Programming Research Group

J'n'J in Jape
Encoding a Natural Deduction System for the Jape Proof Editor

Bernard Sufrin
Richard Bornat

PRG-TR-9-98

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Abstract

In this note we show how JAPE's presentations of proofs are related to the natural-deduction presentations used by Jim Woodcock and Jim Davies in [7] ("the book"). This is followed by description of JAPE implementation – J'N'J– of the logic described in the book. The implementation is designed to present a fairly natural direct-manipulation interface to its users, and we also explain the details of its realisation in the JAPE tactic language.

The note is designed as background reading for anyone wishing to know how to implement a logic with JAPE, or to conduct proofs in J'N'J. It is best read in conjunction with the early chapters of the book, but anyone who is used to the natural-deduction style of proof presentation will be on familiar territory.

In a companion paper [6] we present several case-studies in the use of the Unix/X implementation of JAPE to do proofs with J'N'J.

1 Introduction

JAPE is a computer program designed to provide support for teachers, students and practitioners of formal proof. It is *generic* – in the sense that it can be configured to support the discovery of formal proofs in almost any formal system which satisfies certain (minimal) requirements. When configured for use in a particular formal system it provides a sort of "pedantic pencil" which can be used to write proofs (and partial proofs) which satisfy the rules of the system, and which will not permit itself to be used to write erroneous pseudo-proofs.

In this note we show how JAPE's presentations of proofs are related to the natural-deduction presentations used by Jim Woodcock and Jim Davies in [7] ("the book"). This is followed by description of JAPE implementation – J'N'J– of the logic described in the book. The implementation is designed to present a fairly natural direct-manipulation interface to its users, and we also explain the details of its realisation in the JAPE tactic language.

The note is designed as background reading for anyone wishing to know how to implement a logic with JAPE, or to use JAPE to conduct proofs in J'N'J. It is best read in conjunction with the early chapters of the book, but anyone who is used to the natural-deduction style of proof presentation will be on familiar territory.

1.1 Proofs and Proof Rules

Conjectures[1] in the book are denoted in the form

$$\frac{assumption_1 \quad assumption_2 \quad \cdots}{conclusion}$$

and mean "From the assumptions[2] $assumption_1, assumption_2, \cdots$ we may infer *conclusion*."

For example, the book conjecture

$$\frac{p \wedge q}{q \wedge p}$$

can be read as: "From $p \wedge q$ we may infer $q \wedge p$."[3]

[1]In this note, and elsewhere, we tend to use the phrases "proof tasks", "goals", and "conjectures" more or less interchangeably.

[2]In this note, and elsewhere, we tend to use the words *assumption*, *premiss*, and *hypothesis* interchangeably.

[3]Sometimes we say "If $p \wedge q$ then $q \wedge p$", and sometimes "If we assume $p \wedge q$ then we can prove $q \wedge p$": all these sentences mean the same.

Proofs in the book are presented as trees. For example

$$\dfrac{\dfrac{p \wedge q}{q}\,[\wedge\text{-elim2}] \quad \dfrac{p \wedge q}{p}\,[\wedge\text{-elim1}]}{q \wedge p}\,[\wedge\text{-intro}]$$

The root of the tree is the conclusion, and the leaves of the tree (in this case identical) are the assumptions. Each node of the tree is labelled with the name of a rule which is used to justify its presence there.

Some proof rules can be presented as "glue" for inference trees. For example, the rule

$$\dfrac{p \quad q}{p \wedge q}\,[\wedge\text{-intro}]$$

means: "If we have a proof tree $\bigtriangledown_1$ whose root is p, and a proof tree $\bigtriangledown_2$ whose root is q, then we may construct a tree $\frac{\bigtriangledown_1 \; \bigtriangledown_2}{p \wedge q}$ whose root is $p \wedge q$ and which has subtrees $\bigtriangledown_1$ and $\bigtriangledown_2$. In our earlier example we glued the trees

$$\dfrac{p \wedge q}{q}\,[\wedge\text{-elim2}] \quad \text{and} \quad \dfrac{p \wedge q}{p}\,[\wedge\text{-elim1}]$$

together using this rule.

Some proof rules permit us to make intermediate assumptions, but constrain the region of the proof in which they can be used. For example, the rule $[\vee\text{-elim}]$

$$\dfrac{p \vee q \qquad \begin{matrix}\lceil p \rceil^{[i]} \\ \vdots \\ r\end{matrix} \qquad \begin{matrix}\lceil q \rceil^{[i]} \\ \vdots \\ r\end{matrix}}{r}\,[\vee\text{-elim}]^{[i]}$$

means: "if we can prove $p \vee q$ and we can prove r by assuming p and we can also prove r by assuming q, then we can prove r." Viewed as tree-surgery this is: "If we have a proof tree $\bigtriangledown_1$ with conclusion $p \vee q$, and a proof tree $\bigtriangledown_2$ with conclusion r and assumption p, and a proof tree $\bigtriangledown_3$ with conclusion r and assumption q, then we can make a new tree $\frac{\bigtriangledown_1 \; \bigtriangledown_2 \; \bigtriangledown_3}{r}$ with conclusion r." The assumption p is given the same label – an arbitrary number $[i]$ – as the root inference of the tree to indicate that it not useable *outside* the proof $\bigtriangledown_1$ of r; likewise q is not useable outside the proof $\bigtriangledown_2$ of r.[4]

The rule of implication introduction

$$\dfrac{\begin{matrix}\lceil p \rceil^{[i]} \\ \vdots \\ q\end{matrix}}{p \Rightarrow q}\,[\Rightarrow\text{-intro}]^{[i]}$$

means "if we can prove q by assuming p, then we can prove $p \Rightarrow q$."

[4]It's the specific *occurences* of p and q generated by a particular use of the rule that may not be used elsewhere. Think of the rule as generating a ticket to use p (respectively q) within $\bigtriangledown_1$ (respectively $\bigtriangledown_2$). Another rule may be used elsewhere to generate a different ticket to use p or q within a different subtree.

1.2 Sequent Notation

JAPE uses a different notation to denote conjectures. In the sequent notation, the book conjecture

$$\frac{assumption_1 \quad assumption_2 \quad \cdots}{conclusion}$$

is written as

$$assumption_1, assumption_2, \cdots \vdash conclusion$$

For example, the book conjecture

$$\frac{p \wedge q}{q \wedge p}$$

is written

$$p \wedge q \vdash q \wedge p$$

The sequent notation may also be used to express rules of inference – and is particularly convenient for rules involving subsidiary deductions from assumptions. For example, the rule [⇒-intro] is presented in the sequent notation as

$$\frac{p \vdash q}{\vdash p \Rightarrow q} \text{ [}\Rightarrow\text{-intro]}$$

and the rule [∨-elim] is

$$\frac{\vdash p \vee q \quad p \vdash r \quad q \vdash r}{\vdash r} \text{ [}\vee\text{-elim]}$$

The utility of the sequent notation will become apparent later in the document when we discuss JAPE's internal representation for proofs.

1.3 The Box-Style Proof Presentation

When we attempt to find a proof for a conjecture we must keep track of its premises, and of the scope of assumptions made during subproofs. This book-keeping can become burdensome and error prone during the discovery of a large proof. Fortunately JAPE not only does the book-keeping, but also offers a style of presentation – the box style – which makes it clear what assumptions are available for use at each stage in the discovery. In this style – which was developed from the style used in [4] – a new box is started whenever a new assumption is made: the assumption appears at the top of the box, and the conclusion to which it is directed appears at the bottom.

The examples which follow are chosen to illustrate the main features of this style – which can be viewed as a linearised form of the presentation style used in the book.

