Assignment 5
Semantics, WS 2011-2012

Prof. Dr. Gert Smolka, Dr. Chad Brown
www.ps.uni-saarland.de/courses/cl-ss11/

Read in the lecture notes:

Remark: You may use any of the tactics we used in class including *econstructor*, *congruence*, *firstorder* and *auto*. In addition, the tactic *eassumption* is helpful when the claim has an evar, but otherwise matches an assumption.

Exercise 5.1 Formulate the following equivalences as goals in Coq and prove them.

a) c; skip $\cong$ c
b) if false then c_1 else c_2 $\cong$ c_2
c) while false do c $\cong$ skip
d) while b do c $\cong$ if b then c; while b do c else skip

Exercise 5.2 Use Coq to prove that the approximation relation $\lesssim$ is reflexive and transitive.

Exercise 5.3 Use Coq to prove that program equivalence $\cong$ is reflexive, symmetric and transitive.

Exercise 5.4 Use Coq to prove that if $c_1 \lesssim c_1'$ and $c_2 \lesssim c_2'$, then $c_1; c_2 \lesssim c_1'; c_2'$.

Exercise 5.5 Use Coq to prove that if $c_1 \lesssim c_1'$ and $c_2 \lesssim c_2'$, then if b then c_1 else c_2 $\lesssim$ if b then c_1' else c_2'.

Exercise 5.6 Assume we know the relational semantics is functional.

```
Lemma ceval_functional c st st1 st2 :
c / st || st1 -> c / st || st2 -> st1 = st2.
```

a) Prove if skip $\lesssim$ c, then skip $\cong$ c.
b) Prove if $c \lesssim c'$ and c terminates on all states, then $c \cong c'$.

Exercise 5.7 Assume we have a type of states, an abstract boolean predicate b on states and an abstract function c on states.

```
Variable state : Type.
Variable b : state -> bool.
Variable c : state -> state.
```

Suppose we define a relation *rel* on states by the following two rules.

$$\frac{b\sigma = \mathit{false}}{\mathit{rel}\ \sigma\ \sigma} \qquad\qquad \frac{b\sigma = \mathit{true} \qquad \mathit{rel}\ (c\ \sigma)\ \sigma'}{\mathit{rel}\ \sigma\ \sigma'}$$

Remark: You should be able to do part (a) of this problem with no trouble. Parts (b) - (d) are more challenging. To do a case analysis on the result of a non-variable term t you may write

```
remember t as x. destruct x.
```

instead of

```
destruct t.
```

a) Define a step function *step:nat −> state −> option state* so that the propositi-on

```
forall s s', rel s s' <-> exists i, step i s = Some s'.
```

will be provable.

b) Prove

```
Lemma agree s s' :
rel s s' <-> exists i, step i s = Some s'.
```

c) Prove

```
Lemma monotone i s s' :
step i s = Some s' -> step (S i) s = Some s'.
```

d) Prove

```
Lemma functional s s' s'' :
rel s s' -> rel s s'' -> s'=s''.
```