Forward reasoning in Jape

Richard Bornat
School of Computing Science, Middlesex University, UK

September 27, 2007

Abstract

1 Introduction

Jape began, back in 1992, as a joint project with Bernard Sufrin to build a proof calculator[1] with a convenient user interface, covering lots of different kinds of user-defined logics with user-configurable automation of the proof calculation process. We didn't know how hard it would be to hit some of those goals. In the end I converted most of the automation mechanism into support for the convenient user interface.

One strand of work which bore fruit was the support for 'box and line' proofs, and this note sets out the design and explains the mechanisms which you can use to do the same thing. It isn't easy to do, and in setting it up I've undermined one of Bernard's original purposes, which was to make Jape easy to configure. Now that I know some of the complexities of supporting proof calculation, I think that perhaps Jape is too general for its own good, and that box and line proofs should be supported by a special natural deduction tool. But I haven't built such a tool, so there follows a description of how to do it in Jape.

2 Tree and box display

Jape's first logic encoding was a single-conclusion intuitionistic sequent calculus, with the rules shown in table 1. This is, as Mark Dawson pointed out in the thesis [1, 2] which inspired us in the early days of Jape, an 'all-introduction' calculus: at each step a rule reduces either a hypothesis (left-side) or a conclusion (right-side) formula at a tip of a tree of sequents, generating new tips or (with axioms hyp and $\bot\vdash$) closing the tip. It's then natural to build tree proofs step by step like the partial attempt in figure 1, working backwards from the target sequent towards the tips of the final proof tree.

2.1 A fake box-and-line display

The proof attempt in figure 1 is doomed – you can't prove P from no hypotheses, so the middle tip won't close – but it is a partial proof tree and it can be rendered in fake box-and-line style as in figure 2(a), using a very simple recursive translation from the tree, starting at its root:

[1] We called it a proof 'editor', but it was always in fact a calculator.

Table 1: The rules of the single-conclusion intuitionistic sequent calculus

$$\dfrac{}{\Gamma, A \vdash A}\ \text{hyp}$$

$$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ {\vdash}\wedge \qquad \dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}\ {\vdash}{\rightarrow} \qquad \dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ {\vdash}\vee(\text{L}) \ \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ {\vdash}\vee(\text{R})$$

$$\dfrac{\Gamma \vdash A \rightarrow \bot}{\Gamma \vdash \neg A}\ {\vdash}\neg \qquad \dfrac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash B \rightarrow A}{\Gamma \vdash A \equiv B}\ {\vdash}{\equiv}$$

$$\dfrac{\Gamma \vdash P(m)}{\Gamma \vdash \forall x.P(x)}\ (\text{fresh } m)\ {\vdash}\forall \qquad \dfrac{\Gamma \vdash P(E)}{\Gamma \vdash \exists x.P(x)}\ {\vdash}\exists$$

$$\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}\ \wedge{\vdash} \qquad \dfrac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C}\ {\rightarrow}{\vdash} \qquad \dfrac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C}\ \vee{\vdash}$$

$$\dfrac{\Gamma, A \rightarrow \bot \vdash A}{\Gamma, \neg A \vdash C}\ \neg{\vdash} \qquad \dfrac{}{\Gamma, \bot \vdash C}\ \bot{\vdash} \qquad \dfrac{\Gamma, A \equiv B \vdash C}{\Gamma, A \rightarrow B, B \rightarrow A \vdash C}\ {\equiv}{\vdash}$$

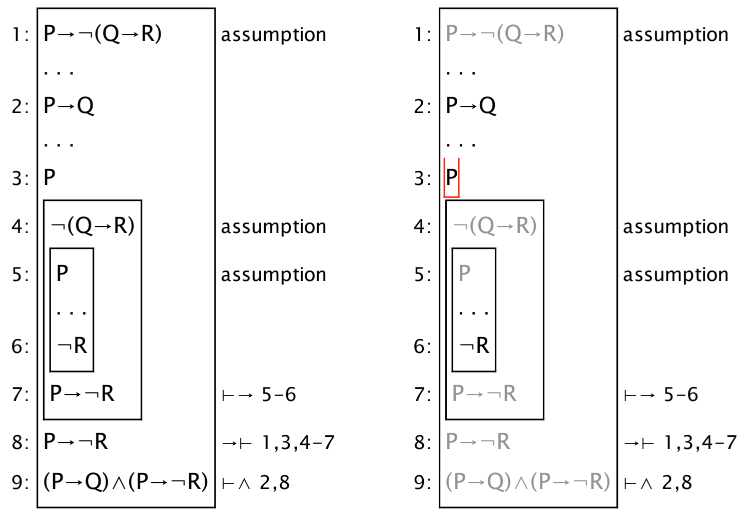
$$\dfrac{\Gamma, P(E) \vdash C}{\Gamma, \forall x.P(x) \vdash C}\ \forall{\vdash} \qquad \dfrac{\Gamma, P(m) \vdash C}{\Gamma, \exists x.P(x) \vdash C}\ (\text{fresh } m)\ \exists{\vdash}$$

