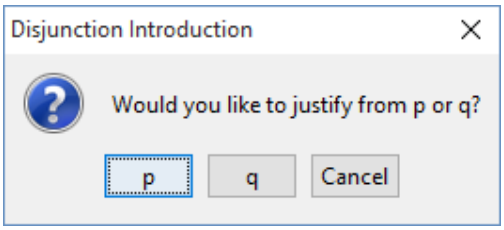

Figure 4: The disjunction introduction dialog.

conjuncts. The choice reminds the user to think ahead.

A classic example of a choice rule is $\vee$**I**, disjunction introduction. Here, a choice must be made as to which disjunct to prove from (at least in **NJ** - under **NK** we could also double negate). In NDP, the user must choose how to proceed. This is implemented through a dialog as shown in Figure 4. This dialog breaks the flow of the proof. Where users click through automatic rules with little thought, disjunction introduction requires more input. This explicit demonstration of the location of choice helps to show when it is required, and why.

An example of the third category of rules, those which require explicit further input from the user, is existential introduction $\exists$**I**. To justify $\exists xFx$ the user must choose a term to justify from (assuming the currently selected resource does not match the pattern Fx). The user is again presented with a dialog, but now must manually input a term to use, instead of simply choosing another button. Figure 5 shows the dialog presented in this case. The term input contains free syntax, and can be as long as required, allowing for the input of functional terms (e.g. $ffgfaffb$). Say the user inputs a, yielding Fa. NDP will then check to see if Fa already appears in scope. If it does, the current goal will be justified. Otherwise, Fa will be added as a new goal. A similar procedure applies for $\forall$**E**. In this case, however, the choice of term is fairly harmless - as with $\wedge$**E** we can always re-use a universal resource. The distinction between the choices of $\exists$**I** and $\forall$**E** become clear in NDP, since once $\exists$**I** is applied that line cannot be selected again, whereas the $\forall$ line can.

3.3 Level of Automation

A standard feature of much proof assistant software is automation of the proof process. NDP, by contrast, has been built with very little automation in mind. The automation implemented is at the level of rules, rather than proofs. That is, we have tried to automate the application of each rule as much as possible, without automating the proof process itself. The reason for this is that the focus is on replicating the pen-and-paper process.

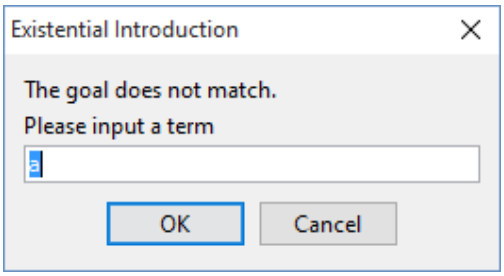

Figure 5: The existential introduction dialog. The user has inputted a.

By automating the writing of each proof line, users are able to move through deductions faster, focussing more on the strategy involved. Speed and the ability to easily "undo" mistakes also removes hurdles from the bulk trial and error method of learning strategy. A student unsure of the next step in a complicated pen and paper proof may be overly wary - a wrong move would result in writing out the whole proof again. In NDP, however, she can chose a rule in the knowledge that the current proof state can easily be retrieved. Similarly, students sometimes find rules like disjunction elimination, which requires creating four new lines and two scopes, to be intimidating and tiresome. Yet disjunction elimination is an automatic rule and should be applied as quickly as possible. In NDP disjunction elimination is achieved with two clicks, a less daunting task.

NDP applies rules, but the user must directly control it. Of course in some cases, the automation can circumvent attempts at learning. For example, the universal introduction rule always generates a new constant, that appears nowhere else in the proof. By doing so, the requirements for terms with that rule are always met. However, this takes some control away from the user. The $\forall$**I** rule does not always require a *completely* new term - if one appears out of scope it can be reused. NDP includes an option to disable automatic parameters. If a user chooses to do this, they will be asked for a term when performing $\forall$**I** and $\exists$**E**. If this term is illegal (violates the requirements for terms with those rules), the user is notified, and the offending justification marked.

3.4 Further Features

A *Rule Palette* allows individual rules to be (de)activated independently of the proof system chosen. The rule palette's layout shows the symmetry of Gentzen's rules, and gives some indication as to how the rules fit into different logical systems. By only activating certain rules, students can complete exercises in subsets of a system before being introduced to it fully, and see where certain rules are needed. For