Illustration 1.3.1

When conducting the proof of

$$(p \wedge q \Rightarrow r) \Rightarrow (p \Rightarrow (q \Rightarrow r))$$

the first move we make is [⇒-intro]. The book[5] shows the resulting partial proof as

$$\dfrac{\begin{array}{c}\lceil\, p \Rightarrow (q \Rightarrow r)\rceil^{[1]} \\ \vdots \\ p \wedge q \Rightarrow r\end{array}}{(p \wedge q \Rightarrow r) \Rightarrow (p \Rightarrow (q \Rightarrow r))}\ [\Rightarrow\text{-intro}]^{[1]}$$

and JAPE shows it as

1 :	p∧q⟹r	assumption	
	. . .		
2 :	p⟹(q⟹r)		
3 :	(p∧q⟹r)⟹(p⟹(q⟹r))	⟹−intro	1−2

Apart from the presence of line numbers and boxes this is actually quite close to the book's presentation. The horizontal ellipsis between the assumption on line 1 and the conclusion on line 2 signifies the presence of a hole in the proof, just as the vertical ellipsis between assumption and conclusion in the book does. The box which surrounds the hypothetical proof plays the same role as the numeric label in the book presentation – it indicates that the scope of the assumption is the subproof within the box, and that it may not be used outside the box.

In the box style, the name of the rule used to justify the presence of a formula appears at the side of that formula, and is annotated with an indication of the position of the formulae or subproofs which satisfy the antecedents of the rule. In this case the subproof used to justify the use of [⇒-intro] at line 3 will be contained in the box which (at present) extends from line 1 to line 2.

The second [⇒-intro] leads to

$$\dfrac{\dfrac{\begin{array}{c}\lceil\, p \wedge q \Rightarrow r\rceil^{[1]} \\ \lceil p\rceil^{[2]} \\ \vdots \\ q \Rightarrow r\end{array}}{p \Rightarrow (q \Rightarrow r)}\ [\Rightarrow\text{-intro}]^{[2]}}{(p \wedge q \Rightarrow r) \Rightarrow (p \Rightarrow (q \Rightarrow r))}\ [\Rightarrow\text{-intro}]^{[1]}$$

which JAPE shows as

[5]Example 2.8

1 :	p∧q⟹r	assumption	
2 :	p	assumption	
	. . .		
3 :	q⟹r		
4 :	p⟹(q⟹r)	⟹−intro	2−3
5 :	(p∧q⟹r)⟹(p⟹(q⟹r))	⟹−intro	1−4

There are now two nested subproofs in play, and the line numbers have changed to reflect the new structure, as have the rule annotations.

The third move leads to

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\lceil p\land q\Rightarrow r\rceil^{[1]}\\ \lceil p\rceil^{[2]}\\ \lceil q\rceil^{[3]}\\ \vdots\\ r\end{array}}{q\Rightarrow r}\,[\Rightarrow\text{-intro}]^{[3]}}{p\Rightarrow(q\Rightarrow r)}\,[\Rightarrow\text{-intro}]^{[2]}}{(p\land q\Rightarrow r)\Rightarrow(p\Rightarrow(q\Rightarrow r))}\,[\Rightarrow\text{-intro}]^{[1]}$$

which JAPE shows as

1 :	p∧q⟹r	assumption	
2 :	p	assumption	
3 :	q	assumption	
	. . .		
4 :	r		
5 :	q⟹r	⟹−intro	3−4
6 :	p⟹(q⟹r)	⟹−intro	2−5
7 :	(p∧q⟹r)⟹(p⟹(q⟹r))	⟹−intro	1−6

Our goal is now to establish r, and the available assumptions are $p \land q \Rightarrow r$, p, and q. Following the book we now make an [⇒-elim] move, which yields

$$\dfrac{\dfrac{\dfrac{\lceil p\land q\Rightarrow r\rceil^{[1]} \qquad \begin{array}{c}\lceil p\land q\Rightarrow r\rceil^{[1]}\\ \lceil p\rceil^{[2]}\\ \lceil q\rceil^{[3]}\\ \vdots\\ p\land q\end{array}}{\dfrac{r}{q\Rightarrow r}\,[\Rightarrow\text{-intro}]^{[3]}}\,[\Rightarrow\text{-elim}]}{p\Rightarrow(q\Rightarrow r)}\,[\Rightarrow\text{-intro}]^{[2]}}{(p\land q\Rightarrow r)\Rightarrow(p\Rightarrow(q\Rightarrow r))}\,[\Rightarrow\text{-intro}]^{[1]}$$

which JAPE shows as

1 :	p∧q⟹r	assumption
2 :	p	assumption
3 :	q	assumption
	. . .	
4 :	p∧q	
5 :	r	⟹−elim 1,4
6 :	q⟹r	⟹−intro 3−5
7 :	p⟹(q⟹r)	⟹−intro 2−6
8 :	(p∧q⟹r)⟹(p⟹(q⟹r))	⟹−intro 1−7

The left subtree of the proof rooted at line 5 is represented by the hypothesis on line 1, and its right subtree by $p \wedge q$ on line 4. We therefore annotate the name of the rule with line numbers 1 and 4.

Our goal is now to prove p and q from the assumptions on lines 1, 2, and 3. Following the book (how else?) we make an [∧-intro], and this closes the proof.

1 :	p∧q⟹r	assumption
2 :	p	assumption
3 :	q	assumption
4 :	p∧q	∧−intro 2,3
5 :	r	⟹−elim 1,4
6 :	q⟹r	⟹−intro 3−5
7 :	p⟹(q⟹r)	⟹−intro 2−6
8 :	(p∧q⟹r)⟹(p⟹(q⟹r))	⟹−intro 1−7

At all stages of this proof, the tree structure which is explicit in the book presentation is here implicit in the line numbers used to annotate the names of proof rules. It is very easy to go about reconstructing a proof tree from this kind of presentation, and we invite our readers to do so for this proof without reference to the book.

Illustration 1.3.2

The following sequence of partial proofs is taken from a proof of the commutativity of ∨. It shows that when a rule has multiple antecedents, then the co-lateral proofs which are used to justify those antecedents may appear within separate co-lateral boxes.

We first see the partial proof before any moves have been made.

1 :	p∨q	assumption
	. . .	
2 :	q∨p	

The first prove move is [∨-elim], The book would show the resulting partial proof as

$$\dfrac{p \vee q \qquad \begin{matrix}\lceil p \rceil^{[1]} \\ \vdots \\ q \vee p\end{matrix} \qquad \begin{matrix}\lceil q \rceil^{[1]} \\ \vdots \\ q \vee p\end{matrix}}{q \vee p}\ [\vee\text{-intro}]^{[1]}$$

JAPE shows it as

1 :	p∨q	assumption	
2 :	p	assumption	
	. . .		
3 :	q∨p		
4 :	q	assumption	
	. . .		
5 :	q∨p		
6 :	q∨p	∨−elim	1,2−3,4−5

This rule has three antecedents.[6] The first antecedent, $(p \vee q)$, corresponds to the original premiss of the proof which is the assumption on line 1. The remaining antecedents are represented by the co-laterally boxed subproofs from lines 2 to 3 and lines 4 to 5 respectively. Neither of those subproofs is complete – a fact which the ellipsis within them shows. The additional assumption p available in subproof 2-3, is not available in subproof 4-5, and the additional assumption q available in subproof 4-5 is not available in subproof 2-3.

The next move we make[7] is [∨-intro2] – it is made in subproof 2-3, and leads to

$$\dfrac{p \vee q \qquad \dfrac{\lceil p \rceil^{[1]}}{q \vee p}\ [\vee\text{-elim}] \qquad \begin{matrix}\lceil q \rceil^{[1]} \\ \vdots \\ q \vee p\end{matrix}}{q \vee p}\ [\vee\text{-intro}]^{[1]}$$

which JAPE shows as

[6] The rule is on page 3.

[7] The details of the interaction are explained in the companion paper.