$$\dfrac{\Gamma \vdash B \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \text{cut} \qquad \dfrac{\Gamma \vdash C}{\Gamma, A \vdash C}\ \text{weaken}\ {\vdash} \qquad \dfrac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C}\ \text{contract}\ {\vdash}$$

$$\dfrac{P\rightarrow\neg(Q\rightarrow R) \vdash P\rightarrow Q \qquad \dfrac{P \quad \dfrac{\neg(Q\rightarrow R), P \vdash \neg R}{\neg(Q\rightarrow R) \vdash P\rightarrow\neg R}\ {\vdash}{\rightarrow}}{P\rightarrow\neg(Q\rightarrow R) \vdash P\rightarrow\neg R}\ {\rightarrow}{\vdash}}{P\rightarrow\neg(Q\rightarrow R) \vdash (P\rightarrow Q)\wedge(P\rightarrow\neg R)}\ {\vdash}\wedge$$

Figure 1: a sequent tree, produced by backward reasoning

1. At a node which includes a hypothesis (left-side formula) which doesn't appear in its parent:
    - start a new box and write the new hypothesis (or hypotheses) as its first line, labelled "assumption";
    - write the conclusion (right-side formula of the node) on the final line;
    - in between hypothesis and conclusion write the translations of all subtrees, working left to right;
    - label the final line with the name of the rule which generated the subtrees, the line number of the hypothesis formula which was reduced in the rule (if any), and the line numbers of the final line of each subtree representation – or, if a box, the first and last line numbers of the box, separated by a hyphen;
    - if the node is a tip, don't label the final line – instead precede it with a line of dots to show that it is an open conclusion.



```
1: P→¬(Q→R)                assumption
   ...
2: P→Q
   ...
3: P
4:   ¬(Q→R)                assumption
5:     P                   assumption
       ...
6:     ¬R
7:   P→¬R                  ⊢→ 5-6
8: P→¬R                    →⊢ 1,3,4-7
9: (P→Q)∧(P→¬R)            ⊢∧ 2,8
```

(a) simple rendering

```
1: P→¬(Q→R)                assumption
   ...
2: P→Q
   ...
3: P
4:   ¬(Q→R)                assumption
5:     P                   assumption
       ...
6:     ¬R
7:   P→¬R                  ⊢→ 5-6
8: P→¬R                    →⊢ 1,3,4-7
9: (P→Q)∧(P→¬R)            ⊢∧ 2,8
```

(b) box-and-line rules broken

Figure 2: fake box-and-line rendering

2. At a node which doesn't introduce a new hypothesis, just do as above, but without the box and without the hypothesis line.

The root of the tree in figure 1, for example, is represented by the box from lines 1 to 9 of figure 2(a); it generated two subtrees by $\vdash\wedge$, whose conclusion lines are written on lines 2 and 8. Line 2 represents the left tip of the tree, which contains no new hypotheses. Lines 3 to 8 represent the right-hand side of the tree. The first step is $\rightarrow\vdash$, which reduced the hypothesis on line 1 and generated line 3 and the box from lines 4 to 7 ... and so on.

Figure 2(a) looks like a box-and-line partial proof, but it isn't, and Jape will show you that it isn't if you click on one of the open conclusions on lines 2, 3 and 6. Figure 2(b) shows the result of clicking on line 3, which corresponds to the middle tip of the tree. $P \rightarrow \neg(Q \rightarrow R)$ on line 1 is greyed out, because it's a hypothesis which doesn't appear at that tip, and a greyed-out assumption can't be used. Line 2 isn't greyed out, but it's a conclusion line and can't be used as an assumption either. But in a real box-and-line proof you can appeal to any preceding line (which isn't enclosed in a box that you aren't also in) to make a step – which means all assumption lines and even conclusion lines. It's clear that figure 2(b) is not a real box-and-line proof display.

Even fake box-and-line displays can be useful, though. It's not evident on this small example, but trees have some considerable disadvantages resulting from repetition of hypothesis and conclusion formulae. In figure 1, for example, the hypothesis $P \rightarrow \neg(Q \rightarrow R)$ appears three times, while in figure 2(a) it appears once, on line 1. In figure 2(a) each conclusion appears only once, apart from $P \rightarrow \neg R$ on lines 7 and 8. If there are a lot of large hypothesis formulae, the tree can get unmanageably wide on the page. The lack of repetition in the fake box-and-line display can be a great advantage, and its almost one-dimensional structure is much easier to traverse with the eye. Fake box-and-line displays are very popular with Jape users for just these reasons.

