Program semantics and verification 2021/22
Exam 4/02/2022, assignment 1 (operational semantics)

Consider the extension of TINY with the following forms of statements (E ranges over arithmetic expressions, B over Boolean expressions, and x over variables):

$$\begin{array}{rcl} S & ::= & \texttt{skip} \mid S_1;S_2 \mid \texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ endif} \mid \texttt{while } B \texttt{ do } S \texttt{ od} \mid \\ & & x := \{S\}\ E \mid \texttt{reject} \mid \texttt{resolve } E \end{array}$$

The execution of the new generalised form of assignment $x := \{S\}\ E_0$ proceeds by executing the statement S and then:

i) if `resolve` E is encountered while executing S, the execution of S is interrupted, the current state is modified by assigning the value of E in this state to variable x, and the execution of the assignment ends in this so modified state.

ii) If `reject` is encountered while executing S, the execution of S is interrupted, the state in which the execution of the assignment started is modified by assigning 0 to variable x, and the execution of the assignment ends in this so modified state.

iii) If the execution of S ends without encountering either `resolve` E or `reject`, the state after the execution of S is modified by assigning the value of E_0 in this state to variable x, and the execution of the assignment ends in this so modified state.

The statements of the forms `resolve` E and `reject` outside the new generalised assignments block the execution. The statements of other forms work as usual.

Examples: Consider a state $s : \texttt{Var} \to \mathbb{Z}$ such that $s(x) = 1$, $s(y) = 2$. (In the examples below additional standard operators, like $<$, $>$ and `mod`, are used with the usual meaning.)

i) Let S_1 be the following statement:

$$\begin{array}{l} x := \{\texttt{while } x < 10 \texttt{ do} \\ \quad\quad x := x + 7; \\ \quad\quad y := y + 2; \\ \quad \texttt{od}; \\ \quad \texttt{resolve } (x + y)\}\ 444 \end{array}$$

Then S_1 executed in the initial state s stops in $s[x \mapsto 21][y \mapsto 6]$.

ii) Let S_2 be the following statement:

$$x := \{y := x * y + 5\}\ 36$$

Then S_2 executed in the initial state s stops in $s[x \mapsto 36][y \mapsto 7]$.

iii) Let S_3 be the following statement:

$$\begin{array}{l} x := \{\texttt{while } x > 0 \texttt{ do} \\ \quad\quad \texttt{if } x \texttt{ mod } 5 = 3 \texttt{ then reject else skip endif}; \\ \quad\quad \texttt{if } x \texttt{ mod } 5 = 0 \texttt{ then resolve } x * y \texttt{ else skip endif}; \\ \quad\quad x := x + 1; \\ \quad\quad y := y + 2 \\ \quad \texttt{od}\}\ 36 \end{array}$$

Then S_3 executed in the initial state s stops in $s[x \mapsto 0]$.

iv) Let S_4 be the following statement:

$$
\begin{array}{l}
x := \{\texttt{while } \neg(x = 0) \texttt{ do} \\
\quad x := x + 1; \\
\quad x := \{\texttt{while } y > 0 \texttt{ do} \\
\qquad\qquad y := y + 4; \\
\qquad\qquad \texttt{if } x + y > 30 \texttt{ then reject else resolve } y * x \texttt{ endif} \\
\qquad\quad \texttt{od } \} \ 10 \\
\texttt{od}\} \ 55
\end{array}
$$

Then S_4 executed in the initial state s stops in $s[x \mapsto 55][y \mapsto 10]$.

Define small-step operational semantics for this new language: define the set of configurations, the set of terminal configurations and inference rules for the generalised assignments, `resolve` E and `reject`. Indicate if and how the standard rules for the statements of other forms have to be modified (or write them out). The semantics of expressions (and Boolean expressions) may be assumed to be given.