How to type-set Fitch natural deductions using GNU troff, pic and eqn

Sigrid Lundberg
sigfrid (at) sigfrid-lundberg.se

Fitch is a notation for natural deduction (Pelletier and Hazen, 2024), and `troff` is a software system for type-setting using Unix™ and related operating systems (Ossanna and Kernighan, 1994). Brian W. Kernighan was one of the creators of Unix and the C programming language. pic is a system for typesetting graphs, also created by Kernighan (1982). GROFF AKA GNU troff is the implementation I am using (FSF, 1990). There are other competitors, but this is the version I use.

The Fitch notations has got its name after its inventor, Fredric Fitch. This notation seems to be a de facto standard: It is used in all the text books I have been able to find electronically, and seems to be taught at logics courses in mathematics as well as philosophy. I wrote this note while learning Fitch; I used the writing was a method for learning. My intention is to demonstrate how to format natural deduction on this platform. I cannot teach you how to format scientific text, neither can I give an introduction to natural deduction.

Table 1. Unicode characters for logical signs and operators. On some operating systems you can type them by pressing `ctrl-shift-u` and then the four character code (following u+). The Groff name is usually better to use than the Unicode character, but I tend to use the latter.

Unicode	**Character**	**Groff name**
u+00AC	¬	\[no]
u+2227	∧	\[AN]
u+2228	∨	\[OR]
u+2200	∀	\[fa]
u+2203	∃	\[te]
u+2192	→	\[->]
u+2194	↔	\[<>]
u+22A5	⊥	\[pp]
u+22A2		
u+2261	≡	\[==]
u+25A1	□	\[sq]
u+25C7		
u+2234	∴	\[tf]
U+2208	∈	\[mo]
U+2209	∉	\[nm]

References

FSF, Free Software Foundation, *Groff* (1990).

Kernighan, Brian W., "PIC — A language for typesetting graphics," *Software: Practice and Experience* **12** (1982).

1	$A \vee B$	
2	$\neg A$	
3	A	
4	$\perp$	$\perp$ Intro: 3,2
5	B	$\perp$ Elim: 4
6	B	
7	B	Reit: 6
8	B	$\vee$ Elim: 6-7,3-5,1

Figure 1. Proof that $A \vee B, \neg A \therefore B$. The line numbering is in the left-most margin. Then there is a vertical line, as long as the proof. The step 1-2 in the proof is where the premises lives. The horisontal line after step 2 is usually referred to as the *fitch line*. The two groups, 3–5 and 3–6 are sub-proofs, with their own premisses, vertical lines and fitch lines

Ossanna, Joseph F. and Kernighan, Brian W., “Troff Userâs Manual,” C*omputing Science Technical Report* **54** (1994).

Pelletier, Francis Jeffry and Hazen, Allen, “Natural Deduction Systems in Logic” in *The Stanford Encyclopedia of Philosophy (Spring 2024 Edition)*, ed. Zalta, E. N. and U. Nodelman (2024).

```
#
# The proof is initialized by calling this macro, which
# informs the scripts on the number of steps in the proof
# and its maxiumum depth, i.e., how many proofs we have
# inside proofs.
#

set_steps_and_depths(8,3)

#
# Any proof (the root proof or any sub-proof) starts
# with the start_proof() which also names that proof.
#
# after started we add its premises, and end it with
# premis_end()

start_proof(START);
add_premis(START,"A∨B");
add_premis(START,"¬A");
premis_end(START);

#
# Here comes the sub-proofs
#

start_proof(SUB1);
add_premis(SUB1,"A");
premis_end(SUB1);

#
# The add_step() macro has three argument, the name of the
# current proof, the result of the step, and finally the
# references to the steps needed for reaching the step.
#

add_step(SUB1,"⊥","⊥ Intro: 3,2");
add_step(SUB1,"B","⊥ Elim: 4");
end_proof(SUB1);

start_proof(SUB2);
add_premis(SUB2,"B");
premis_end(SUB2);
add_step(SUB2,"B","Reit: 6");
end_proof(SUB2);

add_step(START,"B","∨ Elim: 6-7,3-5,1");
end_proof(START)
```

Figure 2. The PIC code needed to generate Figure 1.

1	$A \vee B$	
2	$\neg B \vee C$	
3	A	
4	$A \vee C$	$\vee$ Intro:3
5	B	
6	$\neg B$	
7	$\bot$	$\bot$ Intro:6,5
8	$A \vee C$	$\bot$ Elim:7
9	C	
10	$A \vee C$	$\vee$ Intro:9
11	$A \vee C$	$\vee$ Elim:6-8,9-10,2
12	$A \vee C$	$\vee$ Elim:1,3-4,5-11

Figure 3. A longer example: Prove that $A \vee B, \neg B \vee C \therefore A \vee C$.