

Lab Handout 1, Math 387

Dr. Holmes

February 9, 2004

This is the corrected version.

This is a brief guide to using my sequent prover on the MG104 machines.

The prover runs under the Moscow ML interpreter, which is installed on the server for the MG104 machines. To invoke the Moscow ML interpreter, type `mosml` at the command prompt in a terminal window.

Before invoking the ML interpreter, you need the source for the prover. This is in the file `/home/public/holmes/marcelsequent.sml`. The command `cp /home/public/holmes/marcelsequent.sml .`

typed in a terminal window should get it for you. Note that a space after `cp` and a space followed by a period at the end are needed, as shown.

Now type

```
mosml
```

to invoke the ML interpreter.

Type these lines

```
compile "marcelsequent.sml";  
load "marcelsequent";  
open marcelsequent;
```

```
(* load and open were interchanged in the original version *)
```

and you are in business. You only need to type the first line if you are actually compiling the prover, which you only need to do the first time you copy it into your directory. You will need to compile it again if you have to get a new version (this might happen between now and the next lab; I've already made some changes in the program preparing for this lab).

It's useful to know that all ML command lines end with a semicolon, and that any command without an argument needs a dummy argument (). You should see this in the examples.

Things you need to know about the prover:

To begin a proof, type a line like

Start "(P1->P1)";

In a propositional calculus proof, this is the only time you will have to type an expression yourself. The prover uses somewhat different notation:

\wedge	&
\vee	v
\rightarrow	- >
\leftrightarrow	==
\neg	~
\oplus	= / =
\forall	&
\exists	v
\in	E

It requires all possible parentheses. This is something I may have fixed by the next lab session; for the time being, I suggest writing your expressions in an emacs window (I'll help you with this); emacs will count parentheses for you, then you can drag-and-drop them into the prover window.

Propositional letters are all of the forms P1, P2, P3, etc. Variables (when we get to predicate logic) are all of the form x1, x2, x3, etc. and constants are of the form a1, a2, a3, etc. When I'm fixing the precedence I may take a look at this, too.

The prover displays a sequent as two numbered lists of sentences, separated by a \vdash , vertically rather than horizontally. The top list is the left side of the sequent and the bottom list is the right side.

The prover commands `LeftRule()`; and `RightRule()`; will automatically apply the appropriate rule (it will know which one) to the first sentence on the left or right side of the sequent, respectively and redisplay the sequent.

The prover commands `GetLeft n`; and `GetRight n`;, where `n` is a number, will move line `n` on the left or right to the front. You'll need this to apply rules to the right term or to recognize that a sequent is finished.

The command `Done()`; will close off the current proof if the first lines of the two sides of the sequent are the same (and automatically send you to

the next line to continue working, if any lines remain unfinished). If nothing happens after `Done()`; , the proof is complete.

The command `showall()`; will show you the current state of your proof. You need to hit return repeatedly to see everything with this command.

The command `saveproof "name"`; will save your proof to a text file called `"name.prf"`.

These are all the commands you need for propositional logic proofs with this prover. It also does predicate logic, but we'll look at that later.

Your lab assignment is to carry out several of the proofs we have done in class. I would like this to include

$$((P \vee Q) \wedge (\neg Q \vee R)) \rightarrow (P \vee R)$$

among others.

When you do an example, print out the proof and convert it manually back to our sequent notation (a proof tree). Notice that the prover may write sentences in sequents in an unexpected order, so you may have to think just a little to recognize what rule is being applied.

Try taking at least one example like

$$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$$

where you have a choice of which rule to apply and try making each choice and diagramming the two separate proofs manually.

Your lab writeup should include printed proofs and sequent diagrams prepared manually from the printed proofs. I'd like to have at least two other examples in addition to the two explicitly given here. An example of extraction of a counterexample from a non-tautology would also be good. In this case the printout will not show your full proof tree: it may show some proved subgoals (labelled with -Proved-), and it will show the invalid sequent representing at least some of the counterexamples you find (unproved goals are labelled with -Goal- in the printout). I wrote originally that the prover is stalled when it encounters an invalid sequent, but this is not the case. The command `Rotate n` rotates the proof tree n levels down from your position, which will change the goal you are working on. Try `Rotate 1`; , `Rotate 2`: in succession to get to a different goal.