2.2 Display style configuration

The variable `displaystyle` governs the tree/box display decision. Its values are `tree` and `box`, and for obvious historical reasons the default is `tree`. The logic encoding can set the session default for

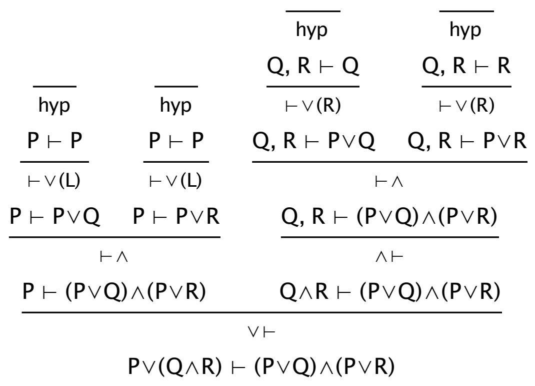


Figure 3: A completed proof

new proof windows with an INITIALISE, and the user can alter it on a proof-by-proof basis. SCS.JT, for example, includes commands in its EDIT menu to alter the setting:

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

3 Hiding identity lines

Figure 3 shows a completed proof in the calculus of table 1, and figure 4(a) its fake box-and-line rendering. In box view, every proof step of the tree is reproduced, and in particular all four uses of the hyp axiom are shown. But now these look like mere indirections: line 6, for example, is proved by $\vdash\vee$(L) from line 5, which is proved by hyp from line 2, and apart from their labelling, lines 2 and 5 are identical. It simplifies the proof display to cut out the hyp-indirection, and to show figure 4(b), in which all the hyp lines have been eliminated. Line 6 has become line 4, which points straight to line 2.

This is Jape's default rendering of fake box-and-line proofs. It isn't the same logic as the sequent logic, but it's possible from it to work out what the tree must be, behind the scenes, so it's not a dishonest rendering, and it's more compact so it's more useful.

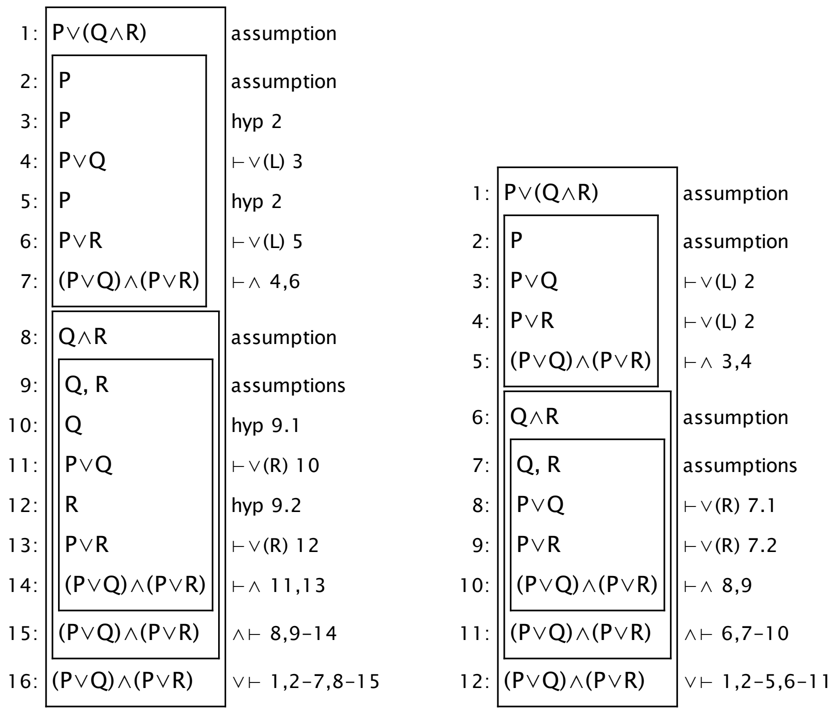
3.1 Identity line configuration

If a rule is of the right kind – a conclusion formula is eliminated by matching with an identical formula from the hypotheses, with no antecedent subproofs – then it is a candidate for identity rule elimination. Declare the rule to be an identity rule – SCS.JT, for example includes a line

```
STRUCTURERULE IDENTITY hyp
```

– and set the variable `hidehyp`[2] to `true` (its default value).

[2] Apology for the historical name: clearly it ought to be `hideidentitysteps` or something.



