Semantyka i weryfikacja programów 2022/23.
Egzamin 6/02/2023, zadanie 1 (semantyka operacyjna)

Napisz semantykę operacyjną (małych lub dużych kroków — choć zapewne wygodniejsza może się okazać semantyka dużych kroków') instrukcji języka o gramatyce:

$$
\begin{aligned}
Num \ni n &::= \texttt{0} \mid \texttt{1} \mid -\texttt{1} \mid \texttt{2} \mid -\texttt{2} \mid \cdots \\
Var \ni x &::= \texttt{x} \mid \texttt{y} \mid \cdots \\
Expr \ni e &::= n \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2 \\
BExpr \ni b &::= \texttt{true} \mid \texttt{false} \mid e_1 < e_2 \mid e_1 = e_2 \mid b_1 \wedge b_2 \mid \texttt{not}\ b \\
Instr \ni I &::= x := e \mid I_1; I_2 \mid \texttt{if}\ b\ \texttt{then}\ I \mid \texttt{while}\ b\ \texttt{do}\ I \mid \\
&\qquad \texttt{attempt}\ I\ \texttt{end} \mid \texttt{protect} \mid \texttt{reject} \\
Prog \ni P &::= \texttt{prog}\ \{\texttt{attempt}\ I\ \texttt{end}\}
\end{aligned}
$$

Jest to język z mechanizmem wycofywania skutków operacji przypisania.

Wykonanie programu `prog {attempt` I `end}` polega na wykonaniu bloku `attempt` I `end`. Z kolei wykonanie bloku postaci `attempt` I `end` polega na wykonaniu instrukcji I, w której mogą się pojawić niestandardowe instrukcje `reject` i `protect`.

Instrukcja `reject` powoduje przerwanie wykonania najbliższego otaczającego bloku `attempt ... end` i wycofanie zmian wprowadzonych przez instrukcje w tym bloku (od jego początku lub od ostatniego w tym bloku wykonania instrukcji `protect`). Instrukcja `protect` powoduje "utrwalenie" stanu w miejscu jej wystąpienia dla bieżącego bloku `attempt ... end`, tak że jeśli w dalszej części bieżącego bloku `attempt ... end` wystąpi instrukcja `reject`, nie spowoduje ona wycofania do początku bloku, ale do stanu z chwili ostatniego przed `reject` wykonania w tym bloku instrukcji `protect`.

Semantyka pozostałych konstrukcji jest standardowa.
Na przykład, po wykonaniu programu:

```
prog {attempt
  x := 0;
  attempt
    x := 1;
    protect;
    x := 2;
    reject
  end
end}
```

zmienna `x` przyjmuje wartość 1. Natomiast po wykonaniu programu:

```
prog {attempt
  x := 0;
  attempt
    x := 1;
    protect;
    while x <= 10 do
    ( x := x + 1;
      protect;
      if x = 5 then reject
    )
  end
end}
```

zmienna `x` przyjmuje wartość 5.

Bloki `attempt ... end` mogą być zagnieżdżane, a skutki instrukcji `protect` i `reject` dotyczą zawsze najbliższego otaczającego bloku, zatem po wykonaniu programu

```
prog {attempt
  x := 3;
  protect;
  attempt
    x := 1;
    protect;
    x := 2;
    reject
  end;
  reject
end}
```

zmienna `x` przyjmuje wartość 3.

W rozwiązaniu należy zdefiniować zbiór konfiguracji oraz podać reguły przejścia dla programów oraz instrukcji.

Wszystkie wykorzystywane zmienne są globalne. Przyjmujemy też, że zawsze mają zainicjalizowaną wartość. Można więc w rozwiązaniu wykorzystać dziedzinę stanów $\mathbf{State} = \mathit{Var} \to \mathbf{Int}$ (nie ma konieczności podziału na środowisko i skład).

Można też przyjąć, że dane są pomocnicze funkcje semantyczne dla wyrażeń:

$\mathcal{E}$: $\mathit{Expr} \to \mathbf{State} \to \mathbf{Int}$

$\mathcal{B}$: $\mathit{BExpr} \to \mathbf{State} \to \mathbf{Bool}$

gdzie, jak zwykle, **Int** to zbiór liczb całkowitych, a $\mathbf{Bool} = \{\mathtt{tt}, \mathtt{ff}\}$.

Zbiór konfiguracji powinien zawierać konfiguracje początkowe dla programów, postaci $\langle s, P\rangle$, oraz konfiguracje końcowe postaci s (gdzie $s \in \mathbf{State}$, a $P \in \mathit{Prog}$), a także — zapewne — dodatkowe konfiguracje innych postaci.

UWAGA: w konfiguracjach nie należy dopuszczać struktur danych zawierających dowolnie dużą liczbę stanów — należy przyjąć, że każda konfiguracja może zawierać co najwyżej trzynaście stanów.

