Programming Research Group

Using J'n'J in Jape
Natural Deduction using the Jape Proof Editor

Bernard Sufrin
Richard Bornat

PRG-TR-10-98

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

Abstract

In this note we describe, by means of a number of case studies, the user interface to the JAPE implementation – "J'N'J" – of the logic described in [7] ("the book").

The introduction is intended for someone who may not have used JAPE before. It assumes the reader has a basic grasp of the Unix operating system and the X window system and has read section 1 of [5] – which is a companion to this note..

1 Getting Started with J'N'J

BEFORE doing anything else at all, you should have a look at the first few chapters of the book, and read section 1 of the companion to this document [5]. The book explains the logic and the companion paper explains the relationship between the natural-deduction style of presentation used in the book and the style of presentation which JAPE normally uses for the logic.

First find out where the JAPE system lives in your filestore – on Solaris machines in the Computing Lab at Oxford it will be somewhere with a name like `/PACK/jape/default`. Next, make sure that your `PATH` contains the JAPE binary directory: `/PACK/jape/default/bin`.

Once you've done so, and providing you are running X-windows on your machine,[1] you will be able to start JAPE with the Unix command

```
jape  /PACK/jape/default/examples/jnj.jt
```

After a short pause, the terminal emulator to which you gave the command should give some sort of indication that JAPE is starting up. Expect to see something like this

```
Jape proof engine 2003/01/11 [/auto/users/sufrin/JAPE6/japeserver]
[OPENING "/users/sufrin/JAPE6/examples/jnj/jnj.jt"]
[CLOSING "/users/sufrin/JAPE6/examples/jnj/jnj.jt"]
```

Windows, Linux, MacOsX

If you're working on a Linux, Windows, or MacOsX machine of your won, then you can find the latest Jape distribution through the Jape Web page.

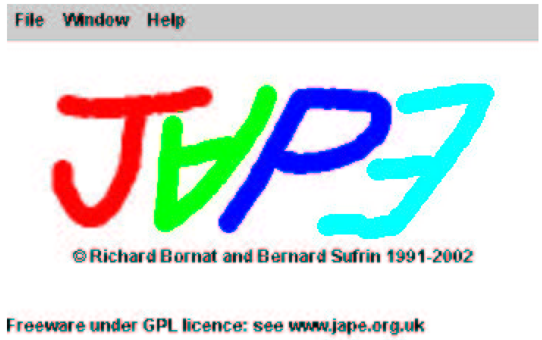
`www.jape.org.uk`

Follow the installation instructions there.
On Linux machines you can run JAPE *in much the same way as on Solaris machines.*
On the Windows and MacOsX machines you can start JAPE *by doubleclicking on its icon. You'll soon see the* JAPE *window appear, and you can use the* `Open New Theory` *button on the* `File` *menu to start a theory-selection dialogue.*

Unless something is badly wrong, you will soon see two windows come up on your screen.

[1] If you don't know what X-Windows is, or how to set your `PATH`, then consult a system wizard.

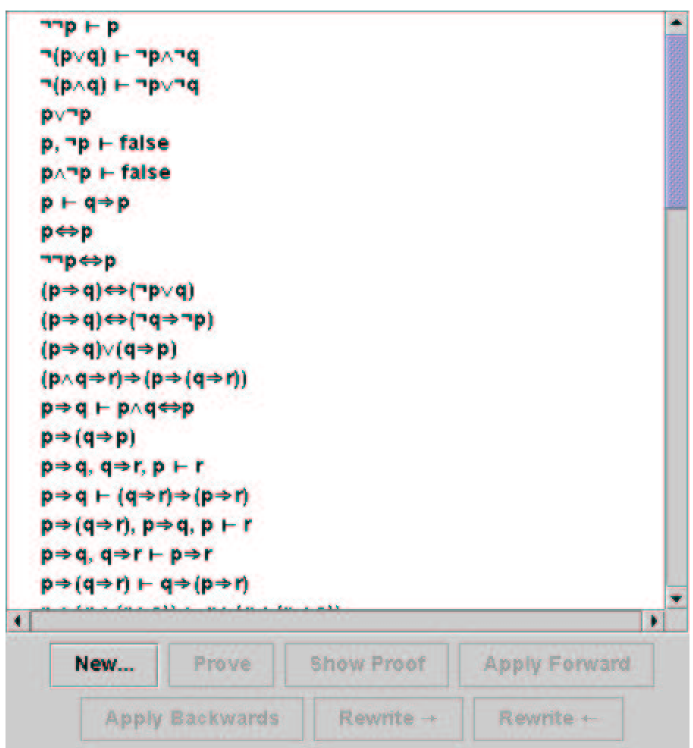
The Jape Window



This is the top-level controlling window for the session. You can use it to load new theories, to add parts of theories to theories you are already working with, and to switch proof windows.

The Conjectures Window

The Conjectures window may take a while to arrive



It consists mainly of a scrolled list of the conjectures which the J'N'J implementation designer decided would be worthwhile attempting to prove.

There are also several buttons at the foot of the window, all of which but `New` are greyed-out: their function will be explained as we proceed.

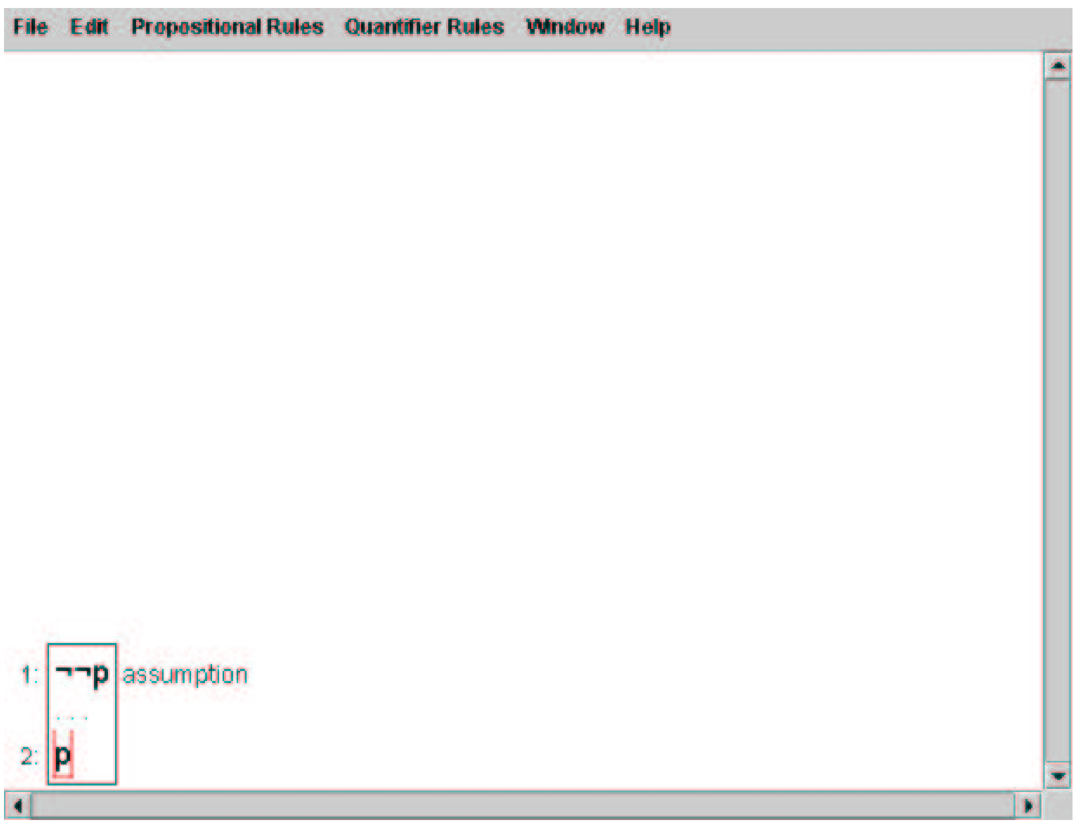
Proof Session Windows

The most important kind of window is the proof display and control panel – known as the Proof Session Window. You won't see a session window until you choose a conjecture to try to prove.

