Sample Book

Author

June 2, 2017

Contents

Part I

Installation and first steps

installation

0.0.1 Mac OSX

1. Download and install the latest version of Coq (it needs to be at least 8.6) from :

   https://coq.inria.fr/download

   Move it to your apps folder.

2. Download and unpack spatchocq.app from canvas move the spatchcoq.app to Applications and start it.

3. when prompted find the Coq installation you have just move above. Navigate to

   `/Applications/CoqIDE_8.6.app/Contents/Resources/bin/`

   and choose coqtop. See Figure 1

   You only do this once.

4. You only have to do this once. You might also need to install gtk, the simplest way to do this is via homebrew

   brew install gtk+

5. enjoy

0.0.2 Windows

1. get the zipfile spatchcoq.zip, unzip it in a folder on a usb stick and doubleclick the application file spatchcoq. Note this version includes an instalation of Coq (not very extensively tested yet)

2. enjoy

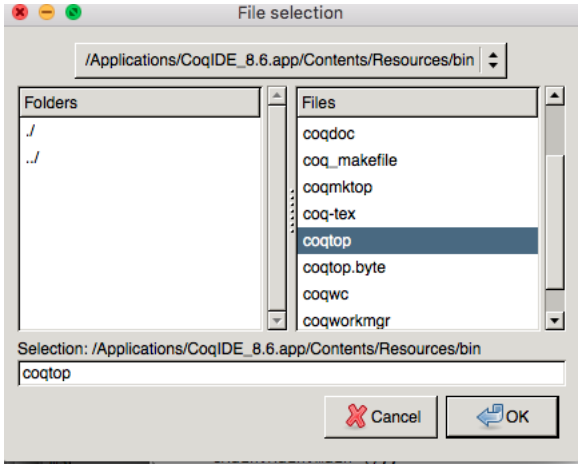

Figure 1: Choose the Coq app in a Mac env

0.0.3 Linux

1. Download and install the latest version of Coq it needs to be at least 8.6 so do not use apt-get install coq
2. go to https://github.com/corneliuhoffman/spatchcoqocaml/tree/master to build from scratch.
3. when prompted go to the Coq folder you just installed with opam and find the application called **coqtop**
4. enjoy

0.1 Introducing the GUI

0.1.1 Main windows

Figure 2 is a view of the GUI. As you can see there are 4 different windows and three buttons.

1. The Green window : This is the window that keeps the text that has already been processed.
2. The Yellow window ': This is the only window you can type your commands into.
3. The Gray window : this is the Coq feedback window.

4. The White window : this is a window for messages.
5. The run button: this sends the first line from the input window to Coq.
6. The undo button: this undoes the last command.
7. The draw tree button: this draws the proof trees for all the completed theorems,.

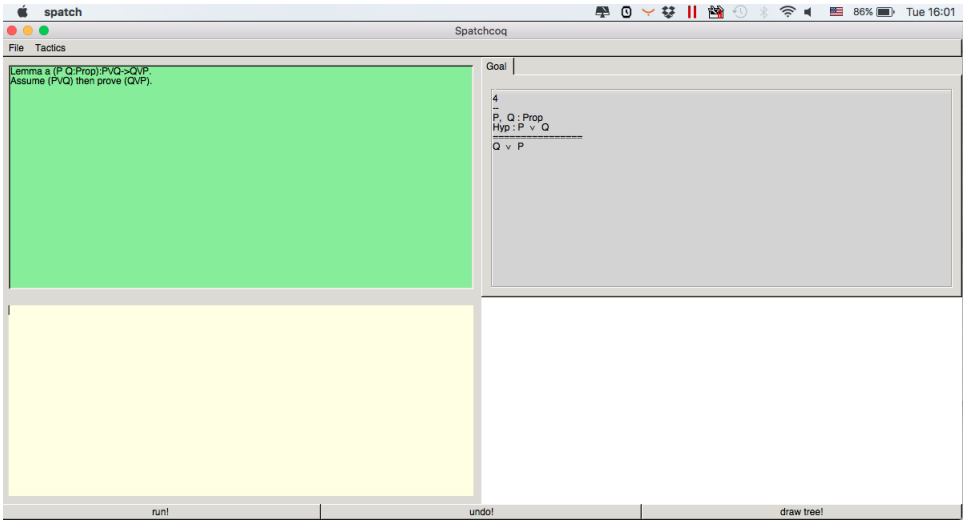

Figure 2: the GUI

0.1.2 The menus

The File menu (Figure 3) is quite standard:

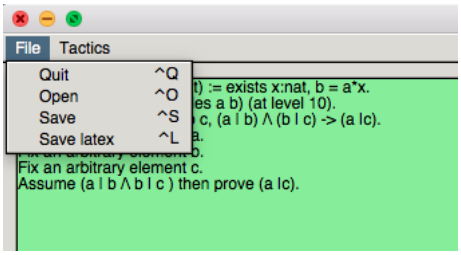

Figure 3: the File Menu

The Tactics menu (Figure 4) allows one to pick one of the predefined tactics. Note the place keeper VAR These can be modified in the customisation menu at start. More on these later.