MOŻLIWE ROZWIĄZANIE

Konfiguracje Γ — suma następujących:

- $\mathbf{STATE} \times \mathbf{Prog} \ni \langle s, P\rangle$ — konfiguracje początkowe
- $\mathbf{STATE} \ni s$ — konfiguracje końcowe dla programów oraz dla instrukcji zakończonych wykonaniem `reject`
- $\mathbf{STATE} \times \mathbf{STATE} \ni \langle s_1, s_2\rangle$ — konfiguracje końcowe dla instrukcji bez `reject`, s_1 to stan do wycofania, s_2 to stan bieżący
- $\mathbf{STATE}\times\mathbf{STATE}\times\mathbf{Instr} \ni \langle s_1, s_2, I\rangle$ — kofiguracje bieżące dla instrukcji, s_1 to stan do wycofania, s_2 to stan bieżący

SEMANTYKA DUŻYCH KROKÓW

$$\frac{\langle s, s, I\rangle \rightsquigarrow s'}{\langle s, \texttt{prog } \{\texttt{attempt } I \texttt{ end}\}\rangle \rightsquigarrow\ s'}\ \textit{program} \qquad \frac{\langle s, s, I\rangle \rightsquigarrow \langle s_1', s_2'\rangle}{\langle s, \texttt{prog } \{\texttt{attempt } I \texttt{ end}\}\rangle \rightsquigarrow\ s_2'}\ \textit{program}$$

$$\frac{}{\langle s_1, s_2, x := e\rangle \rightsquigarrow\ \langle s_1, s_2[\mathcal{E}[\![e]\!]s_2/x]\rangle}\ \textit{przypisanie}$$

$$\frac{\langle s_1, s_2, I_1\rangle \rightsquigarrow s'}{\langle s_1, s_2, I_1; I_2\ \rangle \rightsquigarrow\ s'}\ \textit{złożenie}$$

$$\frac{\langle s_1, s_2, I_1\rangle \rightsquigarrow \langle s_1', s_2'\rangle \quad \langle s_1', s_2', I_2\rangle \rightsquigarrow k}{\langle s_1, s_2, I_1; I_2\ \rangle \rightsquigarrow\ k}\ \textit{złożenie} \qquad \text{dla } k = \langle s_1'', s_2''\rangle \text{ lub } k = s''$$

$$\frac{}{\langle s_1, s_2, \texttt{protect}\rangle \rightsquigarrow\ \langle s_2, s_2\rangle}\ \texttt{protect} \qquad \frac{}{\langle s_1, s_2, \texttt{reject}\rangle \rightsquigarrow\ s_1}\ \texttt{reject}$$

$$\frac{\langle s_2, s_2, I\rangle \rightsquigarrow s'}{\langle s_1, s_2, \texttt{attempt } I \texttt{ end}\ \rangle \rightsquigarrow\ \langle s_1, s'\rangle}\ \texttt{attempt}$$

$$\frac{\langle s_2, s_2, I\rangle \rightsquigarrow \langle s_1', s_2'\rangle}{\langle s_1, s_2, \texttt{attempt } I \texttt{ end}\ \rangle \rightsquigarrow\ \langle s_1, s_2'\rangle}\ \texttt{attempt}$$

$$\frac{\mathcal{B}[\![b]\!]s_2 = \texttt{tt} \quad \langle s_1, s_2, I; \texttt{while } b \texttt{ do } I \texttt{ end}\rangle \rightsquigarrow\ k}{\langle s_1, s_2, \texttt{while } b \texttt{ do } I \texttt{ end}\rangle \rightsquigarrow\ k}\ \texttt{while} \qquad \text{dla } k = \langle s_1', s_2'\rangle \text{ lub } k = s'$$

$$\frac{\mathcal{B}[\![b]\!]s_2 = \texttt{ff}}{\langle s_1, s_2, \texttt{while } b \texttt{ do } I \texttt{ end}\rangle \rightsquigarrow\ \langle s_1, s_2\rangle}\ \texttt{while}$$

$$\frac{\mathcal{B}[\![b]\!]s_2 = \texttt{tt} \quad \langle s_1, s_2, I; \texttt{if } b \texttt{ then } I\rangle \rightsquigarrow\ k}{\langle s_1, s_2, \texttt{if } b \texttt{ then } I\rangle \rightsquigarrow\ k}\ \texttt{if} \qquad \text{dla } k = \langle s_1', s_2'\rangle \text{ lub } k = s'$$

$$\frac{\mathcal{B}[\![b]\!]s_2 = \texttt{ff}}{\langle s_1, s_2, \texttt{if } b \texttt{ then } I\rangle \rightsquigarrow\ \langle s_1, s_2\rangle}\ \texttt{if}$$