Using MacOS Jape

Richard Bornat, September 1997 (Jape version 3.2)

This note describes the process of running Jape under MacOS, explains the menus and various other logic-independent features of the program.

1. Hardware requirements

Jape needs a 68020/030/040 processor, or a PowerPC – that is, it will run on anything from a Mac II on up. It doesn't need floating-point hardware or software. At the time of writing you need 7MB of memory (real or virtual) to run Jape[1]. Jape will probably run on a machine with only 8MB of real RAM, but very slowly.

At QMW we have found that some elderly Cx processors, with particular configurations of system software, won't run Jape at all unless virtual memory is turned on; but by now that kind of ancient information is surely becoming irrelevant.

[1] This profligate use of memory is embarassing. It's mostly caused by the memory management policies of SML/NJ, the compiler which we have used to implement the proof engine, but some of it is just modern graphical-interface bloat.

2. Proof engines and the graphical interface

At the time of writing, MacOS Jape is two applications: the *proof engine* does the logical work; the *graphical interface*, called MacJape, shows you the results and allows you to control the work of the proof engine.

There is only really one Jape proof engine (naa nana nana, only one Jape proof engine, ...). The engine reads the definition of a logic and is then capable of carrying out proofs in that logic under your direction. It is possible also to *pre-initialise* the engine with a logic definition and save the result as an application. Then there may seem to be several Jape engines. For example, at QMW there are

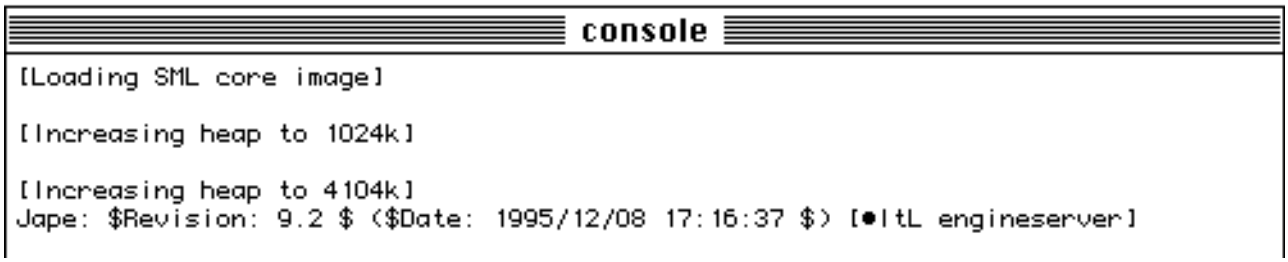
- *jape engine*, which is the raw proof engine, not pre-initialised;
- *ItL engine*, which is pre-initialised with the natural deduction logic taught in the first-year undergraduate 'Introduction to Logic' course;
- *SCS engine*, which is pre-initialised with a version of the intuitionistic fragment of the sequent calculus;
- *MCS engine*, which is pre-initialised with the classical sequent calculus.

But each of these applications is really just the same Jape proof engine, and by using the Close and Open commands from the File menu, you can turn any one into any other.

3. Starting MacOS Jape

You start by double-clicking on a proof engine. The proof engine checks that it is the only running proof engine – to have two would cause much confusion for both operating system and user – and then starts the graphical interface. As soon as the graphical interface is ready, the proof engine goes into deep background mode and disappears. It isn't necessary, or even possible, to interact with it in any way.

Start a proof engine by double-clicking it, or Opening it from the Finder. It puts up a window which looks something like this:



There will be a delay while it loads the code of the proof engine and starts the graphical interface. After a few moments while the menu bar does silly things[1] the console window will disappear, and you will be left with a menu bar that has a representation of a mediæval jester at its right:

If you are running a pre-initialised proof engine, you will probably also see at least one *panel* window, at the top right of the screen. If you are running the raw jape engine, you will have to use the "Open new theory" command from the File menu to open a logic definition; then Jape will put up the panels described there.

[1] The silliness is all because MacOS Jape is configured as two applications. I hope that one day I will find a simpler way to implement Jape under MacOS, while preserving its platform-independent separation of proof engine and interface.

4. Panels and menus

MacOS Jape is a kind of interpreter, following the instructions in a logic definition. Even the user interface details are under the control of the logic definition: it describes the panels and menus which must appear on the screen in addition to Jape's three system menus (File, Edit and Windows), and it may even add extra commands to the File and Edit menus. So it is impossible in this document to say exactly what you will see when you start Jape. It's possible, though, to describe the general features of the interface which are independent of particular logic descriptions.