For example, to start proving the conjecture

$$\frac{\neg\neg p}{p}$$

we select the entry $\neg\neg p \vdash p$ from the conjecture list with a `Click` [2], and then `Click` on the **`Prove`** button. Very soon afterwards the following session window appears:[3]



[2]Pointing to it with the mouse and pressing the left-hand button once.

[3]In some earlier versions of JAPE there was only ever one Session window. In the current version there is one Session window per active proof.

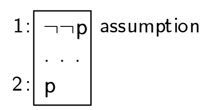
When you are conducting a proof with JAPE it always shows a view of the current (partial) proof on the session window. The `File` and `Edit` menus are permanent fixtures of all Session windows, and the `Propositional Rules` and `Quantifier Rules` menus are features of the theory we are working in. These are the menus that have the J'N'J proof rules on them.

The ellipsis between the line numbered 1 ($\neg\neg p$) and the line numbered 2 (p) is JAPE's way of telling you that the proof of the conclusion p from the assumption (premiss) $\neg\neg p$ is still incomplete. We will complete it as our first case study.

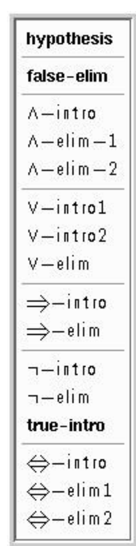
2 Case Studies

2.1 A Proof by Forward Reasoning

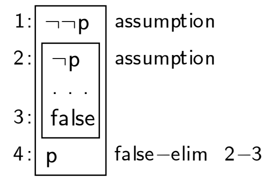
We start with the following partial proof.



Because we've done the proof before, we know that it's a proof by contradiction, so we decide to apply [false-elim]. To do so we `Click` on the `Propositional Rules` menu button, which pops up the following menu,



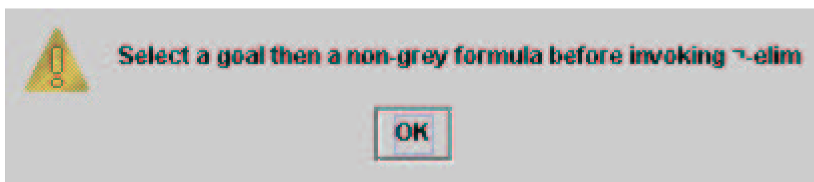
We apply the rule by clicking on its name, and the proof changes to



The next move should be to apply

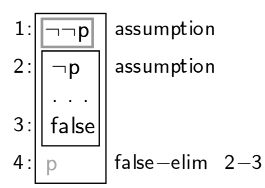
$$\frac{\neg p \quad p}{\mathit{false}}\ [\neg\text{-elim}]$$

If we invoke the rule from the menu, then JAPE complains – using a dialogue window



The J'N'J implementation insists that at least one assumption-like[4] formula be selected when an elimination rule is to be applied.[5] Where more than one assumption is present which can be simplified by an application of the rule, this makes it clear which one is to be simplified.

We acknowledge the complaint by clicking on the `OK` button, and then select the assumption $\neg\neg p$ by clicking on it, whereupon the proof changes to



The box around the formula shows it has been selected,[6] and the p on line 4 turns grey to show that it is not in the scope of the selected assumption. We can then apply the rule by clicking on its menu entry, and this yields the complete proof

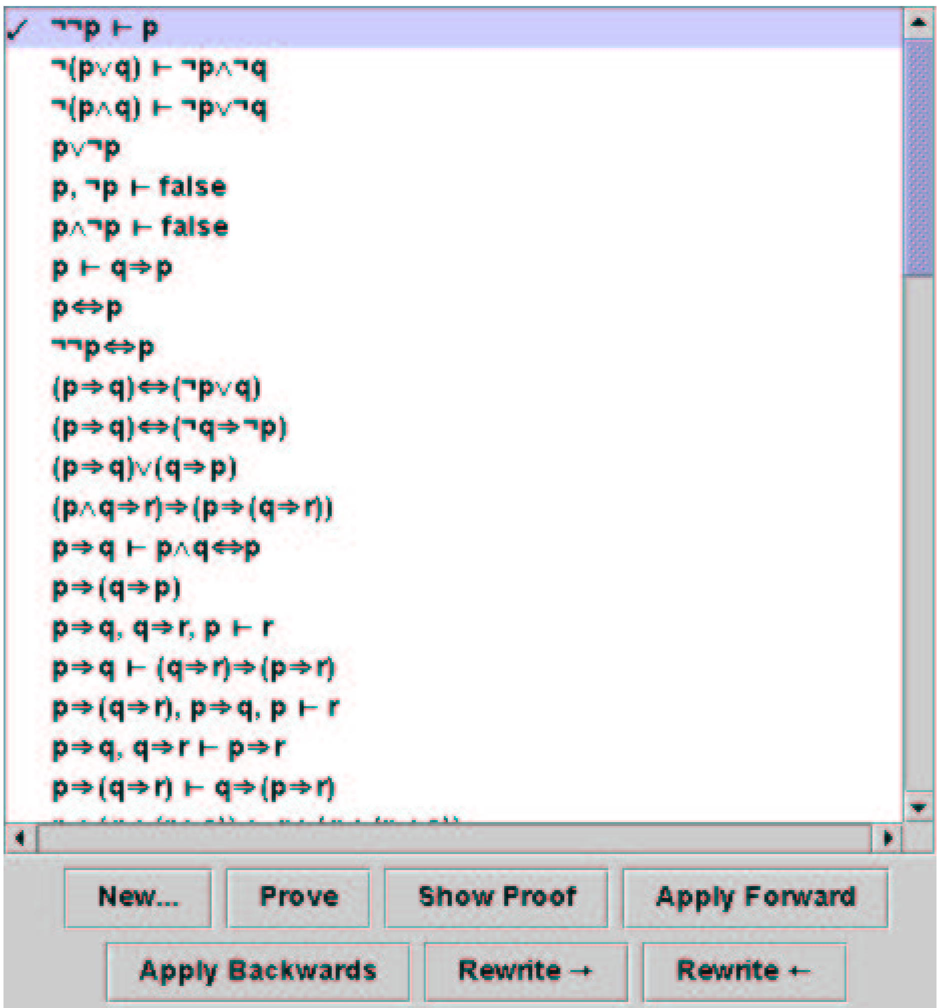
[4]An *assumption-like* formula is either an assumption or a formula which has been derived by means of an elimination rule from an assumption-like formula.

[5]The exception to this rule is [false-elim].

[6]It may show in a different colour on the computer screen.

1:	¬¬p	assumption
2:	¬p	assumption
3:	false	¬−elim 1,2
4:	p	false−elim 2−3

At this point we pull down the `Edit` menu, and select the `Done` button. This registers the proof, and permits the theorem to be used in subsequent reasoning as if it were a proof rule.[7] The conjectures panel changes to reflect this (notice the tick):



[7] Pressing control-D on the keyboard when the cursor is in the session window has the same effect. This is what's known as a "keyboard shortcut".

