## 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	$\neg$	\[no]
u+2227	^	\[AN]
u+2228	<b>V</b>	\[OR]
u+2200	$\forall$	\[fa]
u+2203	3	\[te]
u+2192	$\rightarrow$	\[->]
u+2194	$\leftrightarrow$	\[<>]
u+22A5	$\perp$	\[pp]
u+22A2		
u+2261	<b>=</b>	\[==]
u+25A1		\[sq]
u+25C7		L 13
u+2234	<i>:</i> .	\[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	$A \lor B$		
2	$\neg A$		
3		A	
4		Т	⊥ Intro: 3,2
5		В	⊥ Elim: 4
6		В	
7		В	Reit: 6
8	В		∨ Elim: 6-7,3-5,1

Figure 1. Proof that  $A \lor B$ ,  $\neg A : B$ . 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," Computing 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 \lor B");
add_premis(START,"\neg 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, "\perp", "\perp Intro: 3,2");
add_step(SUB1, "B", "\bot 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", "\vee Elim: 6-7, 3-5, 1");
end_proof(START)
```

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

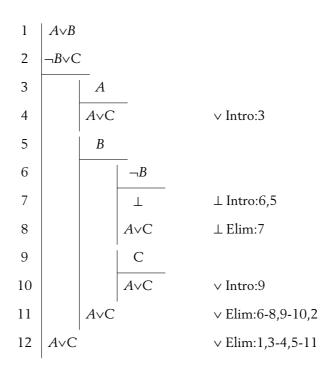


Figure 3. A longer example: Prove that  $A \lor B$ ,  $\neg B \lor C :: A \lor C$ .