Lab Guide to the web-based version of the Marcel Theorem Prover

Dr Holmes

January 29, 2021

Contents

1 Starting out

Marcel is a system for computer assisted theorem proving. I have used it before in classes; some of you may have used it.

You need Python to run the software. I typically run it by editing the file `pythonmarcel.py` with IDLE and running the module.

To start a proof, type the command **s** followed by a logic expression. For example, here is a sample interaction:

```
Enter quit to break out of interface:
```

```
>>  s p?->p?


Line 1:


------------------

1:  p? -> p?



Enter quit to break out of interface:

>>
```

We are only interested in a limited subset of the language of logical expressions of Marcel, for the moment. A lower-case letter (or string of lower case letters) followed by a question mark is a propositional letter (a sentence with an unknown value of True or False). The logical operators have different shapes because of the limitations of ASCII:

1. instead of using $P, Q, R \ldots$ to represent unknown sentences, we use `p?, q?, r?...` (lower case letters followed by question marks);
2. instead of $P \wedge Q$, we type `p?&q?`;
3. instead of $P \vee Q$ we type `p?Vq?` (that needs to be a capital V);
4. instead of $P \rightarrow Q$ we type `p?->q?`;
5. instead of $P \leftrightarrow Q$ we type `p?==q?`; instead of $\neg P$ we type `~p?`; one should leave a space in front of the negation symbol if it is preceded by another operation symbol.

So you can see that what I wrote above is $P \rightarrow P$, and I should be able to prove that.

```
>>  r


Line 2
prove
p? -> p?
by assuming
p?
and deducing
p?:

1:  p?

------------------

1:  p?


Enter quit to break out of interface:

>>
```

We look first at the product of this move before we look at the command. Marcel manipulates *arguments*, and its aim is to show that arguments are valid by suitable manipulations. The argument produced here is

$$\frac{P}{P}$$

which is clearly valid.

The `r` command looks at the conclusion of the previous argument and uses it to guide construction of a simpler argument or arguments. In this case, Marcel is following the familiar strategy for proving an implication: to prove $A \to B$, assume A and try to show B.

The action of the `s` command can be explained here too. When we set out to prove a theorem T, Marcel's view of what we are doing is that we are trying to show that

$$\frac{}{T}$$

is a valid argument. If all of zero premises are true, T follows, as it were.

On the next page we finish this trivial proof.

```
Enter quit to break out of interface:

>>  d


Q. E. D.

Line 1:


------------------

1:  p? -> p?



Enter quit to break out of interface:

>>
```

The `d` command signals to Marcel that we recognize that the sequent is valid. The one circumstance we will see in our examples where we can do this is if the first premise and the conclusion are the same (there are some other situations where `d` will work, but we will not see them in propositional logic).

I'll talk about the actual manipulations of arguments that Marcel does in class. Here I am focussed on how the system works. I will do two examples to illustrate the commands.

For reference I give the five basic commands used in propositional logic proofs (though a real explanation of them is best obtained by reading the examples and actually trying them out). At any moment, Marcel is manipulating an argument, a list of premises from which a conclusion is to be show to follow. There might be no premises (the conclusion is to be proved as a theorem); there might be no conclusion (a contradiction is to be deduced from the premises).

The basic commands are

1. `s` (logical expression): start to prove that the logical expression is a theorem.
2. `r`: manipulate the argument based on the form of the conclusion (try to prove the conclusion as a goal).
3. `l`: manipulate the argument based on the form of the first premise (try to use the first premise as a given statement).
4. `gl`(number): move the indicated numbered premise to first position.
5. `d`: recognize that the current argument is valid (because the first premise and the conclusion are identical): Marcel will then present you with the next thing you have to prove, or tell you that you have proved the theorem you are working on (Q. E. D signals end of proof).