2.2 Direct Manipulation

It is sometimes tedious to invoke rules from menus: it is particularly irritating to have to do so if there is only one possible rule worth applying once we have chosen a particular formula to work on. In fact if we need to work forwards from an assumption-like formula then we will usually want to apply the elimination rule for the main connective of the formula. If we want to work backwards from a conclusion, then we will usually want to apply the introduction rule for the main connective of the formula. The main exception to this is when we decide to do a proof by contradiction: in this case we will need to apply [false-elim], and there will be no obvious connection between the conclusion we're working towards and that rule.

So we have implemented a *direct manipulation* interface for J'N'J. This permits rules to be invoked by a combination of selecting and double-clicking.

To work forward from an assumption or a formula derived forward from an assumption, one double-clicks on the formula in question – whereupon the appropriate elimination rule for the formula in question is invoked. If the formula has more than one (unclosed) conclusion in its scope then one should select the conclusion one wishes to work towards before doing this – otherwise JAPE will complain. One can determine definitively which conclusions are in the scope of an assumption-like formula by selecting (single click) the formula: the conclusions which don't get greyed-out are the ones in scope.

To work backwards from an (unclosed) conclusion formula we double-click on it – whereupon the appropriate introduction rule for the formula is invoked. If the introduction rule for the main connective of the conclusion has antecedents (they usually do), and if the conclusion is in the scope of more than one assumption-like formulae which matches an antecedent (it often is), then one should select the particular assumption-like formula before invoking the rule – otherwise JAPE will either complain, or present one with a (so-called) choice dialogue: which invites one to choose the appropriate instance of the rule which is being applied. One can determine definitively which assumptions have scopes which include a conclusion by selecting (single click) the conclusion: the assumptions which don't get greyed-out are the ones in whose scope the conclusion lies.

The last move in the proof we have just completed could just as easily have been done by selecting $\neg\neg p$ then double-clicking on it; indeed since the first click of the double-click does the selection anyway, a simple double click is all that's necessary.

In the next example we will explore the direct manipulation interface a little more.

2.3 Two ways to find the same proof

There is often more than one way of finding the same proof (and, indeed, there is often more than one proof of a given conjecture). JAPE is designed to be flexible enough to help us explore many ways of finding proofs. In this section we find a proof of

$$\frac{p \Rightarrow q \quad q \Rightarrow r \quad p}{r}$$

in more than one way.

The first way

To start the proof we click on the entry $p \Rightarrow q, q \Rightarrow r, p \vdash r$ in the conjectures panel, then click on `Prove`. The session window displays the partial proof

1:	p⟹q , q⟹r, p	assumptions
	. . .	
2:	r	

The three assumptions are shown on the same line to conserve space – which is a precious commodity during a long proof. The first thing we do is to establish q, by double-clicking on $p \Rightarrow q$ thereby invoking [⇒-elim], or by selecting it with a single click then invoking the rule from the propositional rules menu. This leads to

1:	p⟹q , q⟹r, p	assumptions
2:	q	⟹−elim 1.1,1.3
	. . .	
3:	r	

Notice that the line-numbering has changed, and that line 2 is annotated with the name of the rule by which it was derived and the numbers of the assumptions the rule used.

We can now establish r by invoking the same rule – but this time using the formula $q \Rightarrow r$. We do this either by selecting $q \Rightarrow r$ and invoking the rule from the menu, or by double-clicking on $q \Rightarrow r$. This leads to

1:	p⟹q , q⟹r, p	assumptions
2:	q	⟹−elim 1.1,1.3
3:	r	⟹−elim 1.2,2

and the proof is complete and ready to be registered by clicking on the `Done` entry on the `Edit` menu.

A second way

If you registered the last proof, then select the same conjecture on the conjectures panel, and press the **Prove** button again. JAPE will ask you, by means of a dialogue box, whether you really want to start another proof of something you have already registered: press the **Yes** button in the dialogue box. If you didn't register the last proof, then simply press the **Undo** button on the **Edit** menu a couple of times, or press the **Delete** key a couple of times and you will undo the last two proof moves.[8]

In any event the session window shows

```
1: p⟹q , q⟹r, p   assumptions
   . . .
2: r
```

Now invoke [⇒-elim] on the assumption $q \Rightarrow r$ either by selecting that assumption and invoking the rule from the menu, or by double-clicking on it.

```
1: p⟹q , q⟹r, p   assumptions
   . . .
2: q
3: r               ⟹−elim  1.2,2
```

We have reasoned forwards from an implication whose antecedent is not yet established, and JAPE– which is flexible enough to support this kind of reasoning – has shown us what the structure of a proof which will end with an application of this rule must be. The last detail is filled in by invoking [⇒-elim] on the assumption $p \Rightarrow q$.

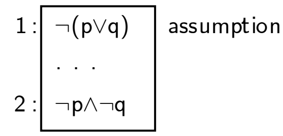
2.4 Mixing forward and backward reasoning

THE examples so far have all exploited forward reasoning with elimination rules. In the next example we will employ a mixture of forward and backward reasoning, and prove de Morgan's law

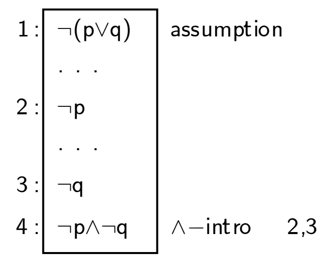
$$\frac{\neg(p \vee q)}{\neg p \wedge \neg q}$$

First we select it in the conjectures panel and press the **Prove** button.

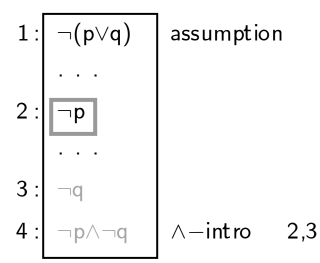
[8]You can re-do an undone move by pressing `Edit/`Redo or the `Tab` key.



We decide to split the task of proving the two conjuncts, and invoke [∧-intro] by double-clicking on $\neg p \wedge \neg q$.



We now have to establish both $\neg p$ and $\neg q$. A good heuristic (rule of thumb) for doing proofs is to use introduction rules backwards to simplify conclusions as far as possible before starting to reason forwards.[9] We decide to work on the former, and select it with a click. The proof changes to

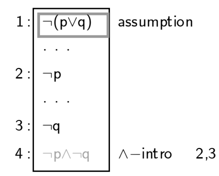


The grey box around $\neg p$ shows that we've selected it as the conclusion we are going to simplify. The greying-out of lines 3 and 4 signify that the formulae on those lines will not be useable in any proof of the selected formula, and the fact that the assumption on line 1 remains black indicates that the conclusion is within the scope of that assumption.

[9]Because it's only a rule of thumb, it doesn't cover every case – and in particular can't help us when the proof requires a [false-elim] move.

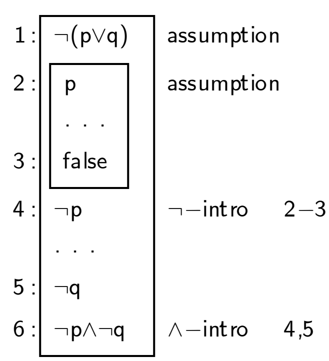
Grey Formulae

The scope of a selected hypothesis can be determined by selecting it. We can see this in the proof state described above by de-selecting ¬p (click anywhere on the proof window outside the area of the proof to do this) and selecting the assumption on line 1. This yields



The fact that lines 2 and 3 remain black indicates that both lines are in the scope of the selected assumption. (This is hardly a surprise, since it was a premiss of the entire conjecture, but in more complicated situations it is very useful to be able to discover the scope of a hypothesis which may have been introduced in one branch of a many-branched proof.)