1 :	p∨q	assumption	
2 :	p	assumption	
3 :	q∨p	∨−intro2	2
4 :	q	assumption	
	. . .		
5 :	q∨p		
6 :	q∨p	∨−elim	1,2−3,4−5

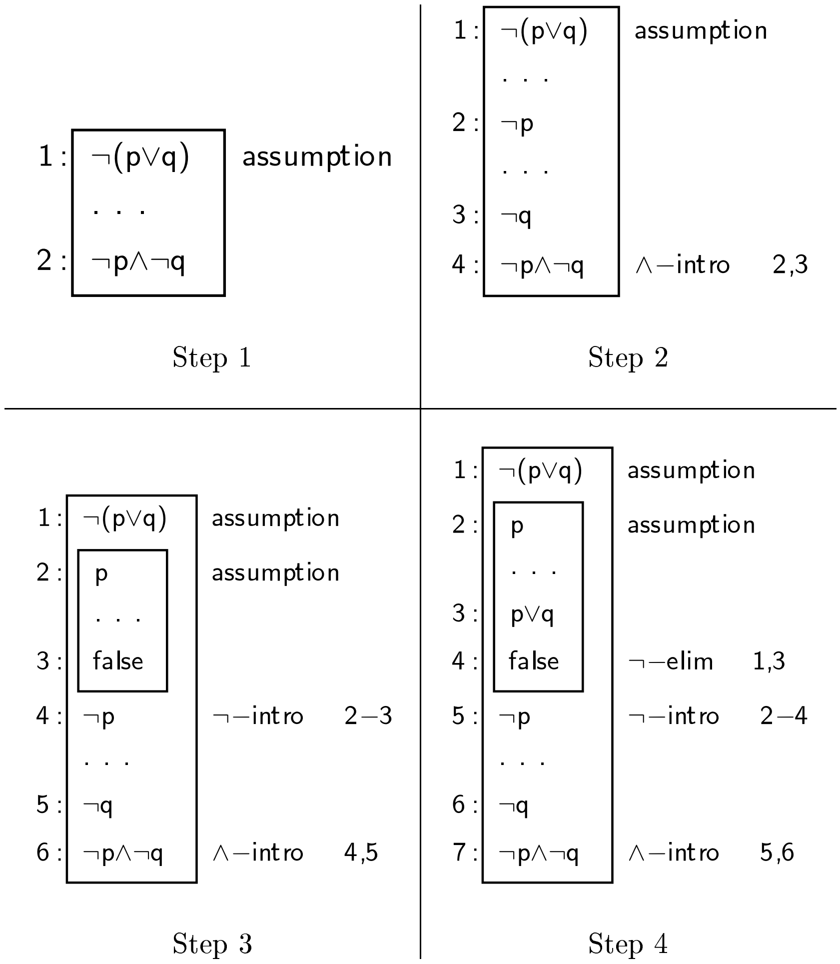
The proof is closed by an [∨-intro1] in subproof 4-5.

1 :	p∨q	assumption	
2 :	p	assumption	
3 :	q∨p	∨−intro2	2
4 :	q	assumption	
5 :	q∨p	∨−intro1	4
6 :	q∨p	∨−elim	1,2−3,4−5

Illustration 1.3.3

Our final sequence of partial proofs arose during the discovery of a proof of $\neg(p \vee q) \vdash \neg p \wedge \neg q$. We present them without comment, for comparison with the corresponding book presentation.[8]

[8]Example 2.10.



2 JAPE's Internal Representation: Sequent Trees

JAPE keeps track of the premisses and assumptions available at each node of the proof tree it is constructing. Its internal representation for a proof is effectively a tree of sequents.

For example, a short way into the proof (Example 2.10) of $deMorgan_1$

$$\frac{\neg(p \vee q)}{\neg p \wedge \neg q}$$

we see the following partial proof tree, in which the premiss is left implicit.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\lceil p \rceil^{[1]}\\ \vdots \\ p \vee q\end{matrix} \quad \neg(p \vee q)}{\mathit{false}}\,[\neg\text{-elim}]}{\neg p}\,[\neg\text{-intro}]^{[1]} \qquad \dfrac{\vdots}{\neg q}}{\neg p \wedge \neg q}\,[\wedge\text{-intro}]$$

In the sequent notation this is

$$\dfrac{\dfrac{\dfrac{\neg(p \vee q), p \vdash p \vee q \quad \neg(p \vee q), p \vdash \neg(p \vee q)}{\neg(p \vee q), p \vdash \mathit{false}}\,[\neg\text{-elim}]}{\neg(p \vee q) \vdash \neg p}\,[\neg\text{-intro}] \qquad \neg(p \vee q) \vdash \neg q}{\neg(p \vee q) \vdash \neg p \wedge \neg q}\,[\wedge\text{-intro}]$$

Notice that the premiss $\neg(p \vee q)$, which is left implicit in the book presentation, is represented explicitly in all the sequents. The assumption p, generated by the [$\neg$-intro] move becomes part of each sequent above that point in the tree.

If we ignore the material at the left of the turnstiles, then this is the same as the book's presentation. In general as we ascend the tree new assumptions are incorporated into the assumption lists of the sequents. In the book presentation – these are placed above the tree proper, and separated from it by vertical ellipsis.

Another way of thinking of a sequent is as a statement of a *goal* or *proof task*. We can think of each application of a rule in a proof as transforming one of the tasks at the fringe of the proof tree into the list of subtasks which must be completed before the task itself is considered complete. A partial proof tree is both a record of the moves made so far and an indication of what subtasks remain. A proof is incomplete when there are unsatisfied goals at the fringe of the partial proof tree, and it is complete when there are none.

For example, the proof tree shown above demonstrates that the [$\wedge$-intro] move at the root of the tree transformed the goal $\neg(p \vee q) \vdash \neg p \wedge \neg q)$ into the two subgoals $\neg(p \vee q) \vdash \neg p$ and $\neg(p \vee q) \vdash \neg q$, and the [$\neg$-intro] move transformed the goal $\neg(p \vee q) \vdash \neg p$ into the goal $\neg(p \vee q), p \vdash \mathit{false}$.

The top left branch can now be closed by [$\vee$-intro1]. This rule transforms the goal $\neg(p \vee q), p \vdash p \vee q$ into an *empty* list of subgoals.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(p \vee q), p \vdash p \vee q}\,[\vee\text{-intro1}] \quad \neg(p \vee q), p \vdash \neg(p \vee q)}{\neg(p \vee q), p \vdash \mathit{false}}\,[\neg\text{-elim}]}{\neg(p \vee q) \vdash \neg p}\,[\neg\text{-intro}] \qquad \neg(p \vee q) \vdash \neg q}{\neg(p \vee q) \vdash \neg p \wedge \neg q}\,[\wedge\text{-intro}]$$

Now consider the branch to its right. This ends with the task

$$\neg(p \vee q), p \vdash \neg(p \vee q)$$

In the book such a task is presented as trivially achievable – requiring no evidence at all. But in the sequent style it needs formal justification by a rule which captures the idea – implicit in the book – that if one has assumed p then one may infer p.[9] This rule is

$$\frac{}{p \vdash p}\ \textit{hypothesis}$$

and its application to the sequent in question closes the $\neg p$ branch of the proof.

JAPE can present proofs as trees of sequents: in this style the complete proof would be displayed as

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(p\vee q),\, p \vdash \neg(p\vee q)}\,\text{hypothesis} \qquad \dfrac{\dfrac{}{\neg(p\vee q),\, p \vdash p}\,\text{hypothesis}}{\neg(p\vee q),\, p \vdash p\vee q}\,\vee\text{-intro1}}{\neg(p\vee q),\, p \vdash \text{false}}\,\neg\text{-elim}}{\neg(p\vee q) \vdash \neg p}\,\neg\text{-intro} \qquad \dfrac{\dfrac{\dfrac{}{\neg(p\vee q),\, q \vdash \neg(p\vee q)}\,\text{hypothesis} \qquad \dfrac{\dfrac{}{\neg(p\vee q),\, q \vdash q}\,\text{hypothesis}}{\neg(p\vee q),\, q \vdash p\vee q}\,\vee\text{-intro2}}{\neg(p\vee q),\, q \vdash \text{false}}\,\neg\text{-elim}}{\neg(p\vee q) \vdash \neg q}\,\neg\text{-intro}}{\neg(p\vee q) \vdash \neg p\wedge\neg q}\,\wedge\text{-intro}$$

The size of this presentation should give the reader some insight into why we also provide the box-style presentation.

1 :	$\neg(p\vee q)$	assumption
2 :	p	assumption
3 :	$p\vee q$	$\vee$−intro1 2
4 :	false	$\neg$−elim 1,3
5 :	$\neg p$	$\neg$−intro 2−4
6 :	q	assumption
7 :	$p\vee q$	$\vee$−intro2 6
8 :	false	$\neg$−elim 1,7
9 :	$\neg q$	$\neg$−intro 6−8
10 :	$\neg p\wedge\neg q$	$\wedge$−intro 5,9

[9] There are systems of formal reasoning in which assumptions are not treated quite so straightforwardly – which is why JAPE has to use this rule explicitly.