Panels

All Mac users will be familiar with menus, but panels may be more novel. Jape's panels are 'floating windows': they are always active, they always appear in front of any proof windows you might have, and they disappear completely when you put MacJape into the background. You can close a panel by pressing its close box (and then you can get it back again by choosing its entry from the Windows menu – see below); you can make it grow or shrink by pressing and dragging on the grow box; you can move it around by pressing and dragging on its title bar; clicking on its title bar brings it to the front.

Conjectures panels

Most logic definitions will show you at least one conjectures panel. A conjectures panel contains a list of conjectures for you to prove, and has a number of buttons, which always include 'New…', 'Prove' and 'Show Proof'. In addition there will be one or more buttons which allow you to apply proved conjectures (theorems) as steps in proofs. Figure 1, for example, shows the conjectures panel defined for the QMW presentation of natural deduction. The state of the panel after is shown after three conjectures have been proved: proved conjectures (theorems) are marked with the ® symbol, and the fifth entry in the list is selected.

The main purpose of a conjectures panel is to start proofs and to manipulate theorems. You start a proof by first selecting a conjecture and then pressing the Prove button. You can display the proof of a theorem by selecting its entry and pressing Show Proof. If you come to a point in the proof at which you can complete a step by using an instance of a theorem, you do that by selecting its entry and pressing Apply.

You can add a conjecture to a panel by pressing New…, and Jape will put up a dialogue window (figure 2 overleaf) into which you can type your new conjecture. At the foot of the window there is an array of buttons which you can use to insert the logical operators if you don't know how to type them. When you