So we invoke [¬-intro] by double clicking on $\neg p$, yielding



Observing that we have not yet simplified the assumption, we apply [¬-elim] by double clicking on line 1. This yields

1 :	¬(p∨q)	assumption
2 :	p	assumption
	. . .	
3 :	p∨q	
4 :	false	¬−elim 1,3
5 :	¬p	¬−intro 2−4
	. . .	
6 :	¬q	
7 :	¬p∧¬q	∧−intro 5,6

and we can see that all we need to do is to apply [∨-intro1]. We can do so by selecting line 3[10] and applying the rule from the menu. We can *also* do so by just double-clicking on line 3. This is because J'N'J tries *both* or-introduction rules when we double-click on a disjunctive conclusion.

1 :	¬(p∨q)	assumption
2 :	p	assumption
3 :	p∨q	∨−intro1 2
4 :	false	¬−elim 1,3
5 :	¬p	¬−intro 2−4
	. . .	
6 :	¬q	
7 :	¬p∧¬q	∧−intro 5,6

The $\neg q$ conclusion can be proven with exactly the same moves and this finishes the proof. We register it by pressing `Edit/`Done, and the conjectures window changes accordingly.

[10] We should, more properly, have written "selecting the *formula* on line 3", but this ponderous formulation quickly gets irritating when repeated. So when there's only one formula on a line we'll refer to the line as if it were the formula.

Grey Formulae

After the first two steps in the proof of q we reach a state where we can select the conclusion $p \vee q$ again. If we do so, the proof window greys out the lines which cannot be used in the proof of this conclusion.

1 :	¬(p∨q)	assumption	
2 :	p	assumption	
3 :	p∨q	∨−intro1	2
4 :	false	¬−elim	1,3
5 :	¬p	¬−intro	2−4
6 :	q	assumption	
	. . .		
7 :	p∨q		
8 :	false	¬−elim	1,7
9 :	¬q	¬−intro	6−8
10 :	¬p∧¬q	∧−intro	5,9

2.5 Using Theorems as Derived Rules

Netx we shall prove the law of the excluded middle

$$\overline{p \vee \neg p}$$

We start the proof in the usual way.

	. . .
1 :	p∨¬p

We are going to prove it by contradiction so we invoke [false-elim] from the menu.

1 :	¬(p∨¬p)	assumption	
	. . .		
2 :	false		
3 :	p∨¬p	false−elim	1−2

Next we shall invoke de Morgan's law – the one we just proved. We do so by selecting it in the conjectures panel, and pressing the `Apply Forward` button in that panel.[11]

[11] We couldn't have applied it backwards, because its conclusion doesn't match the goal *false* which we are working towards.

1 :	$\neg(p\vee\neg p)$	assumption	
2 :	$\neg p\wedge\neg\neg p$	Theorem $\neg(p\vee q)\vdash\neg p\wedge\neg q$	1
	...		
3 :	false		
4 :	$p\vee\neg p$	false−elim	1−3

We notice that the conjunction on line 2 is a contradiction, and that we will therefore be able to prove *false* from it. But there is no *direct* way of doing so. What we will have to do is to dismember the conjunction into its components by applying both $\wedge$-elimination rules to line 2, then use [$\neg$-elim]. The first part of this task is as easily accomplished as the second – simply double-click on line 2, and – because there isn't an obvious way of using just one of the elimination rules,[12] J'N'J uses them both in sequence, yielding

1 :	$\neg(p\vee\neg p)$	assumption	
2 :	$\neg p\wedge\neg\neg p$	Theorem $\neg(p\vee q)\vdash\neg p\wedge\neg q$	1
3 :	$\neg p$	$\wedge$−elim1	2
4 :	$\neg\neg p$	$\wedge$−elim2	2
	...		
5 :	false		
6 :	$p\vee\neg p$	false−elim	1−5

We leave it as an exercise for the reader to decide which line to double click on to invoke the appropriate instance of [$\neg$-elim].

2.6 Proof Unknowns, Backward Elimination

We will need to exploit a few more sophisticated features of JAPE during our next case study, which is a bit more ambitous.

...

1 : $(p\Longrightarrow q)\vee(q\Longrightarrow p)$

[12]If our goal had been $\neg p$ then there would have been an obvious way of using [$\wedge$-elim1]; if it had been $\neg\neg p$ then there would have been an obvious way of using [$\wedge$-elim2].

Automatic Goal Selection

Before starting the proof we invited the system to do some of our work for us by selecting conclusions for us to work towards. We do so by pressing `Edit/`*Automatic Goal Selection, and thereafter* JAPE *always selects the first unsolved conclusion generated after the most recent proof move; or (if there are no such conclusions) the topmost leftmost open conclusion in the proof as a whole. The selected conclusion is shown here with a grey box around it (*JAPE *may use a different colour). If we don't like the conclusion which* JAPE *selects, then we can select another manually.*

We start the proof by observing that it is unlikely that the last line of our proof will involve an or introduction. We therefore suppose that it's a good idea to try and prove a contradiction, so we apply [false-elim] immediately followed by [¬-elim], yielding

1 :	¬((p⟹q)∨(q⟹p))	assumption	
	. . .		
2 :	(p⟹q)∨(q⟹p)		
3 :	false	¬−elim	1,2
4 :	(p⟹q)∨(q⟹p)	false−elim	1−3

The latter rule can be applied either by double-clicking or by selecting the hypothesis $\neg((p \Rightarrow q) \vee (q \Rightarrow p))$ and invoking the rule from the menu.

Now we do some case analysis: if $\neg q$ holds, then so does $q \Rightarrow p$, and if q holds then so does $p \Rightarrow q$. We haven't proven either of these facts yet, but we do know that [∨-elim] is the basis for a proof by case analysis. On the other hand there's not yet an ∨ to eliminate: so what can we do? If we try to use [∨-elim] the system will simply complain that we haven't selected an assumption-like formula. So we take a novel step: we press

`Edit/Permit backward use of elimination rules`

and then apply [∨-elim]. This yields

1 :	¬((p⟹q)∨(q⟹p))	assumption
	. . .	
2 :	_p2∨_q1	
3 :	_p2	assumption
	. . .	
4 :	(p⟹q)∨(q⟹p)	
5 :	_q1	assumption
	. . .	
6 :	(p⟹q)∨(q⟹p)	
7 :	(p⟹q)∨(q⟹p)	∨−elim 2,3−4,5−6
8 :	false	¬−elim 1,7
9 :	(p⟹q)∨(q⟹p)	false−elim 1−8

JAPE has introduced two *formula unknowns* into the proof: they are prefixed with an underscore (to signify that they are unknowns), and a formula-class name (to signify that when they become known they will be formula). By introducing these unknowns JAPE has allowed us to delay taking a decision about the exact instance of the rule which we want to apply.

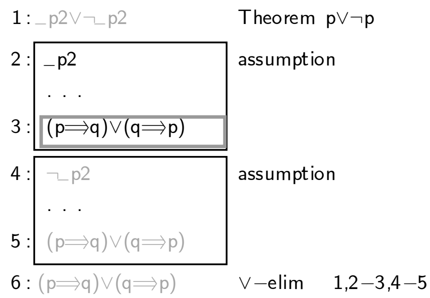
Our original analysis suggested two proof branches: one starting with q and the other starting with $\neg q$, and at the moment branches 3-4 and 5-6 are both consistent with that analysis, but are not in quite the right form.

