Computer Lab I, Math 187, Fall 2017

Dr. Holmes

September 28, 2017

The theorem prover we are using (called Marcel) is implemented in the Python program `pythonmarcel.py` found at `http://math.boisestate.edu/~holmes/marcelstuff/pythonmarcel.py`. You need to open this program in a Python shell.

As we will see in the class demonstrations, at any given point the program displays an argument (a list of premises and a conclusion) which it is trying to show is valid. Negative premises whose numbers are starred have a special status (they are actually treated as alternative conclusions).

The basic commands you will use are

Start, s: enter a statement to prove: for example `s('p?  -> p?')`

Right,r: take appropriate action regarding the conclusion. `r()` is the command. If the conclusion you see is `_|_` (ASCII art for $\perp$), this command will have no effect.

Left,l: take appropriate action regarding the *first* premise. `l()` is the command. This command will not work on a starred negative premise.

gl: rotate the premise numbered by the argument to the top. `gl(3)` is an example of use of this command. This command will not work on a starred negative premise.

gr: rotate the starred negative premise numbered by the argument into the conclusion (and remove the negation; the conclusion will have a negation added and become a starred negative premise): `gr(3)` is an example of use of this command.

Done(): Use this when the first premise and the conclusion are the same to go on to the next bit. This command will not work if the first premise is a starred negative premise.

Showall(): This command will show you your proof.

These are all the basic commands you need for propositional logic. You can put more than one command on a line: separate them with semicolons.

The syntax of statements is governed by the fact that I'm working in ASCII.

propositional letters: strings of lower case letters followed by a question mark.

negation: $\neg P$ becomes $\sim$`p?`

conjunction: $P \wedge Q$ becomes `p? & q?`

disjunction: $P \vee Q$ becomes `p? V q?` [notice that is a capital V]

implication: $P \rightarrow Q$ becomes `p? -> q?` (note the ASCII art)

biconditional: $P \leftrightarrow Q$ becomes `p? == q?`

The prover has order of operations built in: it does negation first, then conjunction, then disjunction, then implication, then biconditional. Write as many parentheses as you need; Marcel will drop ones that it doesn't need.

Before you start your proof you want to open a log file: for example `setlog('JaneDoeproblem1')`. This will create a file `JaneDoeproblem1logfile.py` which will contain Marcel commands which will recapitulate your proof (evidence that you did it!). When you are finished, issue the command `setlog('done')` to close this file so you can read it. And you might want to check it.

If you want to add to a log file that already exists rather than overwrite it, you can open it with `addtolog('JaneDoeproblem1')` (where of course your file name will be different). Again, you should type `setlog('done')` when you are done. [this actually creates a dummy log file `donelogfile.py`; just ignore it].

1 Exercises

1. Carry out the proof of `p? & q? -> r? == p? -> (q? -> r?)`, an example I am doing all or part of in class today.

2. Prove the deMorgan laws $\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q$, and the other one.

3. Prove $(P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$.

4. Prove $\neg(P \rightarrow Q) \leftrightarrow P \wedge \neg Q$.

5. Prove $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow S)) \rightarrow R \vee S$ (showing this theorem amounts to showing the validity of the classical rule called *constructive dilemma*). You are going to need parentheses to write this!

When you are finished with the exercises, email your log files to me. Make sure that the name of your log file(s) include your name and enough information to identify the lab and problem9s) covered. The official due date is a week from today, but I'm fairly relaxed about labs.