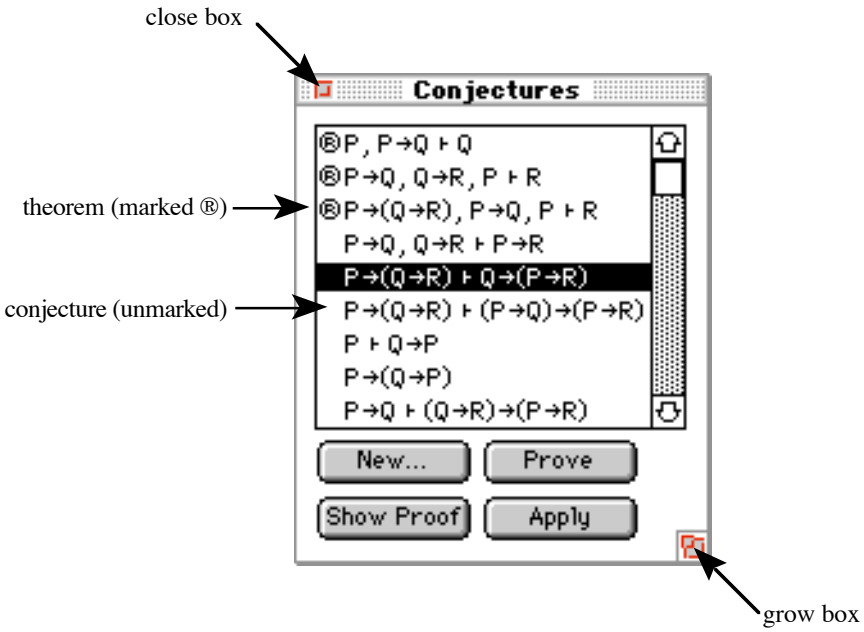


Figure 1: A conjectures panel

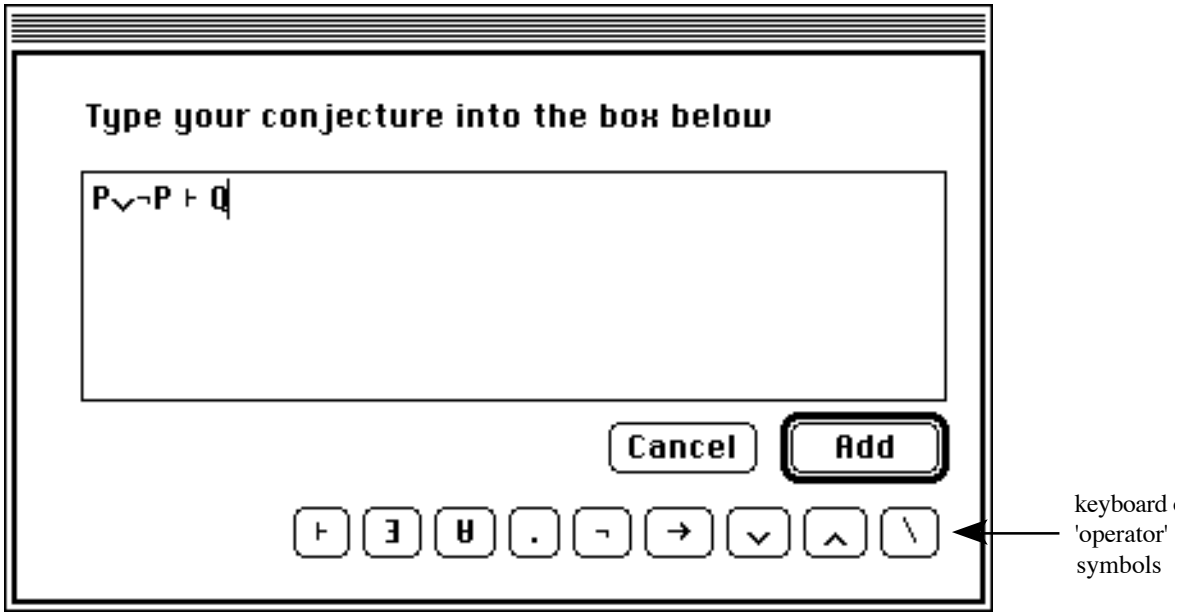


Figure 2:The 'new conjecture' dialog box

have finished your conjecture, pressing Add will put it at the end of the list in the panel. It will automatically be selected, so that you can press Prove immediately if you wish.

Tactic/rule panels

Panels don't have to contain conjectures. It is often useful to have a panel of rules or tactics which can be used to make proof steps, and buttons to use them in different ways. Figure 3 below comes from Bernard Sufrin's encoding of functional-programming reasoning. Each of the buttons corresponds to a different way of applying one of the tactics. It wouldn't be appropriate to go into details here.

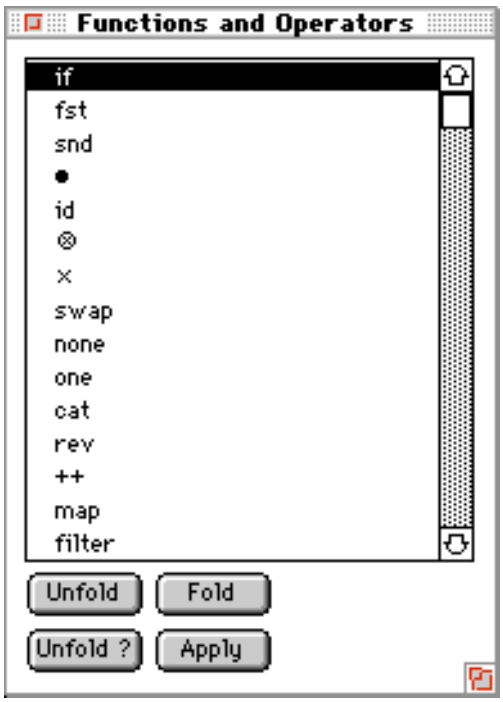


Figure 3: A tactic panel

Figure 4: The File menu

The File menu

The leftmost MacJape menu is always File, and it always contains at least the following commands (see also figure 4 above):

- Open Logic/Proof file is always available, and is used to load or to augment logic descriptions and/or to reload saved collections of proofs;
- Open new theory is always available; it does a Reset to delete the current logic and stored proofs, and then loads in a new collection as you direct;
- Save Proofs and Save Proofs As… are used to save proofs and proofs in progress in a file, and available whenever there are stored proofs, or proofs in progress, that haven't been saved;
- Close Proof Window is available whenever there is a proof window on screen, and has the same effect as pressing in the close box of the topmost proof window;
- Page Setup… and Print are available whenever there is a displayed proof, and can be used to print a picture of that proof;
- Font… is always available, and is used to examine or alter the fonts that Jape uses in menus, in panels and in displaying proofs;
- Text Command is for experts – it allows you to send commands to the proof engine, and it isn't described here;
- Reset is available whenever a logic is loaded, and is used to erase the current logic and any stored proofs, ready to load a new logic;
- Quit is always available, and in normal use you use it to exit gracefully from a session but in abnormal use, if the proof engine loops, or crashes, or otherwise breaks down – which it almost never does – two successive Quits will kill MacJape[1].

[1] The proof engine, if it is still running, should then come out of deep background, and you can use its own Quit command to kill it. This triple-Quit mechanism is rarely necessary, unless you are a Jape developer!