3 The Jape Description of J'n'J

In this section we present the complete Jape source for an implementation of the J'n'J logic. We give brief explanations of the Jape metalanguage and tactic language constructs which we use, but for a complete description of these languages the reader should consult [2].

3.1 Syntax

A Jape theory starts with a declaration of the structure of sequents. If it uses a nonstandard font encoding,[10] then that must be declared first. The `INITIALISE ...` declaration allows rules to be stated in a concise form: it is explained in section 3.4.

```
FONTS   "Laura"
SEQUENT IS BAG ⊢ FORMULA
INITIALISE autoAdditiveLeft true
```

Next we introduce the syntax of propositions. This is perfectly straightforward: the declarations of infix operators specify their syntactic binding power and left/right associativity, and the declarations of prefix operators specifies their binding power. The binding power of juxtaposition (interpreted here as function application) must also be specified.

```
PREFIX   350 ¬
INFIX    50R  ⇔
INFIX    100R ⇒
INFIX    200L ∨
INFIX    300L ∧
JUXTFIX  40000
```

Next we specify that – in the declaration of a rule – identifiers starting with $p, q, r,$ or s stand for any (propositional) formula, whereas identifiers starting with $x, y,$ or z stand for variables.[11] We also introduce two constants, `false` and `true` – which always stand for themselves.

[10] Non-standard font encodings make it possible for a full range of symbols to be used in the theory, rather than restricting us to the rather ugly possibilites offered by the standard Latin encodings.

[11] We won't meet variables until we discuss the predicate calculus.

```
CLASS FORMULA  p q r s
CLASS VARIABLE x y z
CONSTANT false true
```

The notation used for substitutions may be specified once we have declared at a formula class and a variable class. In this case we follow the book's syntax exactly.

```
SUBSTFIX 50000 [ p / x ]
```

3.2 Propositional Logic Rules

The propositional rules can be stated very simply[12]

```
RULE  "∧-intro"  FROM p AND q INFER p∧q
RULE  "∧-elim1"  FROM p∧q INFER p
RULE  "∧-elim2"  FROM p∧q INFER q

RULE  "∨-intro1" FROM p INFER p∨q
RULE  "∨-intro2" FROM q INFER p∨q
RULE  "∨-elim"   FROM p∨q AND p⊢r AND q⊢r INFER r

RULE  "⇒-intro"  FROM p ⊢ q INFER p⇒q
RULE  "⇒-elim"   FROM p⇒q AND p INFER q

RULE  "¬-intro"  FROM p ⊢ false INFER ¬ p
RULE  "¬-elim"   FROM ¬p AND p INFER false

RULE  "false-intro" FROM ¬p AND p INFER false
RULE  "false-elim"  FROM ¬p ⊢ false INFER  p
RULE  "true-intro"  INFER  ⊢ true

RULE  "⇔-intro"  FROM p ⇒ q AND   q ⇒ p INFER p ⇔ q
RULE  "⇔-elim1"  FROM p ⇔ q INFER p ⇒ q
RULE  "⇔-elim2"  FROM p ⇔ q INFER q ⇒ p
```

[12]In the expectation that the book will eventually choose names a bit more systematically, the rule which introduces `false` – known in the first edition of the book as "¬-elim" – also appears here with name "false-intro".

3.3 Structural Rules

Some rules, such as the *hypothesis* rule we discussed earlier, are left implicit in the book's treatment of proof construction. They capture some obvious ideas about the book-style proof trees which are not true in all formal systems, and that is why JAPE needs them to be stated explicitly.[13]

```
RULE hypothesis(p) INFER p ⊢ p
RULE cut(p)        FROM  ⊢ p AND   p ⊢ q  INFER   ⊢ q
RULE thin(p)       FROM  ⊢ q             INFER p ⊢ q
RULE dup(p)        FROM  p, p ⊢ q        INFER p ⊢ q
```

As we discussed earlier, the *hypothesis* rule captures the idea – implicit in the book – that if one has assumed p, then one may infer p.

The *cut* rule would be written as follows in the book

$$\dfrac{p \qquad \begin{array}{c}\lceil p \rceil \\ \vdots \\ q\end{array}}{q}\ [\text{cut}]$$

It captures the idea that a proof of p can be joined to a proof of q from p to produce a proof of q. Another convenient way of thinking about it is that if you need to prove q, and you can't see quite how to do it directly, then prove it from an additional hypothesis, p, and show that p holds.

The *thin* rule captures the idea that if one can prove q without p then one can prove it by assuming p. The *dup* rule captures the idea that if one can prove q by assuming p once, then one can also prove it by assuming p more than once.

JAPE can use the fact that rules of the form of *cut* and *hyp* are present in the logic to make its box style presentation a little prettier. It can also use the fact that a rules in the form of *thin* is present to permit the use of proven conjectures as if they were derived proof rules. We therefore declare

```
IDENTITY RULE hypothesis
CUT      RULE cut
WEAKEN   RULE thin
```

[13]The rules may be invoked from Jape tactics so they have each been given an explicit formal parameter `(p)` – a detailed explanation of why this is necessary appears later.

3.4 JAPE Inference Rules Reconsidered

When the declaration `INITIALISE autoAdditiveLeft true` has been made, as it has in our presentation of this logic, JAPE's notation for inference rules leaves implicit the role and fate of hypotheses other than those mentioned explicitly in the rule. In general any hypotheses which are present in a goal but not mentioned in the consequent of a rule become part of any antecedents which that rule generates.

For example, the rule

```
RULE  "∧-intro" FROM p AND q INFER p∧q
```

really means "To prove $p \wedge q$ from a bag, Γ, of assumptions, prove p from Γ and prove q from Γ." – a rule which logicians would write in something like the following form

$$\frac{\Gamma \vdash p \quad \Gamma \vdash q}{\Gamma \vdash p \wedge q}\ [\wedge\text{-intro}]$$

The rule

```
FROM p∨q AND p⊢r AND q⊢r INFER r
```

really means "To prove r from a bag, Γ, of assumptions, prove $p \vee q$ from Γ, prove r from Γ augmented with p, and prove r from Γ augmented with q." – a rule which logicians would write in the form

$$\frac{\Gamma \vdash p \vee q \quad \Gamma, p \vdash r \quad \Gamma, q \vdash r}{\Gamma \vdash r}\ [\vee\text{-elim}]$$

3.5 Predicate Logic Rules

JAPE is fairly flexible about the syntax of mixfix operators, so we will be able to give a completely straightforward definition of both syntactic forms in which each quantifier appears

$$\begin{array}{ll} \exists\, x : a | p \bullet q & \forall\, x : a | p \bullet q \\ \exists\, x : a \bullet q & \forall\, x : a \bullet q \end{array}$$

The additional syntax declarations which are necessary to support reasoning in predicate logic are

```
INFIX   800R  ∈
LEFTFIX 10  ∀ : •
LEFTFIX 10  ∀ : | •
LEFTFIX 10  ∃ : •
LEFTFIX 10  ∃ : | •
```

We also state that variables beginning with a, b, t or u stand for formulae. Our convention will be that such variables will denote set-values and terms – but at present JAPE cannot police this convention for us, since it doesn't discriminate between classes of formula.

```
CLASS FORMULA a b t u
```

We also need to inform JAPE that ∀ and ∃ are quantifiers – in other words that they bind variables.

```
BIND x SCOPE p   IN ∀ x:a • p
BIND x SCOPE p   IN ∃ x:a • p
BIND x SCOPE p q IN ∀ x:a | p • q
BIND x SCOPE p q IN ∃ x:a | p • q
```

The [∀-intro] rule captures the idea that if we want to prove that q holds for all elements x of a, then all we need to do is to find a fresh name y for an arbitrary element of a, and show that q holds for y. The name y must be chosen so that it is fresh – not used anywhere in the hypotheses. The rule is parameterised by the name chosen, so that it can be supplied at the point of invocation of the rule. The parameter declaration is decorated with the word `OBJECT` so as to guide JAPE in its choice of name if one isn't supplied. The rule given in the book is a special case of this rule, where the name chosen is x.

```
RULE    "∀-intro"(OBJECT y) WHERE FRESH y
FROM    y∈a ⊢ q[y/x]
INFER   ∀ x:a • q
```

