Technische Universität München
Fakultät für Informatik
Dr. Peter Lammich
Simon Wimmer

WS 2018/19
13.11.2018

Semantics of Programming Languages

Exercise Sheet 5

Homework 5.1 Control Flow Graphs

Submission until Tuesday, November 20, 2018, 10:00am.

In this homework, we want to study the concept of *control flow graphs* for IMP and connect it to the small-step semantics. To get started, we first introduce the concept of a *labeled transition system* (LTS). An LTS is a directed graph with edge labels. Similarly, to our previous model of graphs, we represent an LTS as a predicate for its edges:

type_synonym $('q,'l)$ *lts* = "$'q \Rightarrow 'l \Rightarrow 'q \Rightarrow$ *bool*"

A word from *source node* u to *target node* v is the sequence of edge labels one encounters when moving from u to v in the LTS. Analogously to *is_path* for graphs, we define a predicate *word*, such that *word* δ u w v holds iff w is a word from u to v:

inductive *word* :: "$('q,'l)$ *lts* $\Rightarrow 'q \Rightarrow 'l$ *list* $\Rightarrow 'q \Rightarrow$ *bool*" **for** δ **where**
  empty: "*word* δ q [] q"
| *prepend*: "$[\![\delta\ q\ l\ p;$ *word* $\delta\ p\ ls\ r]\!] \Longrightarrow$ *word* $\delta\ q\ (l\#ls)\ r$"

A control flow graph is a labeled transition system, where the edges are labeled with *effects*. An effect is a partial function on states, returning *None* when the test for a Boolean condition fails:

type_synonym *effect* = "*state* $\rightharpoonup$ *state*"
type_synonym $'q$ *cfg* = "$('q,$*effect*$)$ *lts*"

Note that $'a \rightharpoonup 'b$ is a syntactic abbreviation for $'a \Rightarrow 'b$ *option*.

Intuitively, the control flow graph is executed by following a path and applying the effects of the actions to the state. Lift effects to paths. Only paths where all tests succeed shall yield a result $\neq$ *None*.

fun *eff_list* :: "*effect list* $\Rightarrow$ *state* $\rightharpoonup$ *state*" **where**

The control flow graph of a WHILE-Program can be defined over nodes that are commands. Complete the following definition. (*Hint:* Have a look at the small-step semantics first)

inductive *cfg* :: "*com cfg*" **where**
  cfg_assign: "*cfg* $(x ::= a)$ $(\lambda s.$ *Some* $(s(x{:=}$*aval* $a\ s)))$ $(SKIP)$"

| *cfg_Seq2*: "*cfg c1 e c1'* $\Longrightarrow$ *cfg* (*c1*;;*c2*) *e* (*c1'*;;*c2*)"

Prove that the effects of paths in the CFG match the small-step semantics:

theorem *eq_path*: "$(c,s) \rightarrow* (c',s') \longleftrightarrow (\exists \pi.$ *word cfg c* π $c' \wedge$ *eff_list* π $s =$ *Some* $s')$"

Prove the theorem for a single step first:

theorem *eq_step*: "$(c,s) \rightarrow (c',s') \longleftrightarrow (\exists e.$ *cfg c e* $c' \wedge e\ s =$ *Some* $s')$"

Now prove the main theorem:

theorem *eq_path*: "$(c,s) \rightarrow* (c',s') \longleftrightarrow (\exists \pi.$ *word cfg c* π $c' \wedge$ *eff_list* π $s =$ *Some* $s')$"