The Edit menu

The Edit menu is always the second menu in the menu bar. It contains at least the following commands (see also figure 5):

- Undo is available whenever the proof in the front proof window has made a step which can be undone;
- Redo is available whenever that same proof has undone a step;
- Cut, Copy and Paste are only available when there is an active dialogue window which includes a box into which you can type some text (for example, the 'new conjectures' dialogue shown in figure 2);
- Copy Proof puts a copy of the front proof into the clipboard, which you can then paste into other documents (I use it to make example illustrations in manuals and other publications);
- Unify is used to unify formulæ which you have selected in a proof (see 'Using the mouse' below);
- Backtrack is a drastic Undo (if you choose a closed (proved) conclusion or sequent, then Backtrack will take you back to the most recent proof step at which it was open);
- Prune removes parts of the proof (if you choose a closed conclusion or sequent, Prune deletes the whole subproof above it, and thereby makes it open again; luckily[1], both Backtrack and Prune can be Undone);
- Hide/Show subproof is used to remove a subproof from the display (but it is still there in the tree), and then can be used to get it back again;
- Expand/Contract detail is used in logics in which a proof may contain steps that aren't normally shown in the display,. as, for example, in Bernard Sufrin's treatment of equational reasoning in functional programming (its use is best explained by looking at the documentation of such a logic).

[1] Luck? It's nothing of the kind! Just good design. But it could be lucky for you, if you are just exploring the effect of menu commands and you don't like the effect you have caused.

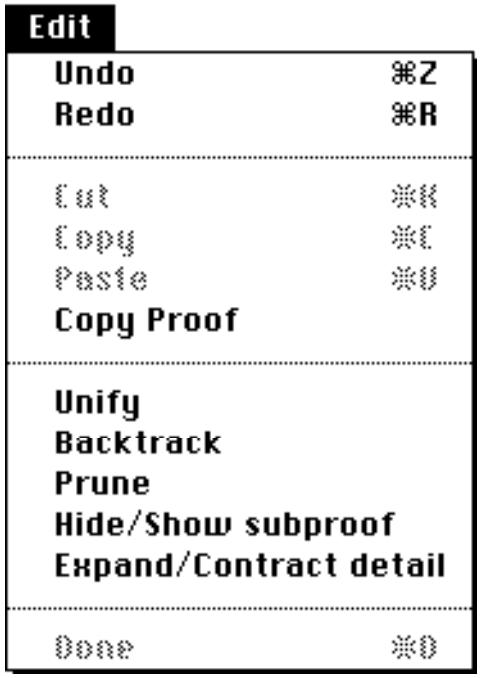


Figure 5: The Edit menu

- Done is available whenever the proof in the front proof window is completed[1][2]; it stores that proof in the engine and closes the proof window.

The Windows menu

The Windows menu allows you to control the visibility of panels, and the relative ordering of proof windows. An example is shown in figure 6 below. Above the line is a list of panels. You can Hide visible panels – which has the same effect as pressing the close box in the panel – or Show those which are hidden. Below the line is a list of proof windows: the front proof window, which is the currently active one, is ticked.

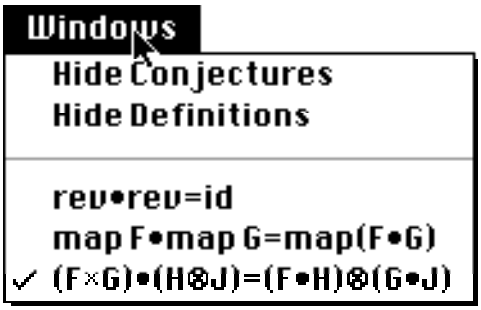


Figure 6: A Windows menu

---

1 A proof of a conjecture is completed only when it has no unproved conclusions/sequents *and* no provisos relating internal names of the proof to names which appear in the conjecture itself.

2 All that happens when a proof is completed is that the Done command becomes available in the Edit menu. People tell me that they expect Jape to play them a fanfare instead. Some hope!

5. Proof windows

When you press Prove in a conjectures panel, Jape shows you a proof window. You can have any number of proof windows at the same time, but only one – the front one, which appears behind any panels which may be floating in its part of the screen – is active. The parts of a proof window are illustrated in figure 7 below.

The proof proper appears in the proof pane – it may be a box, as in this illustration, or a tree. If any provisos arise during the proof they are displayed in the proviso pane. You can alter the boundary between the panes by pressing on the split bar and dragging it upwards or downwards. When a proof becomes large you can scroll it up or down and left or right with the proof scroll bars; similarly you can scroll through a long list of provisos with the proviso scroll bar. You can make the window take up the whole screen by clicking in the zoom box, and you can make it shrink back by clicking again in the same box. You can make the window grow or shrink by pressing and dragging on its grow box.

If there is more than one proof window on the screen, clicking in one that is in the background brings it to the foreground (behind any visible panels).

Clicking in the close box closes the window. If the proof is incomplete, Jape will ask you if you want to abandon it; if it is already complete, it will ask if you want to record it.

