Quick reference for frama-c/Jessie exercises
Summer school 2012

Pierre Courtieu

March 26, 2013

Login in

1. Open a terminal and type:

   ```
   ssh -X eleveX@chou.cnam.fr
   ```

   where `eleveX` is the login that will be given to you during the session.

2. type the password (also given during the session, you may have to answer `yes` first to a question from `ssh`).

   Download the exercises directory, extract it and go into it with:

   ```
   wget http://cedric.cnam.fr/~courtiep/downloads/exercises-jessie.tgz
   tar xvfz exercises-jessie.tgz
   cd exercises-jessie
   ```

3. start an editor:

   ```
   gedit &
   ```

   You can also start `emacs` or `vim` if you prefer (if you don't know these editors, prefer `gedit`).

4. Open a new X terminal from the current terminal

   ```
   xterm &
   ```

   and minimize the first terminal, you will work on the new terminal (this avoids closing the session on the server by closing the initial terminal by mistake).

By now you should be ready to use frama-c.

Before you call the tool for the first time in your account you need to call this command (once and for all):

```
why-config
```

Calling frama-c

Workflow The typical workflow with jessie is the following:

1. edit your C file with your text editor (see above), write code and/or assertions;
2. launch jessie on the file (don't close the text editor):

   ```
   frama-c -jessie file.c &
   ```

   (a) if compilation fails, then solve errors in the files and re-launch jessie;

   (b) otherwise a graphical interface pops up and allows you to launch automatic provers on the proof obligations generated by jessie, launch a prover (click on its column title) and see if all proof obligations are solved (green circle).

      i. if yes, your program is correct with respect to your annotations;

      ii. otherwise there are three possibilities (or any combination of them):

         A. your code is wrong;

         B. your annotations are wrong;

         C. code and annotations are correct but the prover is unable to prove annotations. You may need to help the prover by adding logical information to your annotations. You may try another prover to see if it manages to prove the unproven obligations;

      In any case, once you figured out how to fix the problem, close the graphical interface (CTRL-q) and go to step 1 (in the terminal hitting the "up" arrow allows to call previously called commands). You cannot edit the file from the graphical interface.

Exercises

Quick Reference

`\result`	refers to the result of the current function. Only in function contracts.
`\old(e)`	refers to the initial value of expression `e`. Only in function contracts.
`requires e;`	states the precondition of the current function. Only in function contracts.
`ensures e;`	states the post-condition of the current function. Only in function contracts.
L:;	C label. can only appear in regular C code. See next line
`\at(e,L)`	refers to the value of expression `e` at label `L` (see previous line)
`\forall` **int** `i; e`	$\forall i, e$
`\exists` **int** `i; e`	$\exists i, e$
`e1 ==> e2`	`e1` $\rightarrow$ `e2` (that is `e1` "implies" `e2`).
`e1 == e2`	Equality, `e1 = e2` has no meaning in assertions
`e1 && e2`	and operator
`e1 || e2`	or operator
`!e1`	negation operator
`loop invariant e;`	declares expression `e` as an invariant of the loop immediately after this comment. You can have several invariants declared for the same loop, they will be "anded".
`loop variant e;`	declares expression `e` as the variant of the loop immediately after this comment.

#pragma `JessieIntegerModel(math)`	tells jessie to consider integer as $\mathbb{N}$ instead of $[-2^{31}; 2^{31}-1]$. We will always use this pragma.
#pragma `JessieTerminationPolicy(never)`	tells jessie to not ask for loop variants. We will always use this pragma.

Traps

Parenthesis

Be careful to parenthesize correctly. For example:

- `\forall int i; e ==> f` is not equivalent to `(\forall int i; e) ==> f`

Quantification

The mathematical notation is generally informal. One write for example: $\forall i \leq n, P(n)$. This is not a well formed formula. What is the correct one? You may be puzzled at first but here are the correct formal properties.

informal property	formal property	`jessie` syntax
$\forall i \leq n, P$	$\forall i, (i \leq n \rightarrow P)$	`\forall int i; (i <= n ==> P)`
$\exists i \leq n, P$	$\exists i, (i \leq n \wedge P)$	`\exists int i; (i <= n && P)`
if e then P else Q	$(e \rightarrow P) \wedge (\neg e \rightarrow Q)$	`(e ==> P) && (! e ==> Q)`

Remark: In the above table (and traditionally in logic) $P \wedge Q$ stands for P *and* Q, $P \vee Q$ stands for P *or* Q and $\neg P$ stands for *not* P.