There are several things we can do at this point, and we choose to use the theorem $p \vee \neg p$ backwards at line 2, by selecting it in the conjectures panel, then pressing the `Apply Backward` button. This yields

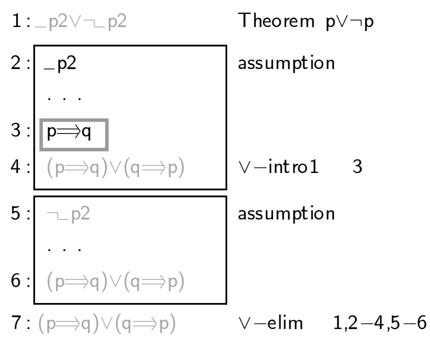
1 :	¬((p⟹q)∨(q⟹p))	assumption
2 :	_p2∨¬_p2	Theorem p∨¬p
3 :	_p2	assumption
	. . .	
4 :	(p⟹q)∨(q⟹p)	
5 :	¬_p2	assumption
	. . .	
6 :	(p⟹q)∨(q⟹p)	
7 :	(p⟹q)∨(q⟹p)	∨−elim 2,3−4,5−6
8 :	false	¬−elim 1,7
9 :	(p⟹q)∨(q⟹p)	false−elim 1−8

which has only one formula unknown in it.

At this point we realize that we don't, after all, need to do a proof by contradiction, and that our early decision was just wrong. We therefore go back to the beginning of the proof by repeatedly pressing the `Edit/`Undo button.[13] We now repeat the two moves we were confident about: [∨-elim] and the backward application of the theorem $p \vee \neg p$. This yields



Now we invoke [∨-intro1], yielding



An interlude – two lemmas

Now we decide it's time to prove the two lemmas[14] which arise naturally from our earlier discussion, namely

$$\frac{q}{p \Rightarrow q}$$

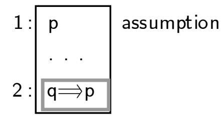
and

$$\frac{\neg\, q}{q \Rightarrow p}$$

The former is an instance of something which is already on the conjectures panel, so we interrupt the current proof by selecting $p \vdash q \Rightarrow p$ from that panel and pressing `Prove`. The session window now shows

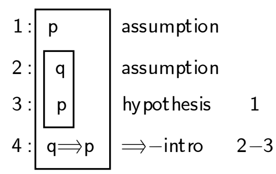
[13]We could have achieved the same effect in X/Jape by pressing the `Delete` key repeatedly, or by selecting line 9 and pressing the `Edit/`Backtrack key.

[14]A lemma is just a theorem which is used in the proof of another theorem.



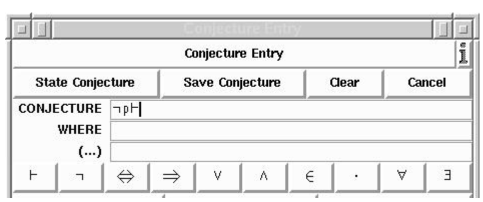
and, if we click on the `Switch Proof` menu we see that there are two proofs in progress. We could switch between them using this menu, or we could abandon the one which is currently in the session window, but we won't do so now.

The lemma yields to an invocation of [⇒-intro][15]



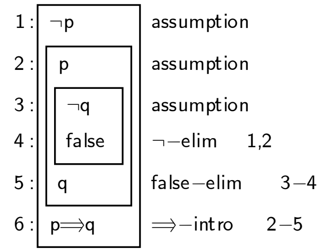
The *hypothesis* rule used to generate line 3 is discussed at length in section 2 of [5], and its automatic application is discussed later in that note. If we register this proof by pressing `Edit/`Done then the proof we were working on reappears.

The second lemma doesn't appear on the conjectures list. We can add it to that list by pressing `(New)` on the conjectures panel. We are greeted with a conjecture entry form which looks something like this.



We enter the conjecture $\neg q \vdash q \Rightarrow p$ on the line labelled `CONJECTURE`, using the buttons at the foot of the form to enter symbols which don't appear on the keyboard. When we have done so, we press the `State Conjecture` button on the form, whereupon the conjecture we stated is added to the conjectures panel whence it can be selected for proof in the usual way. Its proof is

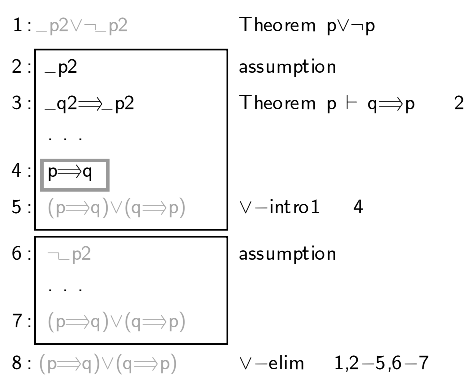
[15]By this time we hope that the reader understands the nature of the interface well enough that we need no longer be explicit about whether we double-clicked or used the menu.



We leave its discovery as an adventure for the reader.

Back to the proof

We are now in a position to use the first lemma forward from line 2. We do so by clicking on line 2 to select it, ensuring that line 3 is selected as our goal conclusion, then selecting the lemma on the conjectures panel, and pressing `Apply Forward`. This yields



There are still formula unknowns present, but by this time we know what we want them to be. There are several ways in which we can force them to be what we want, but the simplest is to select line 3 (which is assumption-like, because it was derived forward from the assumption on line 2), ensure that line 4 is also selected (it's a conclusion), then invoke the [*hypothesis*] rule from the `Propositional Rules` menu. This forces (by unification) line 3 and line 4 to be identical, thereby resolving the values of all the remaining unknowns.

1 :	q∨¬q	Theorem p∨¬p
2 :	q	assumption
3 :	p⟹q	Theorem p ⊢ q⟹p 2
4 :	(p⟹q)∨(q⟹p)	∨−intro1 3
5 :	¬q	assumption
	. . .	
6 :	(p⟹q)∨(q⟹p)	
7 :	(p⟹q)∨(q⟹p)	∨−elim 1,2−4,5−6

Working on the remaining branch of the proof, we can now apply the second lemma forward from the assumption on line 5 (select line 5, then select the lemma at the bottom of the conjectures panel and press the `Apply Forward` button). This yields

1 :	q∨¬q	Theorem p∨¬p
2 :	q	assumption
3 :	p⟹q	Theorem p ⊢ q⟹p 2
4 :	(p⟹q)∨(q⟹p)	∨−intro1 3
5 :	¬q	assumption
6 :	q⟹_q2	Theorem ¬p⊢p⟹q 5
	. . .	
7 :	(p⟹q)∨(q⟹p)	
8 :	(p⟹q)∨(q⟹p)	∨−elim 1,2−4,5−7

There is *still* an unknown present, because although we know where we are headed when we apply the lemma forwards, JAPE doesn't.[16]

But an application of [∨-intro2] – either from the menu or by double-clicking on line 7 – resolves the unknown and completes the proof.

1 :	q∨¬q	Theorem p∨¬p
2 :	q	assumption
3 :	p⟹q	Theorem p ⊢ q⟹p 2
4 :	(p⟹q)∨(q⟹p)	∨−intro1 3
5 :	¬q	assumption
6 :	q⟹p	Theorem ¬p⊢p⟹q 5
7 :	(p⟹q)∨(q⟹p)	∨−intro2 6
8 :	(p⟹q)∨(q⟹p)	∨−elim 1,2−4,5−7

[16]The fact that the unknown has a name identical to a previous unknown which appeared in the proof is entirely coincidental: the two uses of the name are unrelated. Once an unknown's value is completely resolved, its name becomes available for re-use.

