Ficha 2

Semântica das Linguagens de Programação

2019/20

1. Considere o seguinte programa:

   ```
   z := 0; while (x<=y) do {z := z+1; x := x-y}
   ```

   (a) Determine um estado para o qual a sequência de derivação deste programa é finita e outro para o qual é infinita.

   (b) Escreva sequência de derivação gerada pela semântica operacional estrutural (*small-step*) para o caso finito.

   (c) Apresente as árvores de derivação que justificam cada um dos três primeiros passos de redução.

2. Considere o seguinte programa da linguagem **While**:

   ```
   x := 1;
   while (b>0) do {
     x := x*a;
     b := b-1 }
   ```

   (a) Recorrendo à semântica de transições (*small step*) simule a execução do programa a partir do estado inicial s em que $s\,a = 3$ e $s\,b = 2$. Apresente as árvores de derivação que justificam cada uma das 4 primeiras transições.

   (b) Considere o programa `x:=0` e o programa `while (x>0) do x:=x-1`. Serão estes dois programas equivalentes? Justifique a sua resposta com base na noção de equivalência induzida pela semântica de transições (*small step*).

3. Recorrendo à semântica operacional *small-step* prove a *equivalência* dos seguintes comandos:

   (a) `while` b `do` C e `if` b `then` $\{C\,;$`while` b `do` $C\}$ `else skip`

   (b) $\{C_1\,;C_2\}\,;C_3$ e $C_1\,;\{C_2\,;C_3\}$

4. Pretende-se estender a linguagem **While**, acrescentando-lhe uma nova forma de ciclo de acordo com a seguinte sintaxe abstracta:

$$\mathbf{Stm} \ni C ::= \ldots \mid \texttt{do}\ C_1\ \texttt{while}\ b$$

A descrição informal da semântica deste comando é a seguinte:

> *O comando C_1 é executado repetidamente enquanto o valor da expressão b for verdadeiro, sendo o teste feito depois da execução do comando.*

(a) Especifique formalmente o comportamento deste novo ciclo, escrevendo regras apropriadas para a *semântica operacional estrutural* da linguagem. As regras não devem fazer referência a outros ciclos.

(b) Tendo em conta as regras que propôs, prove a *equivalência* dos dois comandos seguintes:

$$\texttt{do}\ C\ \texttt{while}\ b \qquad \text{e} \qquad C\ ;\ \texttt{while}\ b\ \texttt{do}\ C$$

5. Podemos adoptar uma abordagem operacional *small-step* para dar semântica às expressões aritméticas e booleanas. Para isso, considere $\mathbf{State} = \mathbf{Var} \to \mathbf{Num}$.

(a) Defina uma semântica de transições *small-step* para as expressões aritméticas.

(b) Defina uma semântica de transições *small-step* para as expressões booleanas.

(c) Com poderia fazer uma avaliação "curto-circuito" das expressões booleanas (ao estilo do C) ?

6. Construa em Haskell um programa que simule a avaliação de programas, num dado estado, de acordo com a semântica operacional estrutural (*small-step*).

(a) Defina a função `stepSOS` que implementa a relação de transição $\langle C, s\rangle \Rightarrow \gamma$ que representa o primeiro passo de execuçao do programa C no estado s.

(b) Defina a função `nstepsSOS` que simula n passos de execução (se possível) de um programa C num estado s.

(c) Defina a função `evalSOS` que simula a execução completa de um programa C num estado s.

(d) Defina alguns exemplos de programas e estados e teste as funções que definiu.