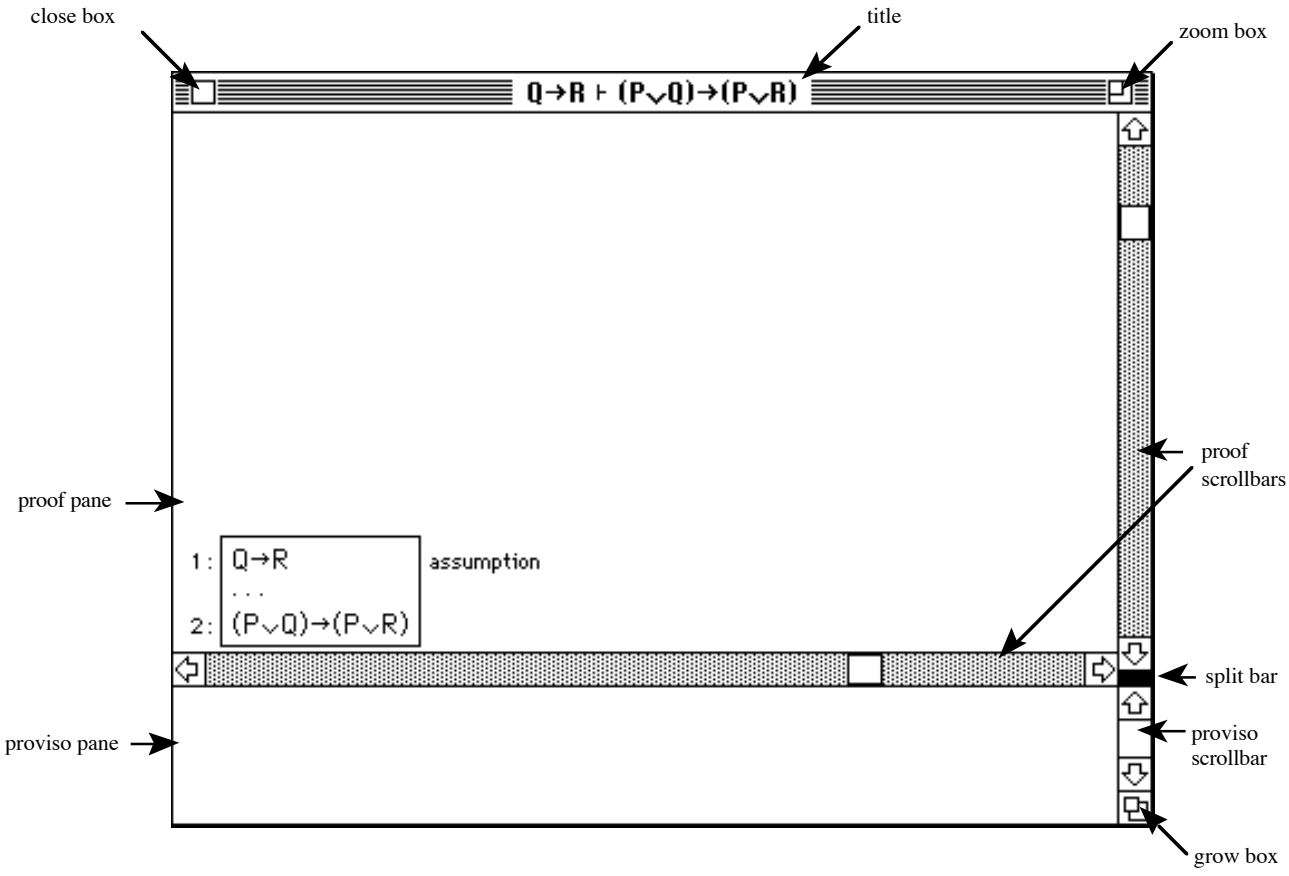


Figure 7: A proof window

6. Use of the mouse

As well as the standard clicks, presses and drags used to manipulate its menus, scroll bars and buttons,, MacJape recognises three different kinds of gesture within the proof pane:

- clicking on a hypothesis or a conclusion *selects* it (and cancels any other selection of the same kind of formula), and Jape highlights the selection by drawing a box round it;
- clicking on a blank part of the proof pane cancels all formula selections in that pane;
- double-clicking on a hypothesis or a conclusion first selects it, cancelling any other selection of the same kind of formula, and then *hits* it, and Jape takes action if the logic description says what to do when that sort of formula is hit;
- option-press[1] and drag over some or all of the characters of a formula *text selects* them (and cancels any other text selections in the window), and then Jape highlights them in the usual way, by showing a background colour or by inverting text and background.
- option-click on a blank part of the proof cancels all text selections in that pane;
- command-option-press[2] and drag over some or all of the characters of a formula adds an additional text selection;
- shift-option-press[3] and drag can be used to stretch or shrink an existing text selection.
- press and drag starts a ‘drag and drop’ gesture: the formula you are over is selected, and can be dragged over other formulæ in the proof, or as a text clipping into other applications. This gesture is mainly useful to deal with the effect of context-splitting (multiplicative) rules.