2.7 A Predicate Logic Proof without Provisos

Next we shall prove one of the quantifier duality laws, namely

1:	$\neg(\exists x : a \bullet q)$	assumption
	. . .	
2:	$\forall x : a \bullet \neg q$	

If we double-click on line 2 – to invoke the [∀-intro] the proof becomes

1:	$\neg(\exists x : a \bullet q)$	assumption
2:	$y \in a$	assumption
	. . .	
3:	$\neg q\,[y/x]$	
4:	$\forall x : a \bullet \neg q$	∀−intro 2−3

Notice that a new variable y has been introduced into the proof.

Next we invoke [¬-intro], giving

1:	$\neg(\exists x : a \bullet q)$	assumption
2:	$y \in a$	assumption
3:	$q\,[y/x]$	assumption
	. . .	
4:	false	
5:	$\neg q\,[y/x]$	¬−intro 3−4
6:	$\forall x : a \bullet \neg q$	∀−intro 2−5

Now the only resource left for us to use is the assumption on line 1. Eliminating the negation gives

1:	$\neg(\exists x{:}a \bullet q)$	assumption
2:	$y \in a$	assumption
3:	$q\,[y/x]$	assumption
	$\ldots$	
4:	$\exists x{:}a \bullet q$	
5:	false	$\neg$–elim 1,4
6:	$\neg q\,[y/x]$	$\neg$–intro 3–5
7:	$\forall x{:}a \bullet \neg q$	$\forall$–intro 2–6

We can now see that there is an element in a which satisfies q, it's the y introduced on line 2 by the universal introduction rule.

We want to use the existential elimination rule

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

with an explicit argument for t, so that it avoids inventing a *new* term unknown. We can do so by *text-selecting* the name y which has the property we want.

Text Selection

A formula or part of a formula can be text-selected by moving the cursor into the formula. Then press the text-selection mouse-button at the left end of the subformula that is to be selected, and drag the mouse rightwards with the button pressed until the the desired subformula has been selected. One can also start at the right end and drag the mouse leftwards through the text to be selected. The selected material changes colour as the mouse moves. A slight overshoot can be rectified by moving the mouse back towards the start of the selection, and a big mistake (starting at the wrong place, for example) can be rectified by double-clicking the text-selection mouse button within *the formula from which the selection was being made.*
Clicking the left-hand mouse button on the session window anywhere outside of the proof cancels both the assumption and conclusion selections as well as any current text selections.

Now we invoke existential introduction at line 4, and this completes the proof.

1:	¬(∃x:a▪q)	assumption
2:	y∈a	assumption
3:	q[y/x]	assumption
4:	∃x:a▪q	∃−intro 2,3
5:	false	¬−elim 1,4
6:	¬q[y/x]	¬−intro 3−5
7:	∀x:a▪¬q	∀−intro 2−6

If we hadn't text-selected the y then JAPE would have invented an unknown, leaving us a bit more work to do

1:	¬(∃x:a▪q)	assumption
2:	y∈a	assumption
3:	q[y/x]	assumption
	. . .	
4:	_t∈a	
	. . .	
5:	q[_t/x]	
6:	∃x:a▪q	∃−intro 4,5
7:	false	¬−elim 1,6
8:	¬q[y/x]	¬−intro 3−7
9:	∀x:a▪¬q	∀−intro 2−8

Using the *hypothesis* rule with lines 3 and 5 selected, or with lines 2 and 4 selected will also finish the proof.

We can also finish it by text-selecting both the unknown ($\_t$), and y, and pressing the `Edit/Unify Selected Terms` button. This invokes a command which is not a formal proof rule, but which – by attempting to unify the terms which are text-selected, may resolve one or more unknowns to the point where the automatic application of *hypothesis* (which J'N'J does after every command: see Section 4.1 of [5]) can finish the proof.

Now that we have seen how to provide parameters to rules, we will go back and re-do the proof – this time without inventing the y on the first step. The first thing we do, then, is to text-select an x and apply [∀-intro].

1:	$\neg(\exists x : a \bullet q)$	assumption
2:	$x \in a$	assumption
	. . .	
3:	$\neg q$	
4:	$\forall x : a \bullet \neg q$	$\forall$−intro 2−3

Next we use [¬-intro] and [¬-elim] in sequence, yielding

1:	$\neg(\exists x : a \bullet q)$	assumption
2:	$x \in a$	assumption
3:	q	assumption
	. . .	
4:	$\exists x : a \bullet q$	
5:	false	¬−elim 1,4
6:	$\neg q$	¬−intro 3−5
7:	$\forall x : a \bullet \neg q$	$\forall$−intro 2−6

Now we text select an x, and invoke [∃-intro] on line 4, and this completes the proof

1:	$\neg(\exists x : a \bullet q)$	assumption
2:	$x \in a$	assumption
3:	q	assumption
4:	$\exists x : a \bullet q$	$\exists$−intro 2,3
5:	false	¬−elim 1,4
6:	$\neg q$	¬−intro 3−5
7:	$\forall x : a \bullet \neg q$	$\forall$−intro 2−6

2.8 Proofs with Provisos

2.8.1 A proof with invariant provisos

For our next case study we shall prove

```
1: ∃x:a•∀y:a′•q     assumption
   . . .
2: ∀y:a′•∃x:a•q
```

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

The provisos we start with simply aver that x and y are distinct, and that neither x nor y are free in a or a'. They will remain the same during the entire proof.[17]

We first construct an argument in support of the conjecture which will help guide us as we make the proof. Although it's sometimes possible to dispense with such proof plans when conducting a fully formal proof, and we've done so up to now in this document, having a plan can help us keep our bearings when the formal proof looks like it's going off the rails.

The argument is something like this:

1. There is an $x \in a$ such that q holds for every $y \in a'$: let this x be called y_1. We can therefore infer $y_1 \in a, \forall y : a' \bullet q[x \backslash y_1]$
2. We want to show that $\forall y : a' \bullet \exists x : a \bullet q$, and we can do this by picking an arbitrary $y_2 \in a'$, then showing that the y_1 we introduced in the first step is indeed an element of a for which $q[y \backslash y_2]$ holds.
3. So we show that $\exists x : a \bullet q[y \backslash y_2]$ by showing that $q[x, y \backslash y_1, y_2]$

This suggests that our first move should be [∃-elim] on line 1, and this yields

[17] When a proof (or partial proof) has provisos, they are shown in a pane that appears in the lower part of the proof's session window. The boundary between that pane and the proof itself can be dragged using the left hand mouse button.

1 :	∃x:a•∀y:a′•q	assumption
2 :	y1∈a , ∀y:a′•q[y1/x]	assumptions
	. . .	
3 :	∀y:a′•∃x:a•q	
4 :	∀y:a′•∃x:a•q	∃−elim 1,2−3

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

Our second move should be [∀-intro] on line 3, and this leads to

1 :	∃x:a•∀y:a′•q	assumption
2 :	y1∈a , ∀y:a′•q[y1/x]	assumptions
3 :	y2∈a′	assumption
	. . .	
4 :	∃x:a•q[y2/y]	
5 :	∀y:a′•∃x:a•q	∀−intro 3−4
6 :	∀y:a′•∃x:a•q	∃−elim 1,2−5

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

We now eliminate the existential quantifier by text-selecting y_1 then double clicking on line 4, yielding

1 :	∃x:a•∀y:a′•q	assumption
2 :	y1∈a , ∀y:a′•q[y1/x]	assumptions
3 :	y2∈a′	assumption
	. . .	
4 :	q[y2,y1/y,x]	
5 :	∃x:a•q[y2/y]	∃−intro 2.1,4
6 :	∀y:a′•∃x:a•q	∀−intro 3−5
7 :	∀y:a′•∃x:a•q	∃−elim 1,2−6

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

Finally we use the second hypothesis on line 2 to show that y_2 has the property we want. We do so by text-selecting y_2, and double-clicking on the second hypothesis on line 2. This very nearly closes the proof

1 :	∃x:a•∀y:a′•q	assumption
2 :	y1∈a , ∀y:a′•q[y1/x]	assumptions
3 :	y2∈a′	assumption
4 :	q[y1,y2/x,y]	∀−elim 2.2,3
	. . .	
5 :	q[y2,y1/y,x]	
6 :	∃x:a•q[y2/y]	∃−intro 2.1,5
7 :	∀y:a′•∃x:a•q	∀−intro 3−6
8 :	∀y:a′•∃x:a•q	∃−elim 1,2−7

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

Here – despite the `AUTOMATCH hypothesis` declaration – a manual invocation of the hypothesis rule is needed to recognise that $q[x, y\backslash y_1, y_2]$ and $q[y, x\backslash y_2, y_1]$ are really the same formula.