2 First example

```
>>  s ((p? & q?) -> r?) == (p? -> (q?->r?))


Line 1:


------------------

1:  p? & q? -> r? == p? -> q? -> r?
```

We use the `s` command to start the proof of $P \wedge Q \to R \leftrightarrow P \to (Q \to R)$. We make a point by supplying all possible parentheses in the first line. Notice that Marcel, which does know about order of operations conventions for these operations, doesn't need any of the parentheses and does not display them. You will find in other examples that Marcel does know when parentheses are important.

```
Enter quit to break out of interface:

>>  r


Line 2
proving biconditional
p? & q? -> r? == p? -> q? -> r?
Part I =>
:

1:  p? & q? -> r?

------------------

1:  p? -> q? -> r?
```

The `r` command acts on the conclusion, and if you read the hint, it suggests that Marcel is following the familiar strategy of the biconditional. Here it is presenting Part I of the proof: it will present Part II to be proved automatically when Part I is finished..

```
Enter quit to break out of interface:

>>  r


Line 4
prove
p? -> q? -> r?
by assuming
p?
and deducing
q? -> r?:

1:  p?
2:  p? & q? -> r?

------------------

1:  q? -> r?
```

Again, the `r` command acts on the conclusion of the previous argument, which is an implication, and it follows our usual strategy for proving an implication, adding the hypothesis of the complication as a new premise and making the conclusion of the implication the new conclusion of the argument.

```
Enter quit to break out of interface:

>>  r


Line 5
prove
q? -> r?
by assuming
q?
and deducing
r?:

1:  q?
2:  p?
3:  p? & q? -> r?

------------------

1:  r?
```

We do the same thing again!

```
Enter quit to break out of interface:

>>  gl 3


Line 5
prove
q? -> r?
by assuming
q?
and deducing
r?:

1:  p? & q? -> r?
2:  q?
3:  p?

------------------

1:  r?
```

Marcel commands always manipulate either the conclusion or the first premise of an argument. Here, the first premise and the conclusion are just letters, so we use the command `gl 3` to bring the third premise to the front, so we can manipulate it.

```
Enter quit to break out of interface:

>>  l


Line 6
use
p? & q? -> r?
, first part, showing that
p? & q?
or the desired conclusion holds:

1:  q?
2:  p?
3:  ~r?

------------------

1:  p? & q?
```

We use the `l` command to manipulate the first premise. This means we are using the premise $P \wedge Q \to R$ in the argument. The strategy of use of an implication $A \to B$ is basically to attempt to prove A, and when A has been proved, show that the desired consequence follows from B (which is appropriate because modus ponens then gives B). I'll talk about the exact way Marcel sets up this argument in class.

Notice that as expected we are now trying to prove $P \wedge Q$, and we should have the idea that it will not be difficult from the given premises.

```
Enter quit to break out of interface:

>>  r


Line 8
prove
p? & q?
first part:  prove
p?:

1:  q?
2:  p?
3:  ~r?

------------------

1:  p?
```

Proving $P \wedge Q$ breaks down into proving P then proving Q.
This argument is obviously valid, since the conclusion is one of the premises!

```
Enter quit to break out of interface:

>>  gl 2


Line 8
prove
p? & q?
first part:  prove
p?:

1:  p?
2:  ~r?
3:  q?

------------------

1:  p?

Enter quit to break out of interface:

>>  d


Line 9
prove
p? & q?
second part:  prove
q?:

1:  q?
2:  p?
3:  ~r?

------------------

1:  q?
```

Marcel can be informed (with the `d` command) that it should record an argument as valid, when the first premise and the conclusion are the same.

Before we could do this, we used `gl 2` to move the appropriate premise to the front.