(a) simple rendering (b) identity lines hidden

Figure 4: Fakery intensifies

4 Forward steps from hypotheses

Table 2 shows the rules of a natural deduction calculus. It is possible to construct proofs by reasoning backward in this calculus, but it's awkward and it certainly isn't the way that provers want to use it. They prefer to construct proofs by reasoning forward from the premises as often as they can, and Jape can accommodate them. Figure 5(a), for example, shows a forward step from E and $E \to F$ which deduces F. The next step would be to deduce G from F and $F \to G$ to conclude G and close the proof.

Figures 5(b) and 5(c) show what has really happened: a `cut` step has introduced an additional hypothesis which can be used in the proof of G. The box display mechanism by default hides cut steps, conflating the left subtree's conclusion (line 2 of figure 5(b)) with the cut hypothesis (line 3) and the right subtree's conclusion (line 4) with the conclusion of the step (line 5), hiding entirely the box and the cut step justification (line 5). The result is a forward step that looks like a forward step, and can be used like a forward step.

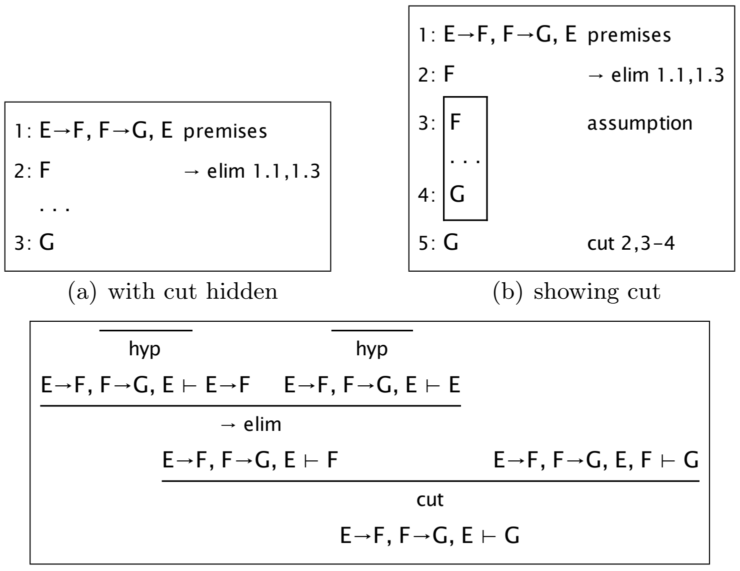
4.1 Configuration

This mechanism needs support from the prover. In `ItL.jt` there is machinery to do the job. It's first necessary to distinguish between proof steps which can be applied forwards (elim steps, on the whole) and those which cannot (intro steps, on the whole) so, for example, the Rules menu contains the entries[3]

[3] The actual encoding uses -E and -I rather than elim and intro, but is otherwise as quoted here.

Table 2: Natural deduction rules

$\dfrac{}{\Gamma, A \vdash A}$ hyp			
$\dfrac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}$ $\to$ elim	$\dfrac{\Gamma \vdash A \land B}{\Gamma \vdash A}$ $\land$ elim(L)	$\dfrac{\Gamma \vdash A \land B}{\Gamma \vdash B}$ $\land$ elim(R)	$\dfrac{\Gamma \vdash B \quad \Gamma \vdash \neg B}{\Gamma \vdash \bot}$ $\neg$ elim
$\dfrac{\Gamma \vdash A \lor B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$ $\lor$ elim		$\dfrac{\Gamma \vdash \forall x.P(x) \quad \Gamma \vdash \text{actual}\, i}{\Gamma \vdash P(i)}$ $\forall$ elim	
$\dfrac{\Gamma \vdash \exists x.P(x) \quad \Gamma, \text{actual}\, i, P(i) \vdash C}{\Gamma \vdash C}$ (FRESH i, i NOTIN $\exists x.P(x)$) $\exists$ elim			
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$ $\to$ intro	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \land B}$ $\to$ intro	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \lor B}$ $\lor$ intro(L)	$\dfrac{\Gamma \vdash B}{\Gamma \vdash A \lor B}$ $\lor$ intro(R)
$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$ $\neg$ intro	$\dfrac{\Gamma, \text{actual}\, i \vdash P(i)}{\Gamma \vdash \forall x.P(x)}$ (FRESH i) $\forall$ intro	$\dfrac{\Gamma \vdash P(i) \quad \text{actual}\, i}{\Gamma \vdash \exists x.P(x)}$ $\exists$ intro	
$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$ contra (classical)	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$ contra (constructive)	$\dfrac{}{\Gamma \vdash \top}$ truth	



```
1: E→F, F→G, E  premises
2: F                    → elim 1.1,1.3
   . . .
3: G
```