The [∀-elim] rule is straightforward: if q holds for all elements x of a, then it holds for a particular term t in a.

```
RULE    "∀-elim"(t)
FROM    ∀ x:a • q
AND     t∈a
INFER   q[t/x]
```

If q holds for a term t which is in a, then there is an x in a for which t holds.

```
RULE    "∃-intro"(t)
FROM    t∈a
AND     q[t/x]
INFER   ∃ x:a • q
```

Existential elimination is a generalisation of or-elimination. Its informal justification is that if we know that there's an element x of a for which p holds, and we can prove r by assuming that p holds for some arbitrary element y of a (about which no further assumptions are made), then we know r. The provisos are to guarantee the arbitrariness of the element in question: `FRESH y` indicates that y must not appear in any of the hypotheses, and the provisos `y NOTIN r` and `y NOTIN ∃ x:a • p` indicate that y must not appear free in r or in the first antecedent.

```
RULE    "∃-elim"(OBJECT y)
WHERE   FRESH y
AND     y NOTIN ∃ x:a • p
AND     y NOTIN r
FROM    ∃ x:a • p
AND     y∈a, p[y/x] ⊢ r
INFER   r
```

Although the book gives direct introduction and elimination rules for this form of quantifier, we prefer to start by treating them as abbreviations defined by logical equivalence – then *deriving* the rules in the book as theorems.

```
RULE "∀|•" IS (∀ x:a|p•q) ⇔ (∀ x:a • p ⇒ q)
RULE "∃|•" IS (∃ x:a|p•q) ⇔ (∃ x:a • p ∧ q)
```

4 The User Interface

THE material we have given so far describes the J'N'J logic, and is – strictly speaking – sufficient to enable a user to start doing proofs. But a user who did so would quickly begin to be irritated by the mechanical details of communicating proof moves to JAPE. In this section we explain the implementation of a user interface for J'N'J which is designed to facilitate communication with JAPE and thereby help the user focus on problems of proof rather than communication.

4.1 Higher Order Tactics

Hand-application of the J'N'J proof rules can be a bit tedious – largely because many of them are biased towards forward proof, whereas JAPE *actually* conducts proofs backwards. So we need to do some work to sustain the illusion that the proof is being conducted forwards. Here we define a few higher order tactics (*tacticals*) to help administer the application of the basic rules.

4.1.1 Applying Elimination Rules

When an elimination rule is applied, we need to know what formula it is being applied to. Since it's rarely very much use to apply an elimination rule backwards, we arrange for the user to indicate the formula by selecting an assumption. For the moment we'll defer thinking about what should happen if no assumption is selected, save to say that in principle we'd like to permit the rule to be applied backwards if an intrepid user so wishes, but that novice users should be advised to select an assumption before using the rule. We therefore define

```
TACTIC ElimRule(rule)
(WHEN
  (LETHYP _p
      (ALT (SEQ      (WITHARGSEL rule) (WITHHYPSEL hypothesis))
           (SEQ  cut (WITHARGSEL rule) (WITHHYPSEL hypothesis))
           (FAIL ("%s is not applicable to formula %s", rule, _p))))
  (LETCONC _p
      (ElimRuleBackward rule))
  (FAIL ("Select an assumption-like formula before applying %s forwards", rule)))
```

This tactic is parameterised by the rule to be used, and its body consists of a single `WHEN` form. Such forms take a sequence of guarded tactics, followed by a default tactic. If one of the guards is satisfied, then its body is executed; if none of the guards is satisfied, then the default tactic is executed.

In this case if a hypothesis is selected (the `LETHYP _p` guard), then we execute the tactic

```
(ALT (SEQ      (WITHARGSEL rule)  (WITHHYPSEL hypothesis))
     (SEQ  cut (WITHARGSEL rule)  (WITHHYPSEL hypothesis)))
```

This tactic tries to apply the rule followed by the `hypothesis` rule; if that sequence doesn't succeed, then the tactic repeats it after first using `cut`. In both cases *rule* is applied using `WITHARGSEL`. This is an inbuilt tactic form which applies *rule* if there is no text selection, and otherwise provides the selected text as an actual parameter to *rule*. The `hypothesis` rule is applied using `WITHYPSEL` – this forces the *selected* hypothesis to be used.

Q Why would one want to pass a text selection as a parameter to a rule?

A When using a rule such as [$\forall$-intro], which requires a variable to be chosen to stand for the arbitrary element of a, the user can choose the variable by text-selecting it within the existing proof (if it appears). If there's no text selection, then JAPE makes up a suitable name on its own.

Q Why try both *rule*, and *cut* followed by *rule*?

A Sometimes the *rule* is enough to close part of a proof on its own. For example, with $x \in \Psi \wedge x \leq \varpi$ as the selected hypothesis, the sequence [$\wedge$-elim2] followed by *hypothesis* will close

$$x \in \Psi \wedge x \leq \varpi \vdash x \leq \varpi$$

It works as follows: [$\wedge$-elim2] transforms the goal to one in which the left conjunct is a proof unknown[14] ($\_p$)

$$\frac{x \in \Psi \wedge x \leq \varpi \vdash \_p \wedge x \leq \varpi}{x \in \Psi \wedge x \leq \varpi \vdash x \leq \varpi}\ [\wedge\text{-elim2}]$$

then [hypothesis] unifies the unknown with $x \in \Psi$, yielding

$$\frac{\overline{x \in \Psi \wedge x \leq \varpi \vdash x \in \Psi \wedge x \leq \varpi}\ [\text{hypothesis}]}{x \in \Psi \wedge x \leq \varpi \vdash x \leq \varpi}\ [\wedge\text{-elim2}]$$

But sometimes the user is just trying to make progress towards a goal, or wishes to establish a fact for later use in several branches of a proof.

For example, the sequence *cut*, [$\wedge$-elim2], *hypothesis* transforms the goal

$$x \in \Psi \wedge x \leq \varpi \vdash x \in ok$$

in three steps. The first yields a proof with a single unknown, ($\_p$)

$$\frac{x \in \Psi \wedge x \leq \varpi \vdash \_p \quad x \in \Psi \wedge x \leq \varpi, \_p \vdash x \in ok}{x \in \Psi \wedge x \leq \varpi \vdash x \in ok}\ [\text{cut}]$$

[14] Variables which are prefixed with an underscore are *proof unknowns* – they stand for object language formulae which are as-yet unknown. They are introduced into proofs by the application of rules whose antecedents contain variables which aren't contained in their consequents. They give users the chance to defer making a decision about the shape of a formula until it becomes obvious what it must be.

The second attacks the leftmost subgoal generated by the first, and yields a proof tree in which there are *two* proof unknowns, $(\_p, \_p_1)$

$$\dfrac{\dfrac{x \in \Psi \wedge x \leq \varpi \vdash \_p_1 \wedge \_p}{x \in \Psi \wedge x \leq \varpi \vdash \_p}\,[\wedge\text{-elim2}] \qquad x \in \Psi \wedge x \leq \varpi, \_p \vdash x \in ok}{x \in \Psi \wedge x \leq \varpi \vdash x \in ok}\,[\text{cut}]$$

The use of [hypothesis] unifies $\_p1 \wedge \_p$ with $x \in \Psi \wedge x \leq \varpi$ and yields the tree

$$\dfrac{\dfrac{\dfrac{}{x \in \Psi \wedge x \leq \varpi \vdash x \in \Psi \wedge x \leq \varpi}\,[\text{hypothesis}]}{x \in \Psi \wedge x \leq \varpi \vdash x \leq \varpi}\,[\wedge\text{-elim2}] \qquad x \in \Psi \wedge x \leq \varpi, x \leq \varpi \vdash x \in ok}{x \in \Psi \wedge x \leq \varpi \vdash x \in ok}\,[\text{cut}]$$

In the box style presentation the use of *cut* is treated specially so as to give the illusion that we have taken a forward proof step.[15]

1:	x∈Ψ∧x∈ϖ	assumption
2:	x∈ϖ	∧−elim2 1
	. . .	
3:	x∈ok	

4.1.2 Applying Introduction Rules

Propositional introduction rules are easy to apply backwards because they have the *subformula property* – every variable appearing in an antecedent appears in the consequent. On the other hand the only sensible way of applying an introduction rule *forward* is to specify the formula to be introduced by text-selecting it.