We made the text smaller so we could fit these remarks on one page. Note that after the `d` command Marcel serves up the second part of the proof of $P \wedge Q$.

```
Enter quit to break out of interface:

>>  d


Line 7
use
p? & q? -> r?
second part, show that the desired conclusion follows
      from
r?:

1:  r?
2:  q?
3:  p?

------------------

1:  r?
```

Q gets proved by a single application of `d`. Marcel then presents us with the task (since we proved $P \wedge Q$) of using R, which we can deduce by modus ponens...to prove our current conclusion, which is also R...

```
Enter quit to break out of interface:

>>  d


Line 3
proving biconditional
p? & q? -> r? == p? -> q? -> r?
Part II <=
:

1:  p? -> q? -> r?

------------------

1:  p? & q? -> r?
```

R gets proved directly by applying `d`. Once this is done, the proof of Part I is complete, and I am going to leave Part II for you to read and think about.

```
Enter quit to break out of interface:

>>  r


Line 10
prove
p? & q? -> r?
by assuming
p? & q?
and deducing
r?:

1:  p? & q?
2:  p? -> q? -> r?

------------------

1:  r?
```

```
Enter quit to break out of interface:

>>  l


Line 11
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  p?
2:  q?
3:  p? -> q? -> r?

------------------

1:  r?
```

```
Enter quit to break out of interface:

>>  gl 3


Line 11
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  p? -> q? -> r?
2:  p?
3:  q?

------------------

1:  r?
```

```
Enter quit to break out of interface:

>>  l


Line 12
use
p? -> q? -> r?
, first part, showing that
p?
or the desired conclusion holds:

1:  p?
2:  q?
3:  ~r?

------------------

1:  p?
```

```
Enter quit to break out of interface:

>>  d


Line 13
use
p? -> q? -> r?
second part, show that the desired conclusion follows
      from
q? -> r?:

1:  q? -> r?
2:  p?
3:  q?

------------------

1:  r?
```

```
Enter quit to break out of interface:

>>  l


Line 14
use
q? -> r?
, first part, showing that
q?
or the desired conclusion holds:

1:  p?
2:  q?
3:  ~r?

------------------

1:  q?
```

```
Enter quit to break out of interface:

>>  gl 2; d


Line 14
use
q? -> r?
, first part, showing that
q?
or the desired conclusion holds:

1:  q?
2:  ~r?
3:  p?

------------------

1:  q?
```

Notice here that you can put more than one command on a line, separated by semicolons.

```
Line 15
use
q? -> r?
second part, show that the desired conclusion follows
      from
r?:

1:  r?
2:  p?
3:  q?

------------------

1:  r?
```

```
Enter quit to break out of interface:

>>  d


Q. E. D.

Line 1:


------------------

1:  p? & q? -> r? == p? -> q? -> r?




Enter quit to break out of interface:

>>
```

And finally, it reports the proof of the theorem.

This example shows us all the proof commands we use: `s` to start the proof, `r` to carry out a reasoning step based on the conclusion of the previous argument, `gl n` to bring the nth premise to first position, and `d` to report to Marcel that the first premise and the conclusion are in agreement.

3 Second example

The next example will show some things which can happen with the display and the rules which we haven't seen in the first example, basically because the first example doesn't involve negation.

```
>>  s ~(p? & q?) == ~p? V ~q?


Line 1:


------------------

1:  ~(p? & q?) == ~p? V ~q?
```

We are proving $\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q$, one of the versions of deMorgan's law.

```
Enter quit to break out of interface:

>>  r



Line 2
proving biconditional
~(p? & q?) == ~p? V ~q?
Part I =>
:

1:  ~(p? & q?)

------------------

1:  ~p? V ~q?
```

First, we apply the usual strategy for proving a biconditional.

```
Enter quit to break out of interface:

>>  r


Line 4
prove
~p? V ~q?
by denying
~q?
and showing
~p?:

1:  ~(p? & q?)
2:  q?

------------------

1:  ~p?
```

Marcel applies the alternative elimination strategy: to prove $A \vee B$, it assumes $\neg B$ and argues to A. Marcel carries out a double negation automatically here.

```
Enter quit to break out of interface:

>>  l


Line 5
use
~(p? & q?)
by denying conclusion and proving
p? & q?:

1:  q?
2:  p?

------------------

1:  p? & q?
```

