Programowanie (M)

Lista zadań nr 4

Na ćwiczenia 3 i 5 kwietnia 2012

Zadanie 1 (1 pkt). Udowodnij, że

1. jeżeli $\langle S_1; S_2, s\rangle \Rightarrow^k s'$, to istnieją k_1 i k_2 takie, że $\langle S_1, s\rangle \Rightarrow^{k_1} s''$, $\langle S_2, s''\rangle \Rightarrow^{k_2} s'$ oraz $k = k_1 + k_2$.
2. jeżeli $\langle S_1, s\rangle \Rightarrow^{k_1} s''$ i $\langle S_2, s''\rangle \Rightarrow^{k_2} s'$, to $\langle S_1; S_2, s\rangle \Rightarrow^{k_1+k_2} s'$.

Zadanie 2 (1 pkt). Zdefiniuj strukturalną semantykę operacyjną wyrażeń arytmetycznych oraz logicznych języka **While**. Zmodyfikuj następnie strukturalną semantykę operacyjną instrukcji języka **While** tak, by odwoływała się do zadanej przez Ciebie semantyki operacyjnej wyrażeń. Udowodnij zgodność tak zadanej strukturalnej semantyki operacyjnej z oryginalną strukturalną semantyką operacyjną języka **While**, odwołującą się do semantyki denotacyjnej wyrażeń.

W oparciu o tak zdefiniowaną strukturalną semantykę operacyjną języka **While**, zaimplementuj jego interpreter w Prologu.

Zadanie 3 (1 pkt). Rozważmy instrukcję lokalnie definiującą zmienną:

$$\texttt{let}\, x \;=\; a\, \texttt{in}\, S.$$

Intuicyjna semantyka tej instrukcji jest następująca. Najpierw liczona jest wartość wyrażenia a, która zostaje przypisana zmiennej x. W tak zmodyfikowanym stanie wykonywana jest instrukcja S. Po zakończeniu jej wykonania zmiennej x przywracana jest jej pierwotna wartość.

1. Podaj semantykę naturalną instrukcji $\texttt{let}\, x \;=\; a\, \texttt{in}\, S$.
2. Podaj semantykę małych kroków instrukcji $\texttt{let}\, x \;=\; a\, \texttt{in}\, S$. Postaraj się nie wprowadzać żadnych konstrukcji pomocniczych.

Zadanie 4 (1 pkt). Podaj strukturalną semantykę operacyjną instrukcji:

1. `repeat` S `until` b,
2. `for` $x := a_1$ `to` a_2 `do` S.

Zadanie 5 (1 pkt). Rozważmy język **While** rozszerzony o instrukcje niedeterministycznego wyboru `or`, awaryjnego przerwania wykonania programu `abort` oraz wypisania wartości wyrażenia arytmetycznego na wyjście `write`:

$$S ::= \cdots \mid \texttt{or} \mid \texttt{abort} \mid \texttt{write}\; a$$

Zdefiniuj strukturalną semantykę operacyjną tego języka, przy założeniu, że relacja przejścia między stanami obliczeń jest etykietowana elementami zbioru $Lab = \{n \mid n \in \mathbb{Z}\} \cup \{\epsilon\}$, gdzie $\gamma \stackrel{n}{\Rightarrow} \gamma'$

oznacza jeden krok obliczeń, w którym nastąpiło wysłanie liczby n na wyjście, a $\gamma \stackrel{\epsilon}{\Rightarrow} \gamma'$ oznacza tzw. ciche przejście, czyli jeden krok obliczeń bez obserwowanych efektów. Na bazie zadanej przez Ciebie relacji przejścia między stanami, kiedy dwie instrukcje powinny zostać uznane za równoważne? Zdefiniuj odpowiednio funkcję semantyczną dla rozważanego języka.

Czy masz pomysł na zdefiniowanie semantyki instrukcji `read` x, wczytania z wejścia liczby całkowitej do zmiennej x?

Zadanie 6 (1 pkt). Rozważmy język **While1** jako rozszerzenie języka **While** o wyrażenia warunkowe znane z języka **C**:

$$\begin{array}{rcl} a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid b \;?\; a_1 : a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid b \;?\; b_1 : b_2 \\ S & ::= & \texttt{skip} \mid x{:=}a \mid S_1;S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S \end{array}$$

Pokaż, że wyrażenia te można uznać za cukier syntaktyczny w języku **While**. W tym celu wykonaj następujące kroki, pamiętając o zadaniu formalnej semantyki (w ywbranym przez Ciebie formacie) każdemu z rozważanych języków. Naszkicuj dowód poprawności każdej transformacji.

1. Zdefiniuj transformację wyrażeń arytmetycznych i logicznych zastępującą wyrażenia $b \;?\; b_0 : b_1$ semantycznie równoważnymi im wyrażeniami logicznymi niezawierającymi logicznych wyrażeń warunkowych (język **While2**):

$$\begin{array}{rcl} a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid b \;?\; a_1 : a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \\ S & ::= & \texttt{skip} \mid x := a \mid S_1; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S \end{array}$$

2. Zdefiniuj, zachowującą semantykę, transformację wyrażeń arytmetycznych i logicznych języka **While2** w wyrażenia, w których arytmetyczne wyrażenia warunkowe są przesunięte w kierunku korzenia drzewa rozbioru (język **While3**):

$$\begin{array}{rcl} A & ::= & b \;?\; A_1 : A_2 \mid a \\ a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid A_1 = A_2 \mid A_1 \leq A_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \\ S & ::= & \texttt{skip} \mid x := A \mid S_1; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S \end{array}$$

3. Zdefiniuj, zachowującą semantykę, transformację języka IMP3 eliminującą wyrażenia logiczne postaci $A_1 = A_2$ i $A_1 \leq A_2$ (język **While4**):

$$\begin{array}{rcl} A & ::= & b \;?\; A_1 : A_2 \mid a \\ a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \\ S & ::= & \texttt{skip} \mid x := A \mid S_1; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S \end{array}$$

4. Zdefiniuj, zachowującą semantykę, transformację języka **While4** eliminującą podstawienia postaci $x := A$ (język **While**):

$$\begin{array}{rcl} a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \\ S & ::= & \texttt{skip} \mid x := a \mid S_1; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S \end{array}$$