(a) with cut hidden

```
1: E→F, F→G, E  premises
2: F                → elim 1.1,1.3
3: │F│              assumption
   │. . .│
4: │G│
5: G                cut 2,3-4
```

(b) showing cut

$$\dfrac{\dfrac{\dfrac{}{E\to F, F\to G, E \vdash E\to F}\,\text{hyp} \quad \dfrac{}{E\to F, F\to G, E \vdash E}\,\text{hyp}}{E\to F, F\to G, E \vdash F}\,\to\text{elim} \qquad E\to F, F\to G, E, F \vdash G}{E\to F, F\to G, E \vdash G}\,\text{cut}$$

(c) the tree

Figure 5: Three views of a forward step

```
    ENTRY "∧ intro"
    ...
    ENTRY "∧ elim(L)"  IS ForwardOrBackward ForwardCut 0 "∧ elim(L)"
    ENTRY "∧ elim(R)"  IS ForwardOrBackward ForwardCut 0 "∧ elim(R)"
```

The first simply labels the ∧ intro rule, to be used backward at some conclusion (Jape will complain if it can't work out which conclusion to use), but the second introduces a tactic which first works out whether the rule is being used forward or backward. That tactic is

```
TACTIC ForwardOrBackward (Forward, n, rule) IS
  WHEN (LETHYP _P
          (ALT (Forward n rule)
               (WHEN
                  (LETARGSEL _Q
                     (Fail (rule is not applicable to assumption ' _P '
                                    with argument ' _Q ')))
                    (Fail (rule is not applicable to assumption ' _P ')))))
       (ALT
          (WITHSELECTIONS rule)
          (WHEN
             (LETARGSEL _P
                (Fail (rule is not applicable with argument ' _P ')))
             (Fail (rule is not applicable))))
```

Essentially, if the prover selects a hypothesis formula (`LETHYP _P`) then what you get is `Forward n rule`, which in the case of the ∧ elim(L) entry turns out to be `ForwardCut 0 "∧ elim(L)"`. If you don't select a hypothesis, you get `WITHSELECTIONS rule`, which in the same case turns out to be `WITHSELECTIONS "∧ elim(L)"`. If either of those tactics fails, then an error message is generated, which needn't detain us.

The `ForwardCut` and `ForwardUncut` tactics do the magic.

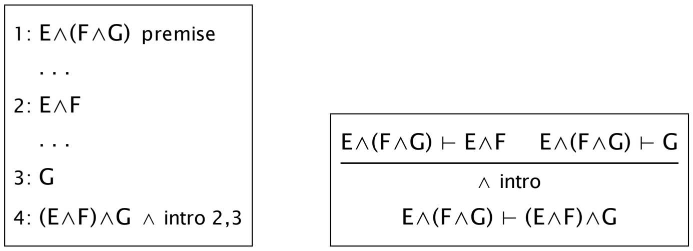
```
TACTIC ForwardCut (n,rule)
    SEQ cut (ForwardUncut n rule)

TACTIC ForwardUncut (n,rule)
    (LETGOALPATH G (WITHARGSEL rule)
                   (GOALPATH (SUBGOAL G n))  (WITHHYPSEL hyp)
                   (GOALPATH G) NEXTGOAL)
```

`ForwardCut` inserts the cut, and then `ForwardUncut` applies the rule. Then it goes to the nth subgoal of the rule and closes it with the selected hypothesis. The rest of the tactic has to do with navigating the tree when `autoselect` is true and can safely be ignored.