Marcel uses a negative hypothesis $\neg A$ to reason to a conclusion C by assuming $\neg C$ and deducing A (this is valid as a contrapositive). This is slightly disguised because again Marcel automatically applies a double negation (proving $\neg P$ from $\neg(P \wedge Q)$ turns into proving $P \wedge Q$ from P).

```
Enter quit to break out of interface:

>>  r


Line 6
prove
p? & q?
first part:  prove
p?:

1:  q?
2:  p?

------------------

1:  p?
```

Proving $P \wedge Q$ turns into two proofs, one of P and one of Q.

```
Enter quit to break out of interface:

>>  gl 2; d


Line 6
prove
p? & q?
first part:  prove
p?:

1:  p?
2:  q?

------------------

1:  p?



Line 7
prove
p? & q?
second part:  prove
q?:

1:  q?
2:  p?

------------------

1:  q?
```

P proved, we turn to Q.

```
Enter quit to break out of interface:

>>  d


Line 3
proving biconditional
~(p? & q?) == ~p? V ~q?
Part II <=
:

1:  ~p? V ~q?

------------------

1:  ~(p? & q?)
```

Proving Q completed the proof of the first part of the main biconditional. Now we start the proof of the second part.

```
Enter quit to break out of interface:

>>  l


Line 8
using hypothesis
~p? V ~q?
first part: assume case 1,
~p?:

1:  ~p?

------------------

1:  ~(p? & q?)
```

I chose to use `l` to demonstrate Marcel using the method of proof by cases (exactly as in our paper system). Here is case 1 with the hypothesis $\neg P$; when it is finished Marcel will resume with the second case, assuming $\neg Q$. The proof could have been carried out by manipulating the conclusion as well.

```
Enter quit to break out of interface:

>>  r


Line 10
prove
~(p? & q?)
by assuming
p? & q?
and deducing a contradiction or alternative conclusion
     :

1:  p? & q?
2:  ~p?

------------------

_|_
```

When we aiim to prove a negative conclusion $\neg A$, Marcel assumes A and reasons to a contradiction. The argument here actually has no conclusion: it is valid because it is impossible for all the premises to be true (as we will show).

```
Enter quit to break out of interface:

>>  l


Line 11
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  p?
2:  q?
3:  ~p?

------------------

_|_
```

We break down the premise $P \wedge Q$ into two premises, an entirely natural move.

```
Enter quit to break out of interface:

>>  gl 3; d


Line 11
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  ~p?
2:  p?
3:  q?

------------------

_|_
```

```
Line 11
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  ~p?
2:  p?
3:  q?

------------------

_|_
```

I think I tried to use `d` and it failed to work.

```
Enter quit to break out of interface:

>>  l


Line 12
use
~p?
by denying conclusion and proving
p?:

1:  p?
2:  q?

------------------

1:  p?
```

Using the premise $\neg P$ to argue to a contradiction can be reduced to arguing for the conclusion P.

```
Enter quit to break out of interface:

>>  d


Line 9
using hypothesis
~p? V ~q?
second part: assume case 2,
~q?:
```

```
1:  ~q?

------------------

1:  ~(p? & q?)
```

And now `d` works, and then serves up the other case of the proof by cases, which is argued in basically the same way (I will not comment it).

```
Enter quit to break out of interface:

>>  r


Line 13
prove
~(p? & q?)
by assuming
p? & q?
and deducing a contradiction or alternative conclusion
     :

1:  p? & q?
2:  ~q?

------------------

_|_
```

```
Enter quit to break out of interface:

>>  l


Line 14
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  p?
2:  q?
3:  ~q?

------------------

_|_
```

```
Enter quit to break out of interface:

>>  gl 3


Line 14
use
p? & q?
by breaking it into its parts
p?
and
q?:

1:  ~q?
2:  p?
3:  q?

------------------

_|_
```

```
Enter quit to break out of interface:

>>  l


Line 15
use
~q?
by denying conclusion and proving
q?:

1:  p?
2:  q?

------------------

1:  q?
```

```
Enter quit to break out of interface:

>>  gl 2; d


Line 15
use
~q?
by denying conclusion and proving
q?:

1:  q?
2:  p?

------------------

1:  q?




Q. E. D.

Line 1:


------------------

1:  ~(p? & q?) == ~p? V ~q?




Enter quit to break out of interface:

>>
```

And this completes the proof.

