

5.5 The order of insert tactics

ProofWeb has one unfortunate restriction, that is caused by the way it is implemented on top of the Coq system. Sometimes, when ‘insert’ing a line, later that line is not visible in a place where according to the (incomplete) proof it *should* be visible. This is caused by the fact that the line that was inserted only will be visible in the part of the proof that corresponds to the dots where it was inserted.

Here is a small example. Suppose we have a proof:

1	A
	\dots
2	D

and we want to insert lines B and C in between the A and D . Then we can do:

`insert H2 (C).`

giving:

1	A
	\dots
2	H2: C
	\dots
3	D

and subsequently:

`insert H1 (B).`

giving:

1	A
	\dots
2	H1: B
	\dots
3	H2: C
	\dots
4	D

However, if we do this, in the proof the line labeled H1 will not be available after the C . It will only be visible in the part of the proof that it was inserted in, which is the part between A and C . So when working on the part of the proof that proves D from C , one will not be able to refer to the B line. This probably will be surprising by that time.

The solution to this problem is to have the `insert` lines in the same order as their corresponding lines occur in the final proof. If one reorders the ‘insert’:

```
insert H1 (B).
insert H2 (C).
```

there will not be a problem anymore.

Therefore, when using `insert` tactics in ProofWeb proofs, the rule of thumb should be:

Put the `insert` lines in the script in the order in which the inserted lines will appear in the proof.

This finishes the chapter on Fitch style ‘box’ proofs in ProofWeb.

6 ProofWeb versus Coq

In ProofWeb one really is using the Coq system. One is processing a Coq script without any change to the Coq syntax. However, the ProofWeb tactics are not the ones that a Coq user normally would use to prove something. The Coq tactics are much more efficient, but less closely related to the natural deduction rules of first order logic. Also the ProofWeb formula syntax deviates a little bit from the way that a Coq user normally would write formulas. We will now briefly indicate what are the differences between using Coq in the ProofWeb style and using it in the Coq style.

6.1 Formula syntax

In Coq one generally does not use `all` and `exi` for the quantifiers, but instead one uses `forall` and `exists`. The reason that ProofWeb defines its own quantifiers is to get the binding strength the way it is in most logic textbooks. Also in ProofWeb one is working in an *unsorted* logic in which all variables range over the same domain, while in Coq variables have a *type*.

So in ProofWeb one writes

$$\text{all } x, P(x)$$

while a Coq user would write

$$\text{forall } x : D, P(x)$$

(To make things subtle, in Coq the typing of the variable can be omitted if the system can deduce the type from the way it is being used.) Similarly in ProofWeb one uses

$$\text{exi } x, P(x)$$

while a Coq user would write

$$\text{exists } x : D, P(x)$$

These are the only differences between ProofWeb formula syntax and the customary Coq formula syntax.

The ProofWeb tactics have been designed to also work with the ‘usual’ Coq quantifiers. If you prefer to have multiple sorts and weakly binding quantifiers in your formula syntax, it still is perfectly possible to use ProofWeb.