COMS22201: Language Engineering

Lab Exercises - Week 23 - Questions

25/04/2016, csxor@bristol.ac.uk

This worksheet will give you practice in using, extending and reasoning about the natural operational semantics.

1. Notice that in the lecture slides we defined three natural semantic rules for conjunction (slide 5) of which the last two are *not* mutually exclusive.

   (a) Use this observation to show that there exist configurations which have more than one possible derivation tree.

   (b) But explain why the semantics still satisfies the formal definition of determinism (slide 10).

   (c) Give an equivalent semantics which ensures each configuration has exactly one proof tree.

2. Is the natural operational semantics compositional?

3. Explain which of the following proof techniques can be used with the natural operational semantics: (a) structural induction, (b) induction on the shape of derivation trees, and/or (c) induction on the length of derivation sequences.

4. Prove that the natural semantics of Boolean expressions is deterministic (i.e. show, if it holds that $\langle b, \sigma\rangle \to v$ and $\langle b, \sigma\rangle \to v'$, then it holds that $v = v'$ for all $b \in Bexp, \sigma \in State, v, v' \in \{tt, ff\}$). You may assume the semantics of arithmetics is deterministic (or you may prove that too).

5. Use the natural operational semantics to evaluate the following program (for computing factorials) from a state s which maps variable x to integer 3 (cf. ex 2.2 on p23-5 of the textbook):

   ```
   y:=1; while ¬(x=1) do (y:=y*x; x:=x-1)
   ```

6. Use the natural operational semantics to evaluate the following program (for computing moduli) from a state t which maps x to 17 and y to 5 (cf. ex 2.3 on p25 of the textbook):

   ```
   z:=0; while y≤x do (z:=z+1; x:=x-y)
   ```

7. Prove that $S_1;(S_2;S_3)$ is equivalent to $(S_1;S_2);S_3$ under the natural operational semantics (cf. ex 2.6 on p28).

8. Write a natural operational semantics for a loop construct of the form `repeat S until b` in a way that does not rely on the existence of any other loop construct (cf. ex 2.7 on p28).

9. Prove that `repeat S until b` is equivalent to

   `S; if b then skip else repeat S until b`

   (cf. ex 2.7 on p28 again).

10. Prove `repeat S until b` is equivalent to `S; while ¬b do S`

    (cf. ex 2.10 on p30)

11. Write a natural operational semantics for a loop construct of the form `for x := a1 to a2 do S` in a way that does not rely on the existence of any other loop construct (cf. ex 2.8 on p28).

    Note that there are numerous possible interpretations of this construct. When $a1$ and $a2$ are numerals and x is not used outside the loop or modified within the loop, then almost everyone can agree on its meaning. But conventions differ wildly in almost all other cases. This construct is often intended to represent a loop where the number of iterations is known in advance, in which case $a1$ and $a2$ should be evaluated once and for all, at the very beginning.

    Should the loop be equivalent to `x:=a1; while x`$\leq$`a2 do S; x:=x+1` or not? Should the loop variable be defined at the end or not? It will be very interesting to discuss your solutions at the tutorial!

12. Write a natural operational semantic evaluator for While in Haskell.