So if there is a text selection (the `LETARGSEL` guard), then the proof is cut with the selected formula, then *rule* is applied backwards. Otherwise we ignore any selected hypotheses and just apply the rule backwards. If that doesn't work we explain why.

```
TACTIC IntroRule(rule)
(WHEN  (LETARGSEL _p (SEQ (cut _p) rule))
       (ALT (LETHYP    _p rule)
            rule
            (MESSAGE (rule is not applicable))))
```

[15]Notice that, although $x \leq \varpi$ is recorded as an assumption in the remaining open sequent, the [$\wedge$-elim] annotation in this presentation perhaps gives a different impression. The only way to be sure that a formula is an hypothesis which is in the scope of a particular conclusion is to select that conclusion (by clicking on it) whereupon all formulae which are not in scope become grey. This is illustrated in [6].

When invoking a quantifier introduction rule we want the user to be able to text-select the bound variable ([∀-intro]) or the existential term ([∃-intro]) if it's obvious what it should be. The `WITHARGSEL` tactic form applies *rule* using the text selection (if there is one) as the actual parameter of the rule; otherwise it just applies the rule with no parameters, and an appropriate proof unknown or variable is supplied by default.

```
TACTIC QuantIntroRule(rule)
(ALT (WITHARGSEL rule)
     (FAIL ("%s is not applicable", rule)))
```

4.1.3 Defining the Menus

Next we define the menus which will appear at the top of the session window – the window where proof activity takes place. Although we will later specify a direct-manipulation interface which invokes rules in response to mouse-hits, this will not make the menu-style interface completely redundant.

```
MENU    "Propositional Rules"
    ENTRY   "hypothesis"
    SEPARATOR
    ENTRY    "false-elim"   IS (IntroRule "false-elim")
    SEPARATOR
    ENTRY    "∧-intro"      IS (IntroRule "∧-intro")
    TACTIC   "∧-elim-1"     IS (ElimRule  "∧-elim1")
    ENTRY    "∧-elim-2"     IS (ElimRule  "∧-elim2")
    SEPARATOR
    ENTRY    "∨-intro1"     IS (IntroRule "∨-intro1")
    ENTRY    "∨-intro2"     IS (IntroRule "∨-intro2")
    ENTRY    "∨-elim"       IS (ElimRule "∨-elim")
    SEPARATOR
    ENTRY    "⇒-intro"      IS (IntroRule "⇒-intro")
    ENTRY    "⇒-elim"       IS (ElimRule  "⇒-elim")
    SEPARATOR
    ENTRY    "¬-intro"      IS (IntroRule "¬-intro")
    ENTRY    "¬-elim"       IS (ElimRule "¬-elim")
    ENTRY    "false-intro"  IS (ElimRule "false-intro")
    ENTRY    "true-intro"   IS (IntroRule "true-intro")
    SEPARATOR
    ENTRY    "⇔-intro"      IS (IntroRule "⇔-intro")
    ENTRY    "⇔-elim1"      IS (ElimRule "⇔-elim1")
    ENTRY    "⇔-elim2"      IS (ElimRule "⇔-elim2")
END
```

It may seem curious at first that the rule named false-introduction is treated as an elimination rule: this is easily explained when one notes that it doesn't satisfy the subformula property, so it makes sense to select the (negative) formula which is to be contradicted, and the elimination rule tactic form is set up to do just that. The false-elimination rule is invoked by the introduction-rule machinery in order to give the user a chance to text-select a formula *sel* whose negation is a contradiction and from which the current goal follows. This is rarely very useful, but when it happens the resulting proof tree is:

$$\dfrac{\dfrac{\vdots}{\Gamma, \neg sel \vdash false}\text{[false-elim]} \quad \dfrac{\vdots}{\Gamma, sel \vdash goal}}{\Gamma \vdash goal}\text{[cut]}$$

For the moment we will place only buttons for the simplest forms of quantifier on the menu. Only when we come to construct the machinery for rewriting formulae using equivalences will we be able to add buttons which we can usefully use to invoke the "abbreviation" rules $\exists|\bullet$ and $\forall|\bullet$.

```
MENU "Quantifier Rules" IS
    ENTRY "∀-intro"   IS QuantIntroRule "∀-intro"
    ENTRY "∀-elim"    IS ElimRule       "∀-elim"
    ENTRY "∃-intro"   IS QuantIntroRule "∃-intro"
    ENTRY "∃-elim"    IS ElimRule       "∃-elim"
END
```

4.1.4 Automatic Invocation of *hypothesis*

At the end of each user-invoked inference we'd like the system to try to close any trivially-closeable proof branches which that move generated. In J'N'J the trivially closeable branches are those which are closeable by an invocation of *hypothesis* which doesn't resolve any proof unknowns. Such an invocation is called a *matching* invocation.[16] We therefore declare

```
    AUTOMATCH hypothesis
```

4.1.5 Direct Manipulation

A double-click on a formula is called a *hit*, and JAPE's direct manipulation infrastructure is designed to support "proof by hitting" – a style of proof which is mostly conducted by

[16]Its antithesis is a *unifying* invocation – one which does resolve proof unknowns.

making hits on formulae. In this section we specify the tactics to be invoked in response to hits.

First, for each form of hypothesis we specify what tactics to execute in response to a hit. Apart from the treatment of conjunction (and the implicit conjunction ⇔) this is completely straightforward.

```
HYPHIT p∨q ⊢ r IS  (ALT (ElimRule "∨-elim")  "∨-elim")
HYPHIT p⇒q ⊢ r IS  (ALT (ElimRule "⇒-elim")  "⇒-elim")
HYPHIT ¬p  ⊢ r IS  (ALT (ElimRule "¬-elim")  "¬-elim")
HYPHIT ∀x:a•q   ⊢ r IS ElimRule "∀-elim"
HYPHIT ∃x:a•q   ⊢ r IS ElimRule "∃-elim"
```

In response to hitting a conjunctive assumption, we first try to match one of the conjuncts with an outstanding conclusion. If neither succeeds, then we simply add both conjuncts to the list of assumptions.

```
HYPHIT p∧q ⊢ r IS
(ALT (SEQ  "∧-elim1"  (hypothesis(p∧q)))
     (SEQ  "∧-elim2"  (hypothesis(p∧q)))
     (SEQ
           cut "∧-elim1" (hypothesis (p∧q))
           cut "∧-elim2" (hypothesis (p∧q))
     ))

HYPHIT p⇔q ⊢ r IS
(ALT (SEQ  "⇔-elim1"  (hypothesis(p⇔q)))
     (SEQ  "⇔-elim2"  (hypothesis(p⇔q)))
     (SEQ
           cut "⇔-elim1" (hypothesis (p⇔q))
           cut "⇔-elim2" (hypothesis (p⇔q))
     ))
```

Next, for each form of conclusion we specify the tactic to execute in response to a hit.

```
CONCHIT ∀x:a•q   IS QuantIntroRule "∀-intro"
CONCHIT ∃x:a•q   IS QuantIntroRule "∃-intro"
CONCHIT ¬p       IS "¬-intro"
CONCHIT true     IS "true-intro"
CONCHIT p⇒q      IS "⇒-intro"
CONCHIT p⇔q      IS "⇔-intro"
CONCHIT p∧q      IS "∧-intro"
```

```
CONCHIT q∨r      IS
 (ALT  (SEQ "∨-intro1" hypothesis)
       (SEQ "∨-intro2" hypothesis)
       (MESSAGE (Neither disjunct q nor disjunct r is available)))

CONCHIT  p ⊢ p IS
(WHEN (LETHYP _p (WITHHYPSEL hypothesis))
      (FAIL ("Please select the hypothesis you want to use")))
```

A hit on a disjunction attempts to find a hypothesis which matches one of the disjuncts. If there isn't one, then the user may need to choose a particular branch of the disjunction to follow.