1 :	∃x:a•∀y:a′•q	assumption
2 :	y1∈a , ∀y:a′•q[y1/x]	assumptions
3 :	y2∈a′	assumption
4 :	q[y1,y2/x,y]	∀−elim 2.2,3
5 :	∃x:a•q[y2/y]	∃−intro 2.1,4
6 :	∀y:a′•∃x:a•q	∀−intro 3−5
7 :	∀y:a′•∃x:a•q	∃−elim 1,2−6

Providing
x NOTIN a
x NOTIN a′
y NOTIN a
y NOTIN a′
y NOTIN x

It's natural to wonder whether we can dispense with the introduction of the extra names y_1 and y_2 in this proof, and the answer is that we can, as Figure 1 shows. During the discovery of that proof we text-selected the bound variable of each quantifier as we used its elimination or its introduction rule.

Q Are there other ways to discover the same proof?

A There may be – the only way for you to find out is to try it yourself.

Q Are there other proofs?

A Yes: Figure 2.8.1 shows the proof we get if we apply the rules in the order [∀-intro], [∃-elim], [∃-elim], [∀-elim].

Q Can we apply the rules in any old order?

A No.

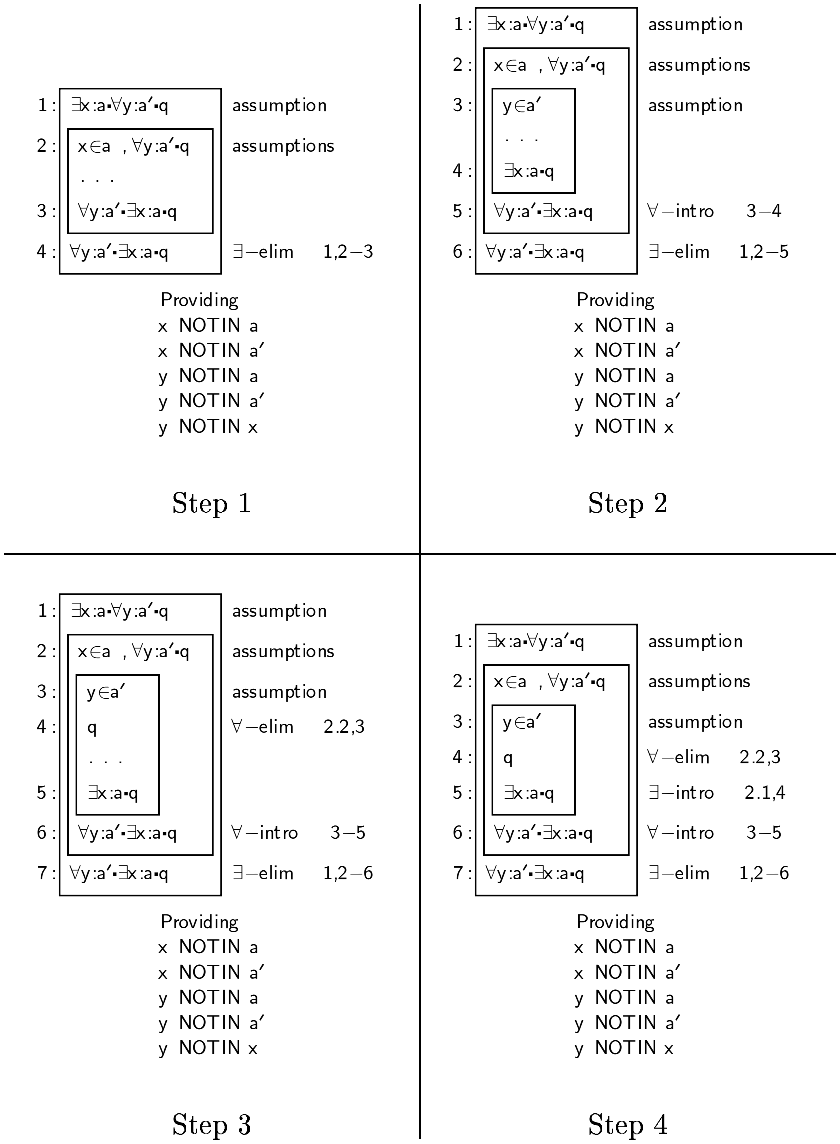


Figure 1: Discovering a proof of $\exists x : a \cdot \forall y : a' \cdot q \vdash \forall y : a' \cdot \exists x : a \cdot q$

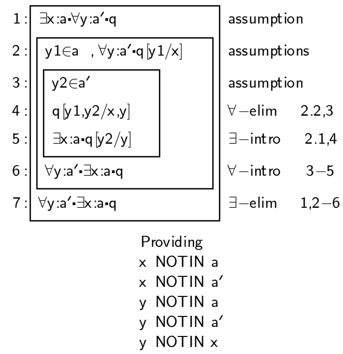
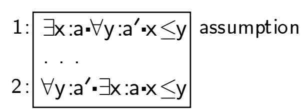


Figure 2: An alternative proof of $\exists x : a \cdot \forall y : a' \cdot q \vdash \forall y : a' \cdot \exists x : a \cdot q$

2.8.2 A proof with changing provisos

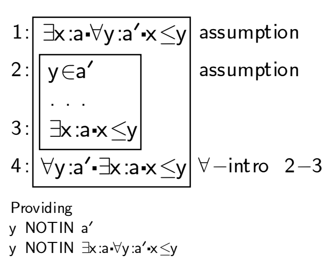
For our final case study we shall see what happens when we try to prove the following conjecture without provisos.



Here we have replaced the abstract proposition letter q with a concrete proposition, namely $x \leq y$, and we have omitted the original provisos.

When we prove a conjecture in Jape we want all sensible substitution instances of the theorem to be valid – because we may be going to use the proven conjecture as a derived proof rule. What the present example proof demonstrate is that whenever Jape applies a rule, it deduces the provisos necessary to make the partial proof shown in the window valid at all substitution instances.