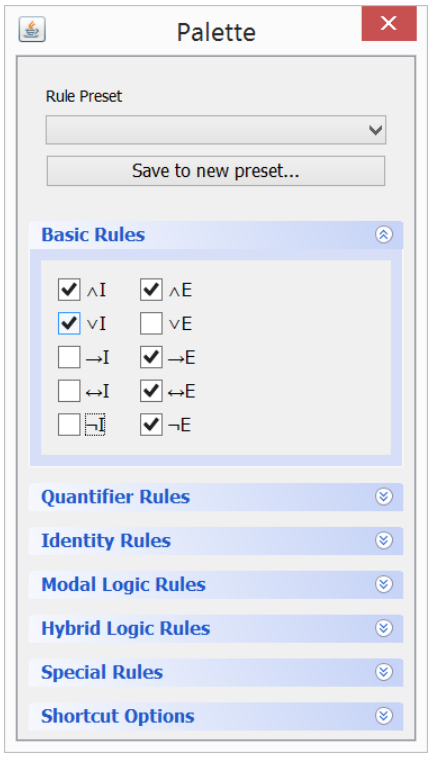

Figure 6: The Rule Palette. Only simple deduction rules are available; rules involving assumptions have been deactivated.

example, the rule palette can be used to demonstrate the importance of double negation elimination in **NK**, by attempting a proof of $(p \vee \neg p)$ without double negation elimination to see how far it goes. Once we get stuck, we turn on double negation elimination to finish the proof. Users can try out their own systems too, to see how different rules interact with each other.

Upon finishing a proof, it can be saved as either an editable proof or a demonstration proof. A demonstration proof has interaction disabled, providing a means to follow through an already complete proof. This is essentially the step by step deduction given above but in electronic form. Editable proofs behave similarly, but allow a user to take over the proof at any point. No work further than completing the deduction is required to generate these. Proofs can also be exported to unicode format and as an image. Complete proofs can also easily be animated in `.gif` format, for use in slides or online. A primary feature of NDP is its ability to export proofs to LaTeX code. This interacts with a LaTeX package (based on Ti*k*Z) which

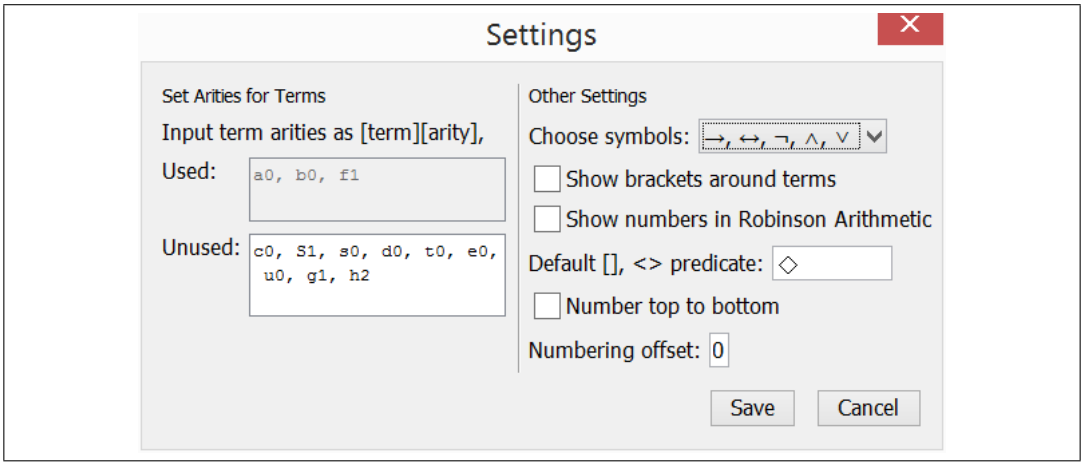

Figure 7: The settings dialog. The constants a and b have been used in the proof, as has the unary function f. It is still possible to change the arities of c, S, s, d, t, e, u, g and h. "Show numbers in Robinson Arithmetic" causes terms like $SSS0$ to be displayed as 3.

generates nicely typeset deductions. The task of producing exercises and their solutions involves little more than completing deductions on NDP - no fiddling about with alignment or trying to recall commands required.

In order to increase flexibility in using NDP, a number of settings are available. Two symbol sets ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ vs. $\sim, \&, \vee, \supset, \equiv$) can be chosen from. The line numbering can be tweaked in two ways. Standard top-to-bottom numbering is possible, and an offset can be applied, so that sub-proofs can be replaced without repeating a proof in full. Figure 7 shows the settings dialog.

Though originally intended to cover only propositional and predicate logic and Peano Arithmetic, we have begun extending NDP to cover other logics, and to consider new features. We've implemented a system of modal logic and hybrid logic using a labelled deduction method. These rules are available in the standard rule palette but are not thoroughly tested. A very rudimentary second order logic is also available, easily implemented due to Java seeing no distinction between predicate and variable symbols when making substitutions. In an attempt to automate the proof process, a *Magic Mode* is provided. This applies any rules which require no extra input for up to 10 iterations. In exceptional circumstances Magic Mode can complete proofs, but in general will only move forward one or two steps. Finally, a method to include custom axioms has been implemented.