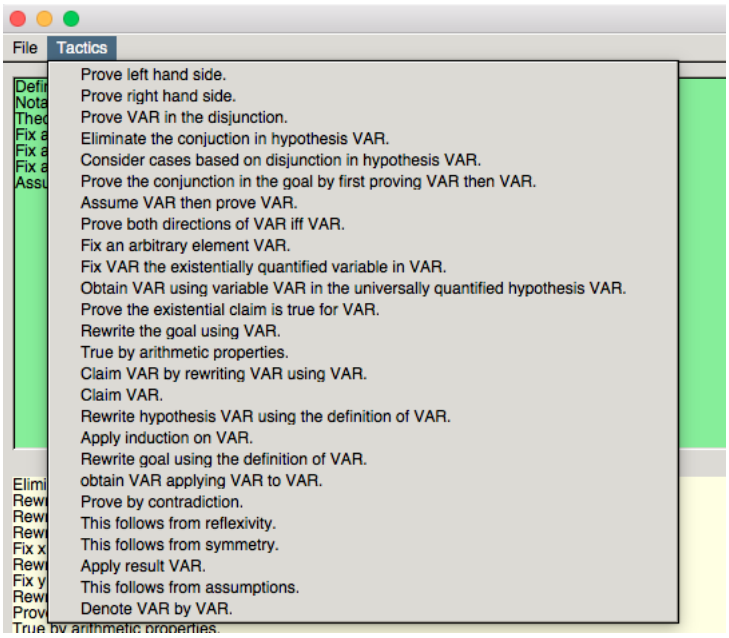

Figure 4: the Tactics/Environment Menus

0.1.3 Keyboard shortcuts

Pressing ESC autocompletes the commands and pressing CTRL-r circles around the various possibilities for VAR.

0.2 Two simple examples

We give two detailed examples that will exemplify the mechanics of the GUI. For clarity we will use colour boxes that will exemplify the window that we refer to. So green boxes refer to the processed window, yellow ones to the input window and gray ones to the feedback window.

0.2.1 Propositional Calculus

We will prove that if P and Q are propositions then

$$P \vee Q \Rightarrow Q \vee P$$

the way to enter this is:

```
Lemma commor(P Q :Prop):  P \/ Q -> Q \/ P .
```

Note that the feedback from Coq says

```
Lemma commor(P Q :Prop):  P \/ Q -> Q \/ P .
```

$P, Q : Prop$
==========
$P \vee Q \to Q \vee P$

This means that the hypotheses are that P and Q are propositions and the conclusion is $P \vee Q \to Q \vee P$. To prove an implication statement we assume the left hand and try to prove the right hand. Here is how you do it in Spatchcoq:

Type "Assume" and press ESC to get

```
Assume VAR then prove VAR.
```

Press CTRL-r to select the first VAR.

write $(P \vee Q)$ to replace the first VAR. Repeat CTRL-r and and replace the second VAR by $(Q \vee P)$.

The text in the yellow window should now be

```
Assume (P ∨ Q) then prove (Q ∨ P).
```

Click run.

The response from Coq is

$P\ Q$: Prop
Hyp : $P \vee Q$
==========
$Q \vee P$

This reflects the fact that we have a new hypothesis tabled Hyp and a new conclusion.

Of course since we have a hypothesis with a disjunction we will use an argument by cases. To do so, type "cases" and press ESC. Choose the following:

```
Consider cases based on disjunction in hypothesis VAR.
```

Press CTRL-r and replace VAR by Hyp. Click run.