4.2 Buttons

Next we extend the `Edit` menu with a few buttons and switches which control various tactic-valued variables and some of Jape's internal variables.

The first pair of buttons allows the user to choose between box and sequent-tree presentations; the first switch determines whether or not Jape selects subgoals on behalf of the user; the second switch controls how fussy Jape is about the use of unproven theorems (conjectures) as derived proof rules.

```
MENU Edit
    RADIOBUTTON displaystyle
         "Sequent-Tree display style"  IS tree
    AND  "Box display style"           IS box
    INITIALLY
         box
    END

    CHECKBOX autoselect
            "Automatic goal selection"
    INITIALLY
            true

    CHECKBOX applyconjectures
            "Unproven theorems may be applied"
    INITIALLY
            false
```

Use of a button-controlled variable whose value is a tactic is a simple way of introducing modal behaviour into an interface. The next pair of buttons we place on the menu allow an intrepid user to enable the "backward" use of elimination rules. Such rules are usually used forwards, after selecting a formula which qualifies as an assumption

```
    RADIOBUTTON ElimRuleBackward
        "Forbid backward use of elimination rules" IS ElimRuleBackwardNo
    AND "Permit backward use of elimination rules" IS ElimRuleBackwardYes
    END

END /* MENU Edit */

TACTIC ElimRuleBackwardYes(rule)
       (WITHARGSEL rule)

TACTIC ElimRuleBackwardNo(rule)
(FAIL ("Select a goal then a non-grey formula before invoking %s",
       rule))
```

4.3 Low-level Tactics

Finally we define some of the low-level tactics which are mostly abbreviations for primitive Jape actions.

```
TACTIC FAIL(x) IS
       (JAPE (fail x))
TACTIC FAILSEL(x, y) IS
       (WHEN (LETARGSEL _p (JAPE (fail x))) (JAPE (fail y)))
TACTIC MESSAGE(m) IS
(SEQ (JAPE(setcomment m))
     (JAPE(setreason m)))

TACTIC COMMENT(x) IS (JAPE(setcomment x))
TACTIC SKIP(x)    IS (SEQ)

MENU    Edit
        SEPARATOR
        RADIOBUTTON Comment IS
                "Comments on during tactics"  COMMENT
        AND     "Comments off during tactics" SKIP
        END
END
```

A Some Predefined Problems

When Jape is used as a teaching tool, the teacher usually makes available a collection of proof problems (conjectures) which students should prove. A student starts proving a conjecture by selecting it from from a list which appears on a *conjecture panel.*

A conjecture panel displays a list of entries – the names of conjectures (unproved theormes) and theorems. One entry at a time can be selected (by clicking on it), and the panel has a button to start a proof of the currently-selected entry and a button to display an existing proof of the currently-selected entry.

```
CONJECTUREPANEL "Conjectures" IS
    THEOREMS Propositional(p,q,r,s) ARE
        ¬¬p ⊢ p
    AND ¬(p∨q) ⊢ ¬p ∧ ¬q
    AND ¬(p∧q) ⊢ ¬p ∨ ¬q
    AND ⊢ p ∨ ¬p
    AND p , ¬p ⊢ false
    AND p ∧ ¬p ⊢ false
    AND p ⊢ q⇒p
    AND ⊢ p⇔p
    AND ¬ ¬ p ⇔ p
    AND (p⇒q) ⇔ (¬p∨q)
    AND (p⇒q) ⇔ (¬q⇒¬p)
    AND ⊢ (p⇒q) ∨ (q⇒p)
    AND (p∧q ⇒ r) ⇒ (p ⇒ (q ⇒ r))
    AND p ⇒ q ⊢ p∧q ⇔ p
    AND ⊢ p ⇒ (q ⇒ p)
    AND p ⇒ q, q ⇒ r , p ⊢ r
    AND p ⇒ q ⊢ (q ⇒ r) ⇒ (p ⇒ r)
    AND p ⇒ (q ⇒ r), p ⇒ q, p ⊢ r
    AND p ⇒ q, q ⇒ r ⊢ p ⇒ r
    AND p ⇒ (q ⇒ r) ⊢ q ⇒ (p ⇒ r)
    AND p ⇒ (q ⇒ (r ⇒ s)) ⊢ r ⇒ (q ⇒ (p ⇒ s ))
    AND p ⇒ (q ⇒ r) ⊢ (p ⇒ q) ⇒ (p ⇒ r)
    AND ⊢ (p ⇒ (q ⇒ r)) ⇒ ((p ⇒ q) ⇒ (p ⇒ r))
    AND (p ∧ q) ⇒ r ⇔  p ⇒ (q ⇒ r)
    AND q ⇒ r ⊢ (p ∨ q) ⇒ (p ∨ r)
    AND true  ⇔ ¬false
    AND ¬true  ⇔ false
    AND p∧true ⇔ p
    AND p∨true ⇔ true
    AND p∧false ⇔ false
```

```
AND p∨false ⇔ p
AND (p⇒p) ⇔ true
AND (p⇒true) ⇔ true
AND (true⇒p) ⇔ p
AND (p⇒false) ⇔ ¬p
AND (p∧p) ⇔ p
AND (p∧q) ⇔ (q∧p)
AND (p∧(q∧r)) ⇔ ((p∧q)∧r)
AND (p∨p) ⇔ p
AND (p∨q) ⇔ (q∨p)
AND (p∨(q∨r)) ⇔ ((p∨q)∨r)
AND p ∨ (q ∨ r) ⇔ (p ∨ q) ∨ r
AND p ∧ (q ∨ r) ⇔ (p ∧ q) ∨ (p ∧ r)
AND p ∨ (q ∧ r) ⇔ (p ∨ q) ∧ (p ∨ r)
AND ¬(p∨q) ⇔ ¬p ∧ ¬q
AND ¬(p∧q) ⇔ ¬p ∨ ¬q
AND (p⇒q) ∧ (q⇒r) ⇒ (p⇒r)
AND (p⇔q) ∧ (q⇔r) ⇒ (p⇔r)
AND ¬p ∧ p ⊢ false
AND ¬(p⇔q)⇔((p∨q)∧¬(p∧q))
END
THEOREMS Predicate
ARE ∀x:a • p∧q ⊢ (∀x:a • p) ∧ (∀x:a • q)
AND (∀x:a • p) ∧ (∀x:a • q) ⊢ ∀x:a • p∧q
AND ¬(∃ x:a•q) ⊢ (∀ x:a•¬q)
AND ¬(∀ x:a•q) ⊢ (∃ x:a•¬q)
AND ¬(∀ x:a|p•q) ⊢ (∃ x:a|p•¬q)
AND WHERE x NOTIN a,a',y AND y NOTIN a,a',x INFER
    ∃x:a • ∀y:a' • q ⊢ ∀y:a' • ∃x:a • q
END
THEOREM "∀|•-intro" IS ∀ x:a • p ⇒ q ⊢ ∀ x:a | p • q
THEOREM "∀|•-elim"  IS ∀ x:a | p • q ⊢ ∀ x:a • p ⇒ q
THEOREM "∃|•-intro" IS ∃ x:a • p ∧ q ⊢ ∃ x:a | p • q
THEOREM "∃|•-elim"  IS ∃ x:a | p • q ⊢ ∃ x:a • p ∧ q
```

Whilst a kind teacher might not place any proof problems which are non-theorems on the list, a conscientious one ought to do so – perhaps giving them special names in an attempt to be kind.

```
THEOREM "?-∃x:a• ∀y:a'•x≤y⊢ ∀y:a'• ∃x:a•x≤y(no provisos)"
IS ∃ x:a • ∀y:a' • x≤y ⊢ ∀y:a' • ∃x:a • x≤y
THEOREM "?- (p⇒q)⇒p" IS (p⇒q)⇒p
THEOREM "?- (p∨¬p)⇒¬p" IS (p∨¬p)⇒¬p
```

```
    THEOREM "?- ∀y:a'• ∃x:a•q ⊢ ∃x:a• ∀y:a'•q"
    WHERE x NOTIN a,a',y AND y NOTIN a,a',x
    IS ∀y:a'• ∃x:a•q ⊢ ∃x:a• ∀y:a'•q
```