3.5 Use in Teaching

We have used NDP as part of a course teaching natural deduction strategies. All the deduction exercises for the course were generated by the software, and it was also made available for students to download. Many students did so, and used NDP to complete exercises and study for tests. We released exercise solutions both as text documents and editable proof files. A novel use NDP was put to was in catching up on missed lectures. Since NDP applies each rule correctly, by studying what happens students could learn the rule themselves. While the motivation and strategy discussed in lectures was absent here, the correct manipulation of the formula was learned. NDP's automated rule application had some downsides though. Some students found overuse of NDP resulted in over reliance - you don't have to remember how to set out implication introduction if the software does it for you. Since tests were by pen and paper, this proved problematic. The best combination seemed to be use of pen and paper to practise rules, and NDP to practise strategy.

In the context of tutorials, NDP allowed for greater flexibility in presentation. Again due to "undo" it was easier to recover from bad choices, encouraging student participation. Also, a source of potentially confusing transcription errors - the tutor's handwriting - was removed. In one on one situations, NDP allowed for a greater flow of conversation. Discussed strategies for stuck proofs could be implemented quickly and results considered in much less time than would be required to write 10 lines of formulas by hand.

3.6 Implementation

NDP was implemented in Java using the `Swing` and `SwingX` graphical user interface libraries. The code was written to be extendable and with a goal of modularity, to allow different interfaces to interact with the same backend.

Formulas are held as strings in a TEX macro format, using prefix notation for easier argument parsing. This also simplifies the process of exporting proofs to LATEX code. For example, the formula $(Fa \wedge \forall x(Fx \rightarrow Gx))$ would be stored as `\con{Fa}{\qa{x}{\imp{Fx}{Gx}}}`. Each line of a proof is an `NDLine` object, which contains information such as the formula, the line number, the type of line and the justification. The `NDLine` class also has methods returning each argument of a formula.

The `ProofMethods` class forms the core of the program. This holds the current proof state as an array of `NDLine`s. The application of a rule results in a relevant modification of the proof array and any lines within it. For example, suppose $\wedge$**E** is applied, as possible in Figure 2. Line 3 is the resource, and `ProofMethods` first

obtains line 3's arguments (p and q) from its `NDLine`. Neither p nor q matches the current goal (line 8, $\perp$), and so the user will be asked whether they wish to extract both p and q, or just one. Suppose the user selects to only extract q (this seems a good move, since we'll be able to get a contradiction with $\neg q$). The current proof state will be extended with a new `NDLine` containing q, justified with 3, $\wedge\mathbf{E}$.

Rules themselves are methods within the `ProofMethods` class, and the system can be extended by adding new methods to give new rules. In practice, this means that additional rules (such as Disjunctive Syllogism) or extensions to the system can be added fairly easily. `ProofMethods` is designed to be as self-contained as possible; methods for printing the proof array to the command line mean it could be used without a graphical interface. In fact, this is how early development proceeded. Unfortunately, `ProofMethods` is not entirely standalone. Specifically, when interaction from the user is required (such as choosing a term for $\forall E$), `ProofMethods` must fall back on `Swing` libraries for graphical input.

On top of `ProofMethods` sits the `ProofPanel` class, a modified `JPanel` which provides user interaction with `ProofMethods`. `ProofPanel` interprets the proof array and arranges the deduction onscreen. The function of the rule palette is implemented entirely within the `ProofPanel`. If conjunction introduction ($\wedge\mathbf{I}$) is turned off then the option to apply that rule becomes unavailable on the `ProofPanel`. That is, even with $\wedge\mathbf{I}$ "disabled" `ProofMethods` is still able to apply that rule - there is just no way for the command to do so to reach it. The rule palette makes extensive use of the `SwingX` library.

A modified `JFrame` constitutes the main window of the Proof Assistant and controls tasks such as New Proof, Save, Open and Export. Proofs are saved in plain text files which contain complete undo histories and settings profiles. There is no difference between `.ndp` (editable) and `.ndu` (demonstration) files — they are read in differently but their contents are identical.

NDP is available on SourceForge at `http://sourceforge.net/p/proofassistant/`.

References

[1] D. Velleman, *How to Prove it: A Structured Approach.* Cambridge University, 1994.

[2] D. Velleman, *How to Prove It: A Structured Approach.* Cambridge University Press, 2006.

[3] A. Cupillari, *The Nuts and Bolts of Proofs.* Elsevier Academic Press, 2005.

[4] D. Solow, *How to read and do proofs: an introduction to mathematical thought processes.* Wiley, 2002.

[5] I. Copi, C. Cohen, and K. McMahon, *Introduction to Logic: Pearson New International Edition*. Pearson Education, Limited, 2013.

Received 11 October 2016