Notice that there are now two goals:

$P\ Q$: Prop
Hyp0 : P
==========
$Q \vee P$

and

$P\ Q$: Prop
Hyp1 : Q
==========
$Q \vee P$

corresponding to the two cases to consider. In first goal we will prove the right hand side of the disjunction in the conclusion. To do so, type "right" and press ESC. You get to pick

```
Prove right hand side.
```

and after clicking run you will get the following feedback (note that the second goal stays unchanged)

$P\ Q$: Prop
Hyp0 : P
==========
P

Finally you can finish this goal by using the hypothesis Hyp0. To do this you use

```
This follows from assumptions.
```

Note that you have now finished this goal. Repeat the argument for the second goal by using:

```
Prove left hand side.
```

```
This follows from assumptions.
```

to get

```
no more goals
```

Now type

```
Qed.
```

to save the theorem. It now appears among the proved theorems:

and you can see its proof tree by clicking on draw tree:

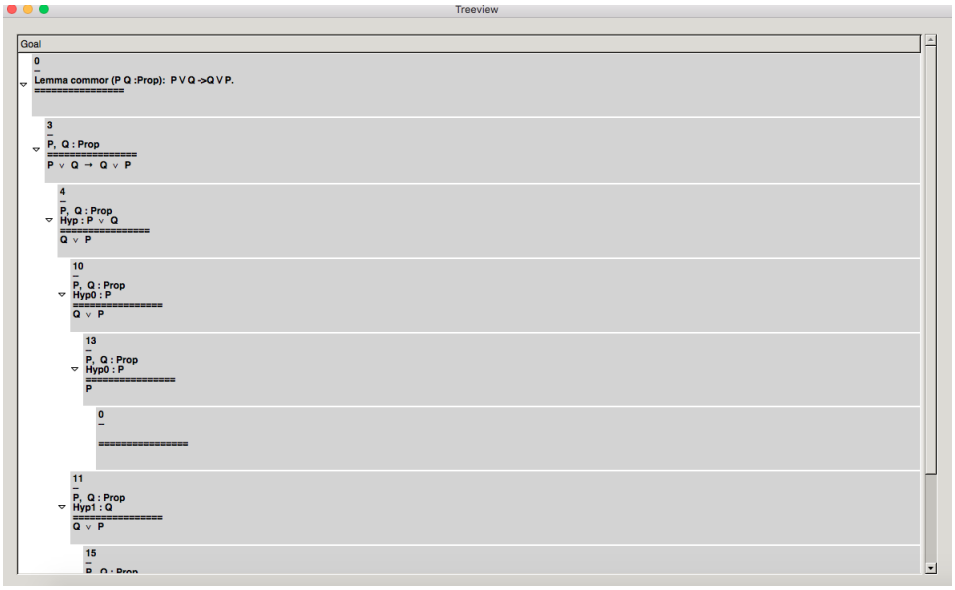

Figure 5: the tree window

0.2.2 An elementary number theory example

We shall prove the transitivity of divisibility. That is we will prove that

$$\forall a, b, c \in \mathbb{N}, a|b \wedge b|c \Rightarrow a|c$$

IN the process we will introduce definitions and notations.

To start we note that we will be talking about objects of type `nat`. We weill introduce the following definition

```
Definition divides a b := exists x:nat, b = a*x.
```

We hope that the format is quite clear, it resembles the one we used before but uses a few new notions, the operator `:=` which the dines the function `divides` and the quantifier `exists`. Note that we have not explicitly stated that a and b should be natural numbers, Coq will deduce that from the context. We could have been very precise as follows:

```
 Definition divides (a b:nat) := exists x:nat, b = a*x.
```

Note that the definition will not get any feedback from Coq. If we want to check that we have correctly defined our notion we can use

```
Check divides.
```

to get

```
Query commands should not be inserted in scripts

divides
:  nat -> nat -> Prop
```

or

```
Print divides.
```

to get a more detailed

Query commands should not be inserted in scripts

divides = $\lambda ab : nat, \exists x : nat, b = a * x$
: nat -> nat -> Prop
Argument scopes are [nat_scope nat_scope]

We will nod describe all this output here but we note the change from `exists` to $\exists$ and the occurrence of λ, a notation for functions.

Next we define a notation for divides

```
Notation " a | b " := (divides a b) (at level 10).
```

Again no feedback from Coq. The definition should be self-evident except for the "(at level 10)" part. We will discuss this elsewhere.