Our proof will follow the same trajectory as the proof in Figure 2.8.1. We first simplify the universal conjunction – text-selecting y before we double click on line 2. This yields



JAPE has added the two provisos

$$y\ NOTIN\ a', \quad y\ NOTIN\ \exists x : a \bullet \forall y : a' \bullet x \leq y$$

If we remind ourselves what the universal-introduction rule was, then we can see where they came from.[18]

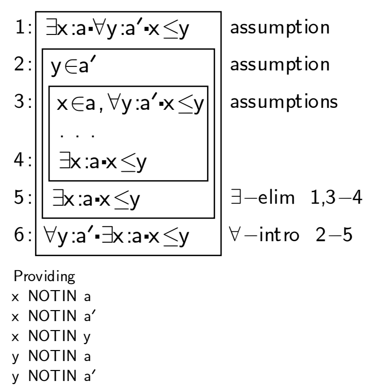
```
RULE    "∀-intro"(OBJECT v) WHERE FRESH v
FROM    v∈s ⊢ q[w\v]
INFER   ∀ w:s • q
```

The instance of the rule used in this case (because we text-selected the y) was

```
WHERE   FRESH y
FROM    y∈a' ⊢ ∃x:a•x≤y,
INFER   ∀ x:a • ∃x:a•x≤y
```

JAPE derived both provisos from the `FRESH v` proviso of the rule – designed to ensure the "arbitrariness" of the element v chosen from s. In the absence of information that x and y are different, the present JAPE doesn't probe the structure of $\exists x : a \bullet \forall y : a' \bullet x \leq y$.

The next step we make is to text-select x, and double-click on line 1, thereby invoking the existential elimination, and yielding



JAPE has, remarkably, transformed the provisos into just those that are necessary to complete the proof. We can explain how it did so by reminding ourselves of the definition of the existential elimination rule, then looking at the precise instance of the rule which JAPE used in this case.

The rule schema is

[18] For the convenience of our readers we have systematically changed the names of the schematic variables used in our statement of the rule.

```
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[x\y] ⊢ r
INFER   r
```

The instance we used in the move was

```
WHERE   FRESH x
AND     x NOTIN ∃ x:a • ∀y:a' • x≤y
AND     x NOTIN ∃ x:a • x≤y
FROM    ∃ x:a • ∀y:a' • x≤y
AND     x∈a, ∀y:a' • x≤y ⊢ ∃ x:a • x≤y
INFER   ∃ x:a • x≤y
```

Now the two `NOTIN` clauses in the provisos of this instance can be simplified to

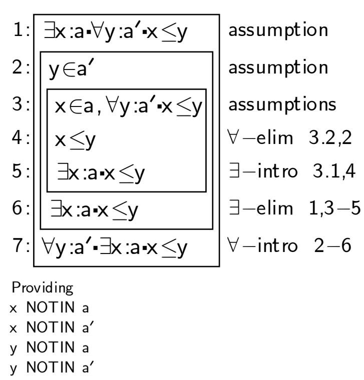
$$x\ NOTIN\ a',\quad x\ NOTIN\ a$$

and the `FRESH` proviso requires that x not be free in any of the hypotheses in scope at the point of application of the rule – and this generates a proviso

$$x\ NOTIN\ y$$

Now JAPE tries to simplify the existing provisos using the newly-generated ones, and this eliminates the now-redundant $y\ NOTIN\ \exists x : a \bullet \forall y : a' \bullet x \leq y$.

The next two moves are universal elimination and existential introduction (in either order) and they lead to



A simpler proof, which would also have discovered the same provisos is one which uses the theorem we proved earlier

1:	$\exists x{:}a \bullet \forall y{:}a' \bullet x \leq y$	assumption	
2:	$\forall y{:}a' \bullet \exists x{:}a \bullet x \leq y$	Theorem $\exists x{:}a \bullet \forall y{:}a' \bullet q \vdash \forall y{:}a' \bullet \exists x{:}a \bullet q$	1

Providing
x NOTIN a
x NOTIN a′
x NOTIN y
y NOTIN a
y NOTIN a′

2.8.3 Proof by Rewriting

A rewrite move can be made by selecting the required equivalence theorem in the conjectures window, tand pressing one of the **Rewrite** buttons in that window. If a text selection has been made, then that selection will be the subject of the rewrite (if possible), otherwise JAPE will search the current goal for a subterm at which the appropriate rule can be applied as a rewrite.

Such moves can often dramatically shorten a proof. For example, compare the following two proofs – the first of which is a proof by rewriting, and the second a conventional proof using the logical inference rules.

1 :	$q \vee \neg p \Longleftrightarrow q \vee \neg p$	Theorem $p \Longleftrightarrow p$	
2 :	$\neg p \vee q \Longleftrightarrow q \vee \neg p$	rewrite Theorem $(p \vee q) \Longleftrightarrow (q \vee p)$	1
3 :	$\neg p \vee q \Longleftrightarrow \neg\neg q \vee \neg p$	rewrite Theorem $\neg\neg p \Longleftrightarrow p$	2
4 :	$\neg p \vee q \Longleftrightarrow (\neg q \Longrightarrow \neg p)$	rewrite Theorem $(p \Longrightarrow q) \Longleftrightarrow (\neg p \vee q)$	3
5 :	$(p \Longrightarrow q) \Longleftrightarrow (\neg q \Longrightarrow \neg p)$	rewrite Theorem $(p \Longrightarrow q) \Longleftrightarrow (\neg p \vee q)$	4

1 :	$p \Longrightarrow q$	assumption	
2 :	$\neg q$	assumption	
3 :	p	assumption	
4 :	q	$\Longrightarrow$−elim	1,3
5 :	false	$\neg$−elim	2,4
6 :	$\neg p$	$\neg$−intro	3−5
7 :	$\neg q \Longrightarrow \neg p$	$\Longrightarrow$−intro	2−6
8 :	$(p \Longrightarrow q) \Longrightarrow \neg q \Longrightarrow \neg p$	$\Longrightarrow$−intro	1−7
9 :	$\neg q \Longrightarrow \neg p$	assumption	
10 :	p	assumption	
11 :	$\neg q$	assumption	
12 :	$\neg p$	$\Longrightarrow$−elim	9,11
13 :	false	$\neg$−elim	12,10
14 :	q	false−elim	11−13
15 :	$p \Longrightarrow q$	$\Longrightarrow$−intro	10−14
16 :	$(\neg q \Longrightarrow \neg p) \Longrightarrow p \Longrightarrow q$	$\Longrightarrow$−intro	9−15
17 :	$(p \Longrightarrow q) \Longleftrightarrow (\neg q \Longrightarrow \neg p)$	$\Longleftrightarrow$−intro	8,16

3 Do you want to know more?

THE following resources are available for those who wish to find out more about JAPE. JAPE is a work-in-progress, so there may be slight inconsistencies between different accounts of it.

- Richard Bornat's note describing a JAPE implementation of a slightly different natural-deduction-style logic is available on the net [1]. This is recommended for those who want to learn more about using JAPE to do natural deduction proofs, and who wish to see more detailed explanations of the way in which some of the rules can be used from JAPE.
- Our note [2] describing how to make your own JAPE logic by taking the reader through several of the example theories we issue with JAPE.
- The directory `/PACK/jape/default/examples/` at your installation contains example theories which you may wish to play with.
- There are JAPE Web sites at Oxford [3] and QMW from which most of the JAPE documentation is accessible, and on which news about JAPE is published from time to time.

  The Unix Jape Companion explains the Unix Jape interface as well as the supporting software you will need if you intend to print or publish proofs made with Jape.

  ```
  http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/JAPE/unixjapeψ.gz
  ```

  You might be interested to read the description of Unix/X JAPE given in one of the following World-Wide Web documents

  ```
  http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/UNIXJAPEDOCHTML/gist.html
  http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/UNIXJAPEDOCHTML/unixjape.html
  ```

  Both are very slightly out of date, but they give a flavour of what it's like to use JAPE, and give reasonably clear descriptions of the purpose of the various windows and panels and buttons which comprise the JAPE user interface. The former describes how to use JAPE to conduct proofs in a variant of predicate logic which differs from J'N'J in some important details. The latter describes how to conduct proofs in a little theory of functional programming.

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: 1.13

Date: 2003/01/20 15:36:02

Latexed: January 20, 2003 at 15:47

Contents