Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Simon Wimmer

WS 2016/17
07. 02. 2017

Semantics of Programming Languages

Exercise Sheet 14

Exercise 14.1 Small-Step Semantics for Parallel Execution

Consider a variant of IMP with parallel execution. That is, we add an operator $c_1 \parallel c_2$ for combining commands c_1 and c_2 in parallel, and add the following rules to the small-step semantics:

- ParLeft: $(c_1, s) \rightarrow (c_1', s') \Longrightarrow (c_1 \parallel c_2, s) \rightarrow (c_1' \parallel c_2, s')$
- ParRight: $(c_2, s) \rightarrow (c_2', s') \Longrightarrow (c_1 \parallel c_2, s) \rightarrow (c_1 \parallel c_2', s')$
- ParSkip: $(SKIP \parallel c, s) \rightarrow (c, s)$

We will denote the set of variables of a command c by $vars\ c$ and write $s_1 \sim_S s_2$ if:

$$\forall x \in S.\ s_1\ x = s_2\ x$$

We want to show that $c_1 \parallel c_2$ can be sequentialized if $vars\ c_1 \cap vars\ c_2 = \emptyset$.

Question 1 For now, you may assume the following fact:

$$\begin{aligned} &(c_1, s') \rightarrow^* (SKIP, s'') \Longrightarrow (c_2, s) \rightarrow^* (c_2', s') \\ &\Longrightarrow vars\ c_1 \cap vars\ c_2 = \emptyset \\ &\Longrightarrow \exists t.\ (c_1, s) \rightarrow^* (SKIP, t) \wedge (c_2, t) \rightarrow^* (c_2', s'') \end{aligned} \tag{1}$$

Prove:

$$\begin{aligned} &(c_1 \parallel c_2, s) \rightarrow^* (SKIP, t) \\ &\Longrightarrow vars\ c_1 \cap vars\ c_2 = \emptyset \\ &\Longrightarrow (c_1;; c_2, s) \rightarrow^* (SKIP, t) \end{aligned} \tag{2}$$

Hint: Start with an induction on the transitive closure *, and use a case analysis on the small-step semantics in the induction step.

Question 2 Now show (1). You may use the following facts:

$$(c, s) \rightarrow^* (c', s') \Longrightarrow s \sim_{\overline{vars\ c}} s' \tag{3}$$

$$(c, s) \rightarrow^* (c', s') \wedge s \sim_{vars\ c} t \Longrightarrow \exists t'.\ (c, t) \rightarrow^* (c', t') \wedge s' \sim_{vars\ c} t' \tag{4}$$

where $\overline{S}$ denotes the complement of S. *Hint:* There is a direct proof, you do not need induction.

Exercise 14.2 Parity analysis

Consider the following IMP program:

```
r := 11;
a := 11 + 11;
WHILE b DO
  r := r + 1;
  a := a - 2;
r := a + 1
```

Add annotations for parity analysis to this program, and iterate on it the *step′* function until a fixed point is reached. (More precisely, let C be the annotated program; you need to compute $(step'\ \top)^0\ C$, $(step'\ \top)^1\ C$, $(step'\ \top)^2\ C$, etc.). Document the results of each iteration in a table.