CS3202, LSV
Semester 2 2007,
Tutorial 1

James McKinna& Stéphane Lengrand

February 13, 2007

Submission Submission should be via MMS in the form of a single COQ vernacular file, named "`CS3202tutorial1.v`". An initial template version of this file has been prepared for you. It should be added to, and regularly saved, during the course of your lab work, and then uploaded to MMS.

Deadline Friday 16th February 2007, by MIDNIGHT.

Credit This tutorial contributes to the 10% of the overall coursework grade allocated for tutorials.

Rubric You are reminded of the University's rules governing Academic Fraud. You may of course work with colleagues in discussing how to go about solving these problems, but any work which you submit MUST be your own, except where you EXPLICITLY reference the work of others.

Task 1: Using Coq

1. (a) login to a JH lab machine
   (b) fire up WWW browser (otherwise the built-in WWW help subsystem of `coqide` will not work. . .)
   (c) get a terminal window
   (d) type `coqide` at the prompt
   (e) check browser-based help is working, check the online tutorial and the reference manual.
2. (a) download `CS3202tutorial1.v` and `CS3202.v` from Studres/Tutorials
   (b) open `CS3202tutorial1.v` in `coqide`
   (c) check the effects of the compilation buttons and observe `coqide`'s answers in the bottom-right window.
   (d) What is the role of the dot `.` ?

The concrete syntax for the connectives is
`/\` for $\wedge$ (e.g. `A/\B`), `\/` for $\wedge$ (e.g. `A\/B`), `->` for $\Rightarrow$ (e.g. `A->B`),
and, in these tutorials, `bot` for $\bot$ and `not` for $\neg$ (e.g. `not A`).

The constructors for proof-trees are, in these tutorials,

$\wedge$-introduction: $\dfrac{A \quad B}{A \wedge B}$ `and_i A B`

$\wedge$-elimination: $\dfrac{A \wedge B}{A}$ `and_el A B` $\dfrac{A \wedge B}{B}$ `and_er A B`

$\vee$-introduction: $\dfrac{A}{A \vee B}$ `or_il A B` $\dfrac{B}{A \vee B}$ `or_ir A B`

$\vee$-elimination: $\dfrac{A \vee B \quad \begin{array}{c}[A]\\ \vdots\\ C\end{array} \quad \begin{array}{c}[B]\\ \vdots\\ C\end{array}}{C}$ `or_e A B C`

$\Rightarrow$-introduction: $\dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \Rightarrow B}$ `imp_i A B`

$\Rightarrow$-elimination: $\dfrac{A \Rightarrow B \quad A}{B}$ `imp_e A B`

$\bot$-elimination: $\dfrac{\bot}{A}$ `bot_e A`

$\neg$-introduction: $\dfrac{\begin{array}{c}[A]\\ \vdots\\ \bot\end{array}}{\neg A}$ `not_i A`

$\neg$-elimination: $\dfrac{A \quad \neg A}{\bot}$ `not_e A`

copy: $\dfrac{A}{A}$ `copy A`

Discharging the leaves called `x` and labelled with `A` in a proof `t` is written

```
fun x:A => t
```

Task 2: Proving

- Check (and understand) the proof of $A \vdash A$.

  I assume the hypotheses of the syntactic entailment relation by giving them names (aka variables). Here, there is just the assumption `A`, which I name `H`.

  ```
  Hypothesis H:A.
  ```

  Now I give a proof-term for the tree concluding `A` from these assumptions. It can thus use the variables declared for the assumptions, and guess what it is in this case...

  ```
  Check H.
  ```

- Check (and understand) the proof of $A \wedge B \vdash B \wedge A$

  Here, there is just the assumption `A/\B`, which I name `H_AB`.

  ```
  Hypothesis H_AB : A /\ B.
  ```

Now I give a proof-term for the tree concluding `B/\A` from the assumption. It can thus use `H_AB`. Let's start with a proof-term for a tree concluding `A`:

```
Check and_el A B H_AB.
```

Let's continue with a proof-term for a tree concluding `B`:

```
Check and_er A B H_AB.
```

Here is my proof-term for a tree concluding `B/\A`:

```
Check and_i B A (and_er A B H_AB) (and_el A B H_AB) .
```

- Check (and understand) the proof of $A \vee B \vdash B \vee A$

  Here, there is just the assumption `A\/B`, which I name `H0`.

  ```
  Hypothesis H0:A\/B.
  ```

  Now I give a proof-term for the tree concluding `B\/A` from the assumption. It can thus use `H0`. Let's start with a proof-term for a tree concluding `B\/A` under the local assumption of `A` (called `H1`) that I discharge:

  ```
  Check fun H1:A => or_ir B A H1.
  ```

  Note that `H1` is not visible after the discharge. `Check H1.` fails! Let's continue with a proof-term for a tree concluding `B\/A` under the local assumption of `B` (called `H2`) that I discharge:

  ```
  Check fun H2:B => or_il B A H2.
  ```

  Again, the variable `H2` is not visible (and in fact I could have called it `H1` without clash.) Now here is my proof-term for a tree concluding `B\/A`:

  ```
  Check
    or_e A B (B\/A)
        H0 (fun H1:A => or_ir B A H1) (fun H2:B => or_il B A H2).
  ```

- Prove $(A \wedge B) \wedge C \vdash A \wedge (B \wedge C)$

  Assume `(A\wedge B)\wedge C\vdash A` by giving it the name `H1`:

  ```
  Hypothesis H1:(A/\B)/\C.
  ```

  Now it's your job to find a proof-term for `A/\(B/\C)` using `H1`!

- Prove $A, (A \Rightarrow B), (B \Rightarrow C) \vdash C$
- Prove $(A \Rightarrow B), (B \Rightarrow C) \vdash A \Rightarrow C$
- Prove $(A \vee B) \vee C \vdash A \vee (B \vee C)$
- Prove $A \Rightarrow \bot \vdash (\neg A)$
- Prove $(\neg A) \vdash A \Rightarrow \bot$
- Prove? $\vdash (((A \Rightarrow B) \Rightarrow B) \Rightarrow A$
- Check whether it is a tautology. Conclusion?