4 Strategy (in fact, you don't need much...)

It is actually a theorem that if you attempt to prove a propositional logic theorem in Marcel and you take pains to simplify something at each step, and move complex premises into first position when you can no longer simplify something, you *will* succeed in proving the theorem. Marcel does this while using a single strategy for each logical form of a premise and a single strategy for each logical form of a conclusion (most of these being the same as the strategies used in my paper system). The only strategy which looks different is the strategy for using an implication, and with experience many student eventually recognize the strategy as modus ponens in disguise (it looks different because Marcel is acting just on a premise $A \to B$ rather than on two premises A and $A \to B$: but notice that if you have the premise A as well you can immediately prove the first argument with `d`, since you will have A as both premise and conclusion.

Marcel supports the logic of quantifiers, the logic of equality, and a system of set theory as well; we might look at quantifiers later. It doesn't guarantee proofs in these more complicated situations as it does in propoiitional logic (and further, if you take an unnatural approach to proving a theorem, it can take quite a while).

5 Saving your work

This section tells you how to save your work so you can send it to me.

If you type the command `SL` followed by a filename, a log file will be created with a name `(your filename)logfile.ilg`. Commands you issue to Marcel will be saved in this file. You won't be able to look at the log file until you close it: I usually close it by issuing the command `SL done` (which creates a new dummy log file, but we don't use it).

So, if you issue `SL (your file name)` before starting a proof and `SL done` after it, you will get a file in the left pane (you can look at it by clicking the tab) which contains all the commands you used to build your proof. If you type `LP` (for "log proof") **before** `SL done`, Marcel will insert an almost

human-readable proof of your theorem into the log file. To hand in your work, the best approach is to build a log file of your proof, cut and paste the log file into a text file and email it to me. One file per proof is acceptable. The names of files you mail to me should include your name.

6 Exercises

Carry out the following proofs in Marcel. I am giving you the exercise of translating our book notation into Marcel notation: this is actually a common mathematical communication issue, converting more usual mathematical notation into a form understood by software.

Please turn in your work by email. You can use the instructions of the previous section to get nicely formatted files to email to me. It is not unacceptable to cut and paste the Python window with your work in it into a text file, though I would prefer the format described above. It is officially due on Feb 10 2021, but this is not inflexible.

You get a check mark for handing it in in the gradebook: the point here is exposure, and I reiterate that it is a mathematical theorem that if you set up your theorem correctly and keep simplifying statements in your proofs, you *will* prove each of these eventually. You might however get some ideas about practical proof strategy in the course of working on the larger proofs.

1. Prove transitivity of implication, $(P \to Q) \wedge (Q \to R) \to (P \to R)$. You might get a feeling for why the rule for handling implications as premises is really rather like modus ponens.

2. Prove $((P \vee \neg Q) \wedge (\neg P \vee R)) \to (Q \to R)$. This was a problem in the second logic homework set.

3. Prove $(P \vee Q) \wedge (P \to R) \wedge (Q \to S) \to (R \vee S)$. This amounts to showing the validity of the rule of constructive dilemma. Start with `r; l; l` to make it look exactly like constructive dilemma. It might be interesting to try using each of the two different proof strategies I used in the notes (alternative elimination, starting with the conclusion, or proof by cases, starting with the first premise).

4. Set up and prove the rule of destructive dilemma in the same way (this is a problem in your current homework).

5. Prove the other de Morgan law $\neg(P \vee Q) \leftrightarrow \neg P \wedge \neg Q$.

6. (optional, this is long) Prove the perhaps surprising theorem

$$((A \leftrightarrow B) \leftrightarrow C) \leftrightarrow (A \leftrightarrow (B \leftrightarrow C)).$$

   You might want to think about what $A \leftrightarrow B \leftrightarrow C$ actually means: it does not mean that A, B, C are equivalent!