The meaning of the gestures is best described by example, and examples should be included in the document which describes a logic, but it is possible to illustrate the principles

Formula selection

MacJape shows formula selections by drawing a box round the selected formula, as shown in figure 8 below. As you can see, a side-effect of formula selection in box display is that part of the proof is ‘greyed out’ to show that it isn’t relevant to the hypothesis and/or conclusion that has been selected.

You can select at most one conclusion and one hypothesis at any time, so if you select a second in either category, the first is cancelled. In the examples below, one of each has been selected.

You cancel your formula selections by clicking on a blank part of the proof pane.

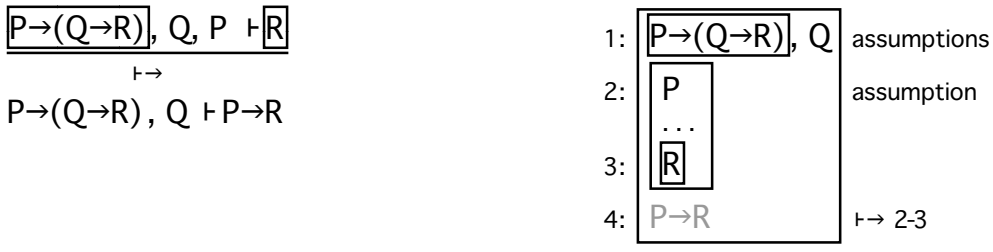


Figure 8: Examples of formula selection

---

1 Pressing on the mouse button while holding the option/alt shift key down.

2 Pressing on the mouse button while holding the command and the option/alt shift keys down. The command key is the one labelled with and/or and option/alt is always next to it.

3 Come on, guess!

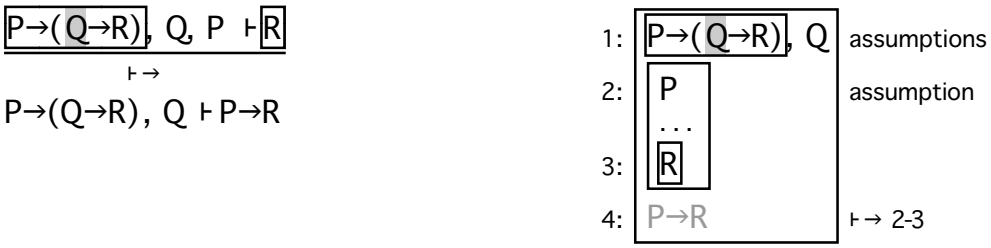


Figure 9a: Examples of primary text selection

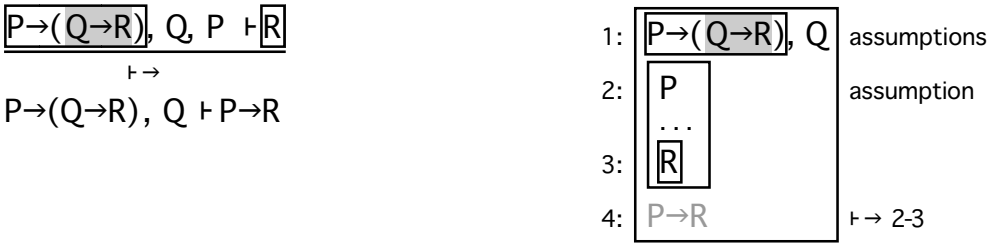


Figure 9b: The same primary text selections, extended by shift-option-press and drag

What formula selection means

When you select a hypothesis and/or a conclusion and then apply a rule or a theorem, Jape tries to apply it in such a way that the formula(e) you selected are involved in the application. For example, if you have a rule which applies to an implication hypothesis, and there is more than one such hypothesis which it could work on, you might select one of them before applying the rule.

If there is more than one open problem sequent – more than one open conclusion in a box display – then at least one formula selection is necessary to point out the sequent which you want to work on.

What hitting means

If the logic description has been set up to allow it, double-clicking a formula can automatically apply a tactic, a rule or a theorem. It might be, for example, that double-clicking an implication formula which appears in the hypotheses of a sequent would automatically invoke some implication-elimination rule or tactic.

