SEMANTYKA I WERYFIKACJA - Zadanie domowe nr 1

Napisz semantykę operacyjną dużych kroków instrukcji języka umożliwiającego prostą obsługę odwoływalnych transakcji. Składnia jest opisana gramatyką:

$$
\begin{aligned}
Num \ni n &::= \texttt{0} \mid \texttt{1} \mid -\texttt{1} \mid \texttt{2} \mid -\texttt{2} \mid \cdots \\
Var \ni x &::= \texttt{x} \mid \texttt{y} \mid \cdots \\
TrId \ni t &::= \texttt{t} \mid \texttt{u} \mid \cdots \\
Expr \ni e &::= n \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2 \\
BExpr \ni b &::= \texttt{true} \mid \texttt{false} \mid e_1 < e_2 \mid e_1 = e_2 \mid b_1 \wedge b_2 \mid \neg b \\
Instr \ni I &::= x := e \mid I_1; I_2 \mid \texttt{if } b \texttt{ then } I_1 \mid \texttt{while } b \texttt{ do } I \mid \\
&\qquad \texttt{try } t : I \mid \texttt{fail } t \mid \texttt{commit}
\end{aligned}
$$

Wyliczanie wyrażeń arytmetycznych i logicznych odbywa się standardowo. Można założyć, że semantyka wyrażeń jest dana.

Instrukcja `try` $t : I$ rozpoczyna *transakcję* o nazwie t, polegającą na wykonaniu instrukcji I.

Instrukcja `fail` t przerywa wykonanie transakcji o nazwie t. Zauważmy, że jednocześnie może się wykonywać wiele zagnieżdżonych transakcji. Jeżeli więcej niż jedna z nich ma nazwę t, to przerwaniu ulega „najbliższa" z nich. Jeżeli aktualnie nie jest wykonywana żadna transakcja o nazwie t, to instrukcja `fail` t nie wywołuje żadnego skutku i program normalnie wykonuje się dalej.

Przy przerwaniu transakcji odwoływane są wszystkie zmiany wartości zmiennych dokonane od chwili rozpoczęcia tej transakcji. Instrukcja `commit` zmienia tę zasadę. Utrwala ona aktualny stan tak, że zmiany dokonane przed jej wykonaniem pozostają w mocy przy przerywaniu transakcji. Innymi słowy, jeżeli podczas wykonania transakcji wykonano co najmniej jedną instrukcję `commit`, to przy przerwaniu tej transakcji odwoływane są tylko zmiany wartości zmiennych dokonane po ostatnim wykonaniu instrukcji `commit` w tej transakcji.

Przykładowo, po wykonaniu programu

```
x := 0; y := 0; z := 10;
try t:
   while true do
      commit; x := x+1;
      try u:
         while true do
            y := y+1; z := z-1;
            if x<y then fail u;
            if z<x then fail t
```

(gdzie wcięcia określają zasięg instrukcji `while` i `try`) zmienna `x` przyjmuje wartość 5, zmienna `y` wartość 0, a zmienna `z` wartość 10.

W rozwiązaniu wystarczy podać reguły semantyczne dla instrukcji `try`, `fail`, `commit` i dla sekwencji (średnika).