We want to add two more buttons: `Apply Forward` – which uses the currently-selected entry as a forward proof rule, and `Apply Backwards` which uses the currently-selected entry as a backwards proof rule.

```
    BUTTON "Apply Forward"   IS apply ThmForward COMMAND
    BUTTON "Apply Backwards" IS apply ThmBackward COMMAND
END /* PANEL */
```

The tactic `ThmForward` tries to apply a theorem (or conjecture) as a derived proof rule in the style of an elimination rule.

```
    TACTIC ThmForward(thm) IS (SEQ cut (WITHHYPSEL (WITHARGSEL thm)))
```

The tactic `ThmBackward` tries its argument backwards:

```
    TACTIC ThmBackward(thm) IS (WITHARGSEL(APPLYORRESOLVE thm))
```

Normally the application of a theorem to a goal requires that the bag of assumptions in the goal match the bag of assumptions of the theorem exactly. If a `WEAKEN` rule has been declared, then the theorem's assumptions need only be a subset of the goal's assumptions – Jape does the weakening itself. The `APPLYORRESOLVE` tactic form modifies this behaviour somewhat: it applies its component tactic – which should be a theorem, a conjecture or an antecedent-free proof rule – in such a way that any assumptions of the tactic which aren't already present in the goal become additional subgoals. Thus, for example, applying the theorem

$$\frac{x \leq y \quad y \leq z}{x \leq z}\ [\leq\text{-trans}]$$

in the proof state

$$\begin{array}{c} \lceil\, b \leq c \,\rceil \\ \vdots \\ a \leq c \end{array}$$

yields the proof state

$$\begin{array}{c} \lceil\, b \leq c \,\rceil \\ \vdots \\ \dfrac{a \leq b}{a \leq c}\ [\leq\text{-trans}] \end{array}$$

B Rewriting

We want to be able to select an equivalence theorem in the conjectures panel, and text-select a subformula of the current goal to be rewritten using the equivalence. What's more, if there is no text selection, then we'd like JAPE to find a place at which the equivalence theorem can be applied automatically.

The following rules (all but the first can be proven using the first) support proof by rewriting.

```
RULE    rewrite(p, OBJECT x)
FROM    p ⇔ q
AND     r[q/x]
INFER   r[p/x]

RULE    rewritebackwards(p, OBJECT x)
FROM    q ⇔ p
AND     r[q/x]
INFER   r[p/x]

RULE    hyprewrite(p, OBJECT x)
FROM    p ⇔ q
AND     r[q/x] ⊢ s
INFER   r[p/x] ⊢ s

RULE    hyprewritebackwards(p, OBJECT x)
FROM    q ⇔ p
AND     r[q/x] ⊢ s
INFER   r[p/x] ⊢ s
```

Before explaining the interface to these rules, we give a simple example of the way in which one of them can be used, together with an equivalence theorem, to rewrite a subformula of the current. goal.

Suppose the current goal is

$$(\neg b \Rightarrow \neg a) \Rightarrow (a \Rightarrow b)$$

Application of the rule `rewrite` with argument $\neg b \Rightarrow \neg a$ yields a tree of the form

$$\frac{(\neg b \Rightarrow \neg a) \Leftrightarrow \_q \quad \_q \Rightarrow (a \Rightarrow b)}{(\neg b \Rightarrow \neg a) \Rightarrow (a \Rightarrow b)}\text{[rewrite]}$$

If this application is part of a sequential tactic, then the left subtree

$$(\neg b \Rightarrow \neg a) \Leftrightarrow \_q$$

gets selected as the next goal. If the equivalence theorem

$$(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$$

is then applied to this goal, then the tree becomes

$$\dfrac{\dfrac{}{(\neg b \Rightarrow \neg a) \Leftrightarrow (\neg\neg b \vee \neg a)}\,[(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)] \qquad (\neg\neg b \vee \neg a) \Rightarrow (a \Rightarrow b)}{(\neg b \Rightarrow \neg a) \Rightarrow (a \Rightarrow b)}\,[\text{rewrite}]$$

If we display only the right-hand subtree, we see a proof which embodies the essence of a subformula rewrite

$$\dfrac{(\neg\neg b \vee \neg a) \Rightarrow (a \Rightarrow b)}{(\neg b \Rightarrow \neg a) \Rightarrow (a \Rightarrow b)}\,[(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)]$$

The rewriting machinery which we describe below generalises the sequential application of a rewrite rule followed by an equivalence theorem to cover most of the situations which are likely to arise during a hand-guided proof.

The next declaration adds two buttons to the conjectures panel: these provide the principal interface to the rewriting machinery by applying the selected entry (COMMAND) as a left-to-right, or a right-to-left rewrite rule.

```
CONJECTUREPANEL Conjectures IS
 BUTTON "Rewrite →" IS
  apply RewriteWith hyprewrite rewrite COMMAND
 BUTTON "Rewrite ←" IS
  apply RewriteWith hyprewritebackwards rewritebackwards COMMAND
END
```

The tactic `RewriteWith` implements these buttons

```
TACTIC  RewriteWith(hyprewriterule, rewriterule, thm) IS
(ALT
  (LAYOUT "rewrite %s" (1)
    (WHEN
      (LETHYPSUBSTSEL  _p (WITHSUBSTSEL hyprewriterule) thm)
      (LETCONCSUBSTSEL _p (WITHSUBSTSEL rewriterule) thm)
      (LETHYP          _p (hyprewriterule _p) thm)
      (LETCONC         _p (rewriterule _p) thm)))
  (FAILSEL ("%s not applicable to text selection", thm)
          ("%s not applicable to hypothesis or conclusion", thm)))
```

It has three arguments,

- `hyprewriterule` – this is used to rewrite a subformula of a hypothesis when one is selected: it is one of the rules `hyprewrite`, `hyprewritebackwards`.
- `rewriterule` – this is used to rewrite a subformula of the conclusion: it is one of the rules `rewrite`, `rewritebackwards`.
- `thm` – this is the equivalence theorem which has been selected in the panel.

The tactic `RewriteWith` first checks (`LET..SEL`) whether there is a text selection in a hypothesis or conclusion, and if there is then it applies the appropriate rewrite rule followed by the theorem. If there is no text selection, then it checks to see whether there's a selected hypothesis, or conclusion: if so then it attempts to rewrite it using the given rewrite rule. Otherwise `RewriteWith` fails, giving an appropriate error message (`FAILSEL...`).

`RewriteWith` is designed to lay the resulting proof out in a way which is convenient to read – the tactic form (`LAYOUT .. (1) ..`) hides the record of the proof of the left antecedent of the substantive rewrite rule which is applied.

We can also use the machinery described above to implement buttons on the quantifier rules menu which support the unfolding and folding of the abbreviations of the complex quantifier forms ∃|• and ∀|•.

```
TACTIC UnfoldWithIff(rule)
(RewriteWith  hyprewrite rewrite rule)

TACTIC FoldWithIff(rule)
(RewriteWith  hyprewritebackwards rewritebackwards rule)

MENU "Quantifier Rules" IS
    SEPARATOR
    ENTRY "∃|•→"      IS UnfoldWithIff "∃|•"
    ENTRY "∃|•←"      IS FoldWithIff   "∃|•"
    ENTRY "∀|•→"      IS UnfoldWithIff "∀|•"
    ENTRY "∀|•←"      IS FoldWithIff   "∀|•"
END
```

References

[1] Richard Bornat. *Using ItL* JAPE
http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/JAPE/itljape.ps.gz

[2] Richard Bornat & Bernard Sufrin. *Roll Your Own* JAPE *Logic*
http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/JAPE/rollyourownjapelogic.ps.gz

[3] Richard Bornat & Bernard Sufrin. *The Oxford* JAPE *Web Site*
http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/jape.html

[4] Fitch, F.B.. *Symbolic Logic.* Ronald Press, New York

[5] Bernard Sufrin & Richard Bornat. J'N'J *in Jape*
http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/JAPE/jnj.ps.gz

[6] Bernard Sufrin & Richard Bornat. *Using* J'N'J *in Unix/X* JAPE
http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/JAPE/usingjnj.ps.gz

[7] Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall International

Manuscript Revision: 1.28

Date: 1998/07/17 16:52:46

Latexed: June 8, 2000 at 16:47

Contents