We are now ready to state out theorem

We can state the theorem as (see the corresponding feedback)

```
Theorem refldiv (a b c:nat):
(a | b) ∧ (b | c) -> (a |c).
```

$a, b, c : nat$
==========
$a|b \wedge b|c \rightarrow a|c$

but we prefer the version

```
Theorem refldiv:  forall a b c, (a | b) ∧ (b | c) -> (a |c).
```

because it is almost identical to the above mathematical statement and it will allow us to show some more tactics. The corresponding feedback is

==========
$\forall a\ b\ c : nat, a|b\ \wedge\ b|c\ \rightarrow\ a|c$

Note that Coq has correctly deduced that a, b, c are natural numbers and replaced the quantifier `forall` with $\forall$. Note also that in this form there are no hypotheses.

We fill fix the three variables with the tactics:

```
Fix an arbitrary element a.
Fix an arbitrary element b.
Fix an arbitrary element c.
```

to get

$a, b, c : nat$
$==========$
$a|b \wedge b|c \rightarrow a|c$

As before, in order to prove an implication $A \rightarrow B$ we use the tactic

```
Assume A then prove B.
```

More precisely, in this case we have

```
Assume (a | b ∧ b | c ) then prove (a | c).
```

to get

$a, b, c : nat$
$Hyp : a|b \wedge b|c$
$==========$
$a|c$

Note that hypothesis Hyp is of type $A \wedge B$. We will split this in two hypotheses with:

```
Eliminate the conjuction in hypothesis Hyp.
```

to get

$a, b, c : nat$
$Hyp0 : a|b$
$Hyp1 : b|c$
$==========$
$a|c$

We seem to have used all the tricks up our selves and so it is time to "unfold" the definitions:

```
Rewrite hypothesis Hyp0 using the definition of divides.
```

$$
\begin{array}{l}
a, b, c : nat \\
Hyp0 : \exists x : nat, b = a * x \\
Hyp1 : b|c \\
================ \\
a|c
\end{array}
$$

then

```
Rewrite hypothesis Hyp1 using the definition of divides.
```

$$
\begin{array}{l}
a, b, c : nat \\
Hyp0 : \exists x : nat, b = a * x \\
Hyp1 : \exists x : nat, c = b * x \\
================ \\
a|c
\end{array}
$$

and

```
Rewrite goal using the definition of divides.
```

$$
\begin{array}{l}
a, b, c : nat \\
Hyp0 : \exists x : nat, b = a * x \\
Hyp1 : \exists x : nat, c = b * x \\
================ \\
\exists x : nat, c = a * x
\end{array}
$$

We now pick x as in the hypothesis Hyp1, that is:

```
Fix x the existentially quantified variable in Hyp1.
```

to get

$a, b, c : nat$
$Hyp0 : \exists x : nat, b = a * x$
$x : nat$
$Hyp1 : c = b * x$
$==========$
$\exists x0 : nat, c = a * x0$

Note the variable name was changed in the goal but not in Hyp0.

We now use the newly formed Hyp1 as follows:

```
Rewrite the goal using Hyp1.
```

to get

$a, b, c : nat$
$Hyp0 : \exists x : nat, b = a * x$
$x : nat$
$Hyp1 : c = b * x$
$==========$
$\exists x0 : nat, b * x = a * x0$

Similarly we pick y as in Hyp0 and replace it in the goal

```
 Fix y the existentially quantified variable in Hyp0.
Rewrite the goal using Hyp0.
```

to get

$a, b, cy : nat$
$Hyp0 : b = a * y$
$x : nat$
$Hyp1 : c = b * x$
$==========$
$\exists x0 : nat, a * y * x = a * x0$

It is now easy to guess that $x0 = y * x$ so we write

```
Prove the existential claim is true for (y*x).
```

to obtain

$a, b, cy : nat$
$Hyp0 : b = a * y$
$x : nat$
$Hyp1 : c = b * x$
$==========$
$a * y * x = a * (y * x)$

which can be proved by

```
True by arithmetic properties.
```

the total proof is

```
 Definition divides (a b:nat) := exists x:nat, b = a*x.
Notation " a | b " := (divides a b) (at level 10).
Theorem refldiv:forall a b c, (a | b) ∧ (b | c) -> (a |c).
Fix an arbitrary element a.
Fix an arbitrary element b.
Fix an arbitrary element c.  Assume (a | b ∧ b | c ) then prove (a |
c).
Eliminate the conjuction in hypothesis Hyp.
Rewrite hypothesis Hyp0 using the definition of divides.
Rewrite hypothesis Hyp1 using the definition of divides.
Rewrite goal using the definition of divides.
Fix x the existentially quantified variable in Hyp1.
Rewrite the goal using Hyp1.
Fix y the existentially quantified variable in Hyp0.
Rewrite the goal using Hyp0.
Prove the existential claim is true for (y*x).
True by arithmetic properties.
```