If the logic description hasn't been set up to interpret hits on the formula you double-clicked, Jape shows an error message.

A formula doesn't have to be selected before you double-click it. Double-clicking first selects it, then proceeds with the effect of formula hitting, if any.

Text selection

MacJape shows text selections by highlighting the characters you drag over. Although the illustrations in figure 9a above happen to show text selections within formulæ which are also formula-selected, this isn't by any means necessary, and the two forms of selection are independent of each other.

Shift-option-press and drag, in a formula which already contains a text selection, allows you to extend or shrink its boundaries, as illustrated in figure 9b.

Command-option-press and drag allows additional text selections, illustrated in figure 9c overleaf. The selections are completely unordered – Jape won't treat one as 'first' and another as 'second'. You can, of course, have as many simultaneous text selections as you wish.

You cancel all your text selections by option-clicking anywhere in the proof pane. You can cancel just one of your text selections by command-option-clicking within it.

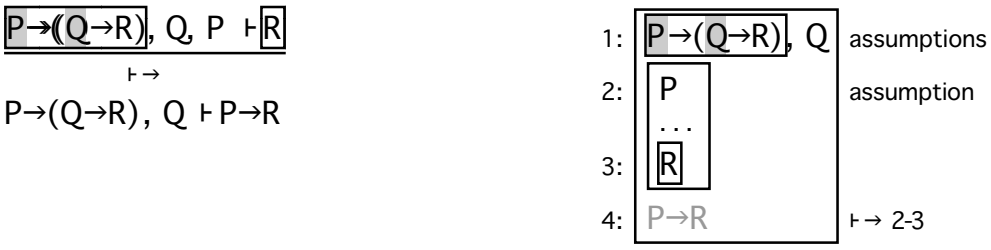


Figure 9c: Multiple text selections with command-option-press and drag

What text selection means

The meaning of text selection is so much under the control of tactics in the logic description that it isn't possible to be very specific. In simple cases, though, a single text selection is often used to provide an 'argument' to the application of a rule, a tactic or a theorem.

Multiple text selections are far more variable. They can't be used to provide multiple arguments, because the selections are unordered. One important use of multiple selection is to indicate the particular instances of a sub-formula which are to be substituted by application of a rule. Another is to indicate several distinct subformulæ which should be unified (see below).

Because there isn't a straightforward interpretation of text selection, use it with care and consult the manual which goes with the logic description you are using. The manual should tell all, if text selection is helpful.

Text selection and the Unify command

Sometimes you make a proof step which introduces unknowns into the display. Unknowns are Jape's way of allowing you to defer a decision until later. Then, sometimes, you notice that the formula which should unify with an unknown is already present in the proof. For example, consider this proof of P∨¬P:

1: ¬(P∨¬P) assumption
...
2: _B∧¬_B
3: ¬¬(P∨¬P) ¬-I 1-2
4: P∨¬P ¬-E 3

The formula which will eventually unify with _B is, in this case, P∨¬P itself[1], The unification can be made to happen later in the proof, as a side-effect of the normal process of rule application, but if you want to short-circuit the process you can.

[1] If this isn't obvious to you, don't worry: I don't think it's obvious to anybody. Indeed I believe that the 'proof' of this classical conjecture in a natural-deduction style is filth. But Jape, because it's a program, just plays by the rules and has no philosophical axes to grind.

You first text select an instance of each of the sub-formulæ that you want to unify – in this case one of _B and one of P∨¬P, using option-press and drag for the first and command-option-press and drag for the second[1]:

1: ¬(P∨¬P) assumption
. . .
2: _B∧¬_B
3: ¬¬(P∨¬P) ¬-I 1-2
4: P∨¬P ¬-E 3

Then Unify will make the necessary adjustment, and the unknowns will disappear:

1: ¬(P∨¬P) assumption
. . .
2: (P∨¬P)∧¬(P∨¬P)
3: ¬¬(P∨¬P) ¬-I 1-2
4: P∨¬P ¬-E 3

Drag and drop

Here is a picture of a partial proof in a multiple-conclusion sequent calculus, just after a multiplicative rule has split both left and right contexts.