5 Why isn't it enough to hide cuts?

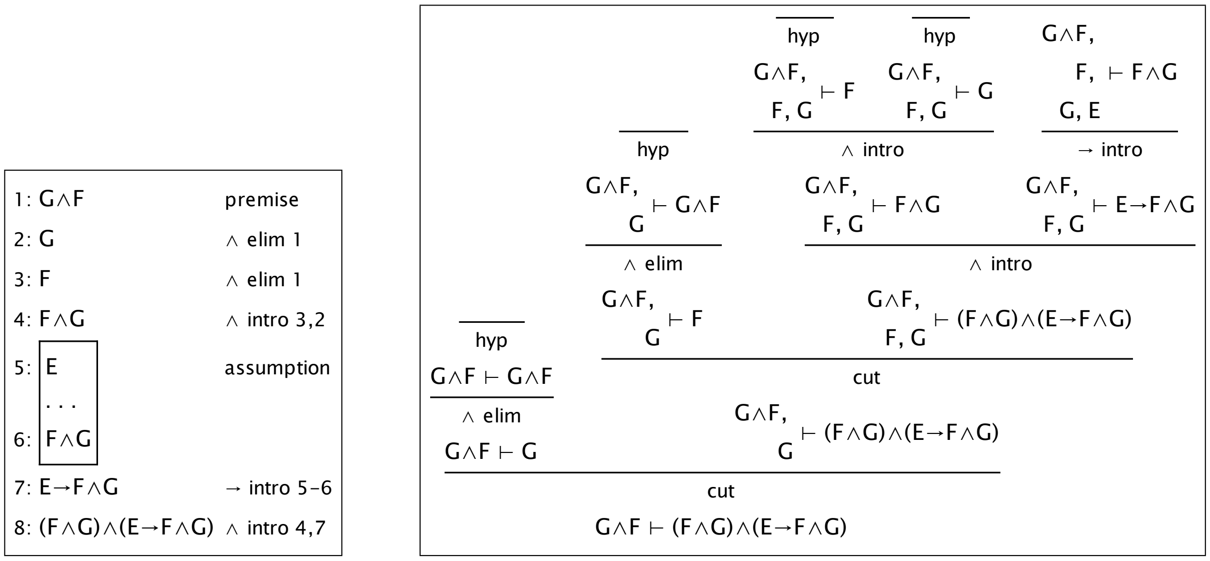
The mechanism of section 4.1 looks good, and in figure 5 it is good, but in general it isn't good enough. There are two problems:

- bifurcated proofs;
- and conclusion lines.



(a) in box form (b) in tree form

Figure 6: Bifurcated proofs don't allow true forward steps



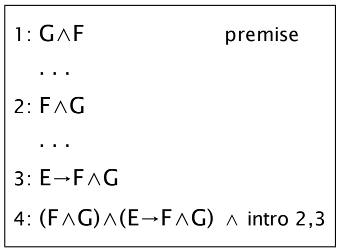
(a) in box form (b) in tree form

Figure 7: Conclusion lines aren't hypotheses

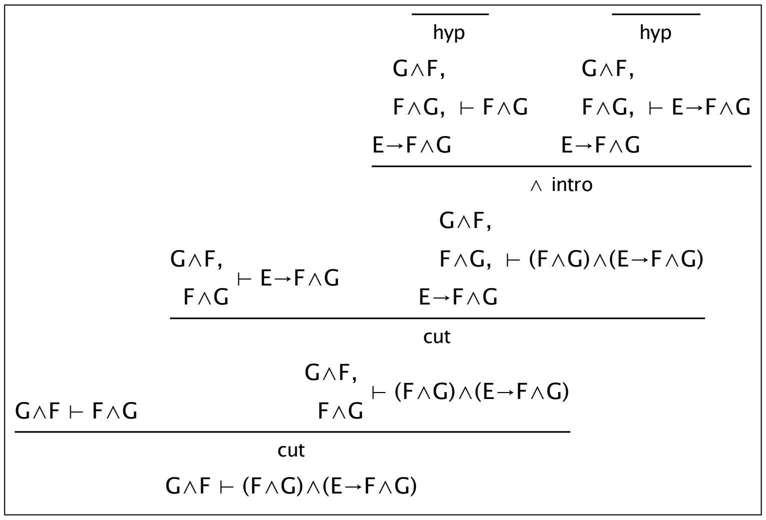
Consider, for example, figure 6, produced by a single application of ∧ intro. In the box display it appears that a forward step of ∧ elim(L) will extract $F \wedge G$, and then another couple of ∧ elims will extract F for line 2 and G for line 3. But from the tree it's clear that this is impossible: we have to apply a cut at one or other tip, and so we have to extract $F \wedge G$ twice, once in each subproof. Provers are driven to plan to complete forward steps before backward steps, a burden they ought not to have to bear. Hiding `cut`s alone doesn't do the job.

Just as bad is the fact that conclusion lines aren't usable as hypotheses in proving subsequent conclusions. It's made worse by the fact that lines introduced by forward steps look just like conclusion lines but can be used as hypotheses. In figure 7 a careful prover has made two forward steps first of all, introducing hypotheses G and F visible throughout the proof, as can be seen in the two cut steps in figure 7(b). Then the left tip ($G \wedge F, F, G \vdash F \wedge G$) has been closed with ∧ intro and a couple of hyps (hidden, of course, in the box display). It looks in the box display as if you could close the proof by hyp: line 6 can see line 4, and they are identical. But it's clear from the tree that you can't: there isn't a hypothesis $F \wedge G$ anywhere in the tree.

We have to do something more drastic, and drastic it is.