Note that one could use a slightly shorter version of this theorem:

```
Theorem refldiv (a b c:nat):  (a | b) ∧ (b | c) -> (a |c).
Rewrite goal using the definition of divides.
Assume ((∃x : nat, b = a∗x)∧(∃x : nat, c = b∗x)) then prove (∃x : nat, c = a∗x).
Eliminate the conjuction in hypothesis Hyp.
Fix x the existentially quantified variable in Hyp1.
Rewrite the goal using Hyp1.
Fix y the existentially quantified variable in Hyp0.
Rewrite the goal using Hyp0.
Prove the existential claim is true for (y*x).
True by arithmetic properties.
```

Note also that if you save the latex form of the proof you will obtain the following:

Definition 1 (divides) $divides\,(a\,b : nat) := x : nat,\ b = a * x.$
Theorem 1 (refldiv) $\forall\, a\, b\, c,\ (a \mid b) \wedge (b \mid c) \Rightarrow (a \mid c).$
Proof: In order to show

$$\forall abc : nat, a|b \wedge b|c \Rightarrow a|c$$

we pick an arbitrary

$$a$$

and show

$$\forall bc : nat, a|b \wedge b|c \Rightarrow a|c.$$

In order to show

$$\forall bc : nat, a|b \wedge b|c \Rightarrow a|c$$

we pick an arbitrary

$$b$$

and show

$$\forall c : nat, a|b \wedge b|c \Rightarrow a|c.$$

In order to show

$$\forall c : nat, a|b \wedge b|c \Rightarrow a|c$$

we pick an arbitrary

$$c$$

and show

$$a|b \wedge b|c \Rightarrow a|c.$$

We will assume

$$Hyp : a|b \wedge b|c$$

and show

$$a|c.$$

Since we know

$$Hyp : a|b \wedge b|c$$

we also know

$$Hyp0 : a|bHyp1 : b|c.$$

We use the definition of

$$divides$$

in

$$Hyp0$$

to obtain

$$Hyp0 : \exists x : nat, b = a * x$$

We use the definition of

$$divides$$

in

$$Hyp1$$

to obtain

$$Hyp1 : \exists x : nat, c = b * x$$

Rewriting the definition of

$$divides$$

in our conclusion

$$a|c$$

, we now need to show

$$\exists x : nat, c = a * x.$$

We choose a variable

$$x$$

in

$$Hyp1$$

to obtain

$$x : natHyp1 : c = b * x.$$

We rewrite the goal using

$$Hyp1$$

to obtain

$$\exists x0 : nat, b * x = a * x0.$$

We choose a variable

$$y$$

in

$$Hyp0$$

to obtain

$$a, b, c, y : natHyp0 : b = a * y.$$

We rewrite the goal using

$$Hyp0$$

to obtain

$$\exists x0 : nat, a * y * x = a * x0.$$

We shall prove

$$\exists x0 : nat, a * y * x = a * x0$$

by showing

$$a * y * x = a * (y * x).$$

This follows immediately from arithmetic.
This is done Now

$$a * y * x = a * (y * x)$$

means that

$$\exists x0 : nat, a * y * x = a * x0.$$

We have now proved

$$\exists x0 : nat, a * y * x = a * x0$$

and so

$$\exists x0 : nat, b * x = a * x0$$

follows. and so we have proved

$$\exists x0 : nat, b * x = a * x0.$$

We have now proved

$$\exists x0 : nat, b * x = a * x0$$

and so

$$\exists x0 : nat, c = a * x0$$

follows. and so we have proved

$$\exists x : nat, c = a * x.$$

Therefore we have showed

$$\exists x : nat, c = a * x$$

and so

$$a|c.$$

therefore we have

$$a|c.$$

therefore we have

$$a|c.$$

We are now done with

$$a|c.$$

We have now showed that if

$$Hyp : a|b \wedge b|c$$

then

$$a|c$$

a proof of

$$a|b \wedge b|c \Rightarrow a|c.$$

Since

$$c$$

was arbitrary this shows

$$\forall c : nat, a|b \wedge b|c \Rightarrow a|c.$$

Since

$$b$$

was arbitrary this shows

$$\forall bc : nat, a|b \wedge b|c \Rightarrow a|c.$$

Since

$$a$$

was arbitrary this shows

$$\forall abc : nat, a|b \wedge b|c \Rightarrow a|c.$$