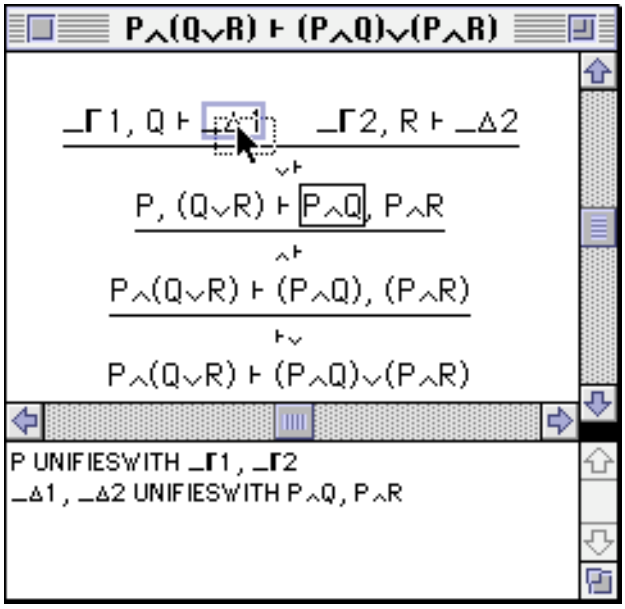
$$\dfrac{\dfrac{\dfrac{\dfrac{\_\Gamma 1,\ Q \vdash \_\Delta 1 \qquad \_\Gamma 2,\ R \vdash \_\Delta 2}{\vee\vdash}}{\mathsf{P},\ (\mathsf{Q}\vee\mathsf{R}) \vdash \mathsf{P}\wedge\mathsf{Q},\ \mathsf{P}\wedge\mathsf{R}}}{\wedge\vdash}}{\mathsf{P}\wedge(\mathsf{Q}\vee\mathsf{R}) \vdash (\mathsf{P}\wedge\mathsf{Q}),\ (\mathsf{P}\wedge\mathsf{R})}}{\vdash\vee}$$

$$\mathsf{P}\wedge(\mathsf{Q}\vee\mathsf{R}) \vdash (\mathsf{P}\wedge\mathsf{Q})\vee(\mathsf{P}\wedge\mathsf{R})$$

P UNIFIESWITH _Γ1, _Γ2
_Δ1, _Δ2 UNIFIESWITH P∧Q, P∧R

[1] I've learnt to be rather lazy, and now I use command-option-press every time.

To resolve the ambiguity it is possible to drag one of the two right-hand side formulæ into _Δ1 and the other into _Δ2. The two possible destinations will light up as the formula passes over them:



If the mouse button is released at the point illustrated, the proof and provisos adjust correspondingly as the formula is ‘dropped into’ the target variable.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\_\Gamma 1,\ Q \vdash \_\Delta 3,\ P\wedge Q}\quad\dfrac{}{\_\Gamma 2,\ R \vdash \_\Delta 2}}{P,\ (Q\vee R) \vdash P\wedge Q,\ P\wedge R}\vee\vdash}{P\wedge(Q\vee R) \vdash (P\wedge Q),\ (P\wedge R)}\wedge\vdash}{P\wedge(Q\vee R) \vdash (P\wedge Q)\vee(P\wedge R)}\vdash\vee$$

P UNIFIESWITH _Γ2, _Γ1
P∧R UNIFIESWITH _Δ2, _Δ3

7. Recording proofs, Loading and saving

The Done command in the Edit menu records a completed proof within the proof engine. The proved conjecture is then marked as a theorem (using the ® mark) in whatever panels it appears. You can recall the proof of a theorem by pressing the Show Proof button on a conjectures panel: at present, all that you are shown is the final state of the proof, and not the stages by which it was built up, so that you can't examine its construction by using Undo/Redo. You can, however, cut out parts of the proof using Prune and repeat them. You can also start a new proof of a theorem by pressing Prove.

Jape can load logic descriptions from text files, several of which are distributed with MacOS Jape. The files are in plain ASCII, in case you are curious to know how it is done. The notation, for those still curious and for those who want to encode a logic for themselves, is described in the manual "Roll your own Jape logic", available from the QMW MacOS Jape site.

Jape can save and reload proofs. When you quit Jape, or when you Close a logic, you are offered the opportunity to save all your recorded proofs and all the proofs in progress in open windows; you can also save your proofs and proofs in progress at any time, using Save or Save As…. You can reload the proofs using Open, provided of course that the logic in which they are proved is already loaded into Jape. The proofs in progress are reinstated, but any steps by which they were built up are lost, so you can't examine them with Undo/Redo.

8. Quitting MacOS Jape

Choose Quit from the File menu: MacJape and the proof engine will each quit, after checking that any proofs which you might have made have been saved in a file.

Since MacJape and the proof engine are separate processes, it is possible that one can crash and the other keep running. If MacJape crashes, the proof engine will come out of the background, and you will see its console window again: just choose Quit from its file menu and it will disappear without fuss. If the proof engine crashes or loops, MacJape will wait for ever for instructions: to quit it you will have to choose Quit twice from its File menu.