(a) as box



(b) as tree

Figure 8: Bifurcation conquered (preparation)

6 True forward steps with `CUTIN`

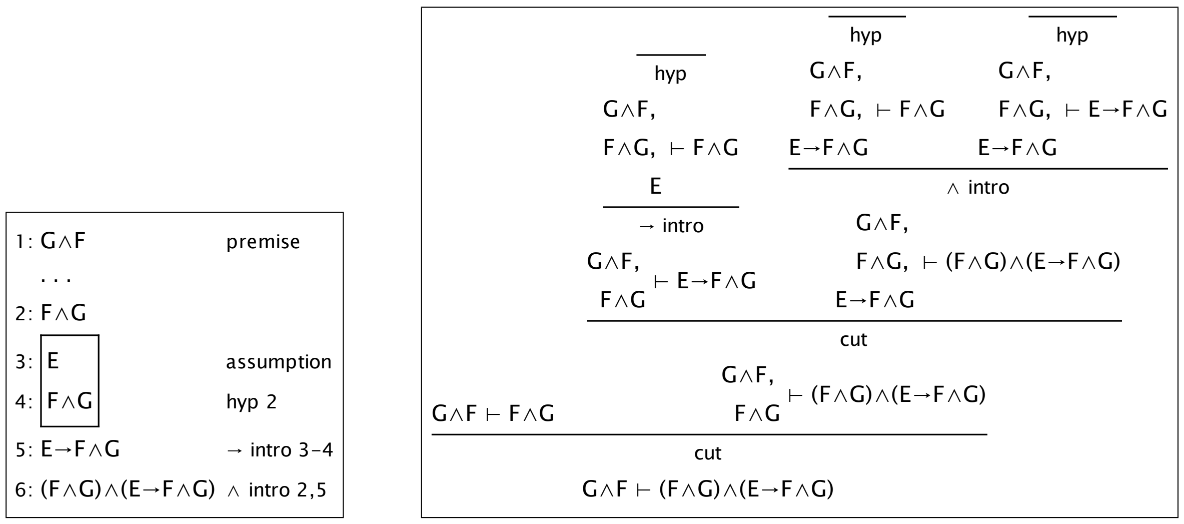
An alert prover can use the mechanism of section 4.1 most effectively by applying forward steps early, getting the cuts in low down in the tree, so that the extra hypothesis formulae they introduce are available to an entire tree. The same applies to forward steps from hypotheses inside a box: do them early, so that they are available to the entire box/subtree. But as you have seen, that isn't enough when a proof bifurcates. We need also to make conclusion lines into hypothesis lines. A single mechanism solves both problems: the built-in tactic `CUTIN`, used for example in the `I2L.jt` encoding.

Look first at what happens when a proof bifurcates, as in figure 8. The box display looks exactly as it would if the proof had bifurcated, but you can see from the tree that it hasn't – or at least the bifurcation is off to the side, and closed off by `hyp`. The two conclusion lines have been cut in as hypotheses, and then the ∧ intro step simply appeals to those hypotheses. This time $F \wedge G$ *is* a hypothesis in the proof of $E \rightarrow F \wedge G$, and as soon as you apply → intro and introduce an open conclusion $F \wedge G$ the earlier line is appealed to, as shown in figure 9.[4]

In accommodating Jape to a much more aggressive use of `cut`, I had to make some changes to the interpretation of formula selection. For some time Jape has shown conclusion-formula selections in box style with an upwards-opening red selection mark, as in figure 2(b), for example, and hypothesis selections with a downwards-opening selection mark. But line 2 in figure 8(a) stands for both an open conclusion at the left-hand subtree of a cut step – that is, it has to be proved as a conclusion – and a hypothesis in the entire right-hand subtree – that is, it can be appealed to as an antecedent in proving subsequent lines. It's possible to select it in two ways, as illustrated in figure 10: for a conclusion selection click in the top half of the formula, for a hypothesis selection click in the bottom half, and in either case the closed end of the selection is dotted to show that it can be selected in the other direction.

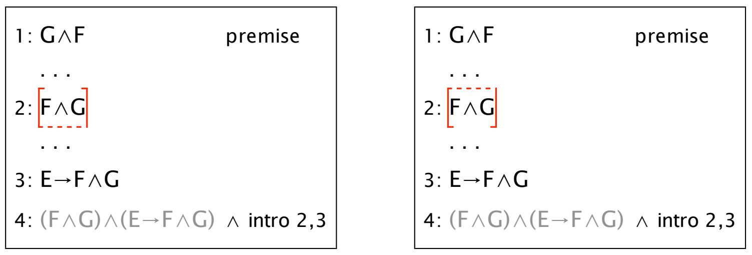
The other change is that it is now possible to select more than one hypothesis formula at a time, and that it is possible to select a hypothesis formula without selecting a conclusion, even if it is

[4] Note that the `hyp` step on line 4 can't be hidden: it isn't simply an indirection, it's an essential part of the box.



