Semantyka i weryfikacja programów

semestr zimowy 2004/2005

Zadanie 1. Napisać semantykę naturalną (duże kroki) języka o następującej gramatyce (TINY z dodanymi dwiema instrukcjami):

$$
\begin{array}{rcl}
N \in \mathbf{Num} & ::= & 0 \mid 1 \mid 2 \mid \ldots \\
x \in \mathbf{Var} & ::= & x_1 \mid x_2 \mid \ldots \\
e \in \mathbf{Exp} & ::= & N \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2 \\
b \in \mathbf{BExp} & ::= & \texttt{true} \mid \texttt{false} \mid e_1 \leq e_2 \mid \neg b \mid b_1 \wedge b_2 \\
S \in \mathbf{Stmt} & ::= & x := e \mid \texttt{skip} \mid S_1; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \\
 & & \mid \texttt{while } b \texttt{ do } S \mid \texttt{for } x = e_1 \texttt{ to } e_2 \texttt{ try } S \texttt{ else } S' \mid \texttt{fail}.
\end{array}
$$

Znaczenie powyższych konstrukcji (poza `for` $x = e_1$ `to` e_2 `try` S `else` S' i `fail`) jest standardowe (takie jak na wykładzie). Znaczenie instrukcji `for` $x = e_1$ `to` e_2 `try` S `else` S' jest następujące: najpierw jest obliczana wartość wyrażenia e_1 i podstawiana na zmienną x. Następnie wykonywana jest instrukcja S (z odpowiednio podstawioną wartością na zmienną x, oczywiście instrukcję S wykonujemy tylko wtedy, gdy $x \leq e_2$). Jeśli w S nie napotkana zostanie instrukcja `fail` to przywracamy starą wartość zmiennej x (tzn. sprzed wykonania instrukcji `for`) i przechodzimy do dalszej części programu. W przeciwnym wypadku przywracamy stan sprzed wykonania instrukcji `for` (zachowując oczywiście wartość zmiennej x), zwiększamy wartość zmiennej x o 1 i powtarzamy procedurę opisaną powyżej tak długo aż nie napotkamy instrukcji `fail` lub $x > e_2$ (wartość wyrażenia e_2 wyliczamy raz, w stanie sprzed wykonania instrukcji `for`). Jeśli $x > e_2$ (zdarza się to wtedy, gdy napotkana została instrukcja `fail` w instrukcji S dla $x = e_2$) to wykonujemy S' w stanie sprzed instrukcji `for` (w szczególności wartość zmiennej x na początku wykonywania S' powinna być taka jaka była przed wykonaniem instrukcji `for`). Instrukcja `fail` poza pętlą `for` powinna spowodować zakończenie programu w stanie "porażki".

Przykład:

$x := 0; y := 1;$
`for` $x = 1$ `to` 5 `try`
  $y := y + 1;$
  `if` $x \leq 4$ `then fail else` $z := x$
`else skip`

Wartością zmiennej x po wykonaniu programu powinno być $x = 0$, $y = 2$, natomiast zmiennej $z = 5$.