(a) as box (b) as tree

Figure 9: Bifurcation conquered (use)



(a) as a conclusion (b) as a hypothesis

Figure 10: Two ways of selecting a cut formula

visible from several open conclusions (set `seektipselection` to `false`: Jape selects the lowest point in the tree at which that hypothesis occurs, which will almost always be a developed node, not a tip – so only do it if you always interpret the hypothesis-only selection as mandating a forward step).

6.1 What `CUTIN` does

Each selection (hypothesis and/or conclusion) corresponds to a unique point in the tree. `CUTIN t` goes down the tree looking for the *lowest* node which contains *exactly* the same hypotheses as the selection, inserts a `cut` node at that point with the previous tree as its right-hand side, augmenting each node in that tree to add the new cut hypothesis and adding `NOTIN` provisos to make sure that it doesn't violate any `FRESH` provisos, runs tactic `t` on the left subtree of the `cut`, and then returns to the original position.

The effect is to add a new hypothesis to the tree, visible to every conclusion that could see the original hypotheses. There's some very nifty work which ensures that path expressions which are used to indicate a tree position still work even once the tree has been mangled with a `cut`. There is a lot of checking to make sure you can do it soundly (`autoAdditiveLeft` must be `true`, which means the hypotheses are a bag and you don't worry about getting extra ones, and there must be exactly one cut rule).

It looks like extreme violence to the tree, which it is, and you might wonder as I did whether it would be easier if I used a box-and-line structure from the first. When I thought hard about it I decided that it would be about as difficult to make a forward step in such a case as it is with the tree, and there is a lot of investment in the tree in Jape as it actually exists. I can't imagine going beyond this hack, though ...

6.2 Configuration: forward steps with `CUTIN`

The `I2L.jt` encoding is complicated by error reporting, but in principle it uses the same technique as section 4.1, but using `CUTIN` rather than `cut` (see `I2L_menus.j`). It works better than the earlier technique, because every line that can see the hypothesis you are reducing gets to see the result of the step too. It is possible to do silly things, but if you start with a situation that obeys the box-and-line rules, then the situation after the forward step also obeys the rules.

6.3 Configuration: backward steps with `CUTIN`

Every time you produce a new conclusion with a backward step, whether it's a bifurcating step or not, you want to make it visible to open conclusion lines below it in the proof – only the ones that can validly see it, of course. There are a couple of tactics in `forwardstep.j` that do the job using `CUTIN`.

First, the badly-named `trueforward` (which is a macro rather than a tactic, and I've forgotten why):

```
MACRO trueforward(tac) IS
    LETGOAL _A
        (CUTIN (LETGOAL _B (UNIFY _A _B) tac))
        (ANY (MATCH hyp))
```

`trueforward t` records the conclusion of the node it's applied to, which has to be a tip. Then it `CUTIN`s a new hypothesis, which it makes the same as the original conclusion, runs tactic `t`, and when that finishes, goes back home and closes the original node by `hyp`.[5] The effect is that `t` is run much lower down the tree, and if each conclusion is treated with `fstep`...

The `fstep` tactic just makes a hypothesis out of the conclusion of the node it's applied to. In effect, that introduces a conclusion which is produced in the left subtree of a cut as a hypothesis in the main spine of the tree.

```
TACTIC fstep IS
    ALT (ANY (MATCH hyp)) (trueforward SKIP)
```

– either find a hypothesis which matches by identity, or use `trueforward` to insert it as a new hypothesis. Very useful, used all over the place in `I2L.jt`, and even more subtly in `hoare.jt`.

7 Summary

If your encoding applies `CUTIN (ForwardCut ...)` to forward steps which don't introduce a box (in natural deduction everything except $\vee$ elim and $\exists$ elim) and `fstep` or `trueforward` every time it introduces a conclusion, you get proof displays which obey the box-and-line rules.

A prover can mess it up by applying rules directly with File:Text Command. If you want to fix that, you probably have to make a version of Jape which uses a box-and-line structure rather than a tree in the background. Feel free!

References

[1] Mark Dawson. *A Generic Logic Environment.* Ph.D. thesis, Imperial College, University of London University, London, UK, 1990.

[2] Mark Dawson. A Generic Logic Environment. In Andrei Voronkov, editor, *LPAR*, volume 624 of *Lecture Notes in Computer Science*, pages 466–468. Springer, 1992.

[5] I ought to make the identity rule it uses a parameter.