

Ficha 1

Semântica das Linguagens de Programação

2021/22

1. Defina na linguagem **While** os seguintes programas

- SWAP - que troca de valores entre as variáveis x e y .
- MIN - que calcula em m do mínimo de x , y e z .
- EXP - que calcula em r do valor de x elevado a y .
- FACT - que calcula em f do valor do factorial de n .

Construa árvores de derivação para as transições correspondentes à execução de cada um dos programas nos estado s , sendo s o estado que mapeia todas as variáveis em 0 excepto os seguintes casos: $s x = 3$, $s y = 2$ e $s n = 3$.

2. Prove que de acordo com a semântica natural da linguagem **While** apresentada nas aulas teóricas os seguintes comandos são semanticamente equivalentes.

- (a) $\{S_1 ; S_1\} ; S_3$ e $S_1 ; \{S_1 ; S_3\}$
(b) `while b do C` e `if b then $\{C ; \text{while } b \text{ do } C\}$ else skip`

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

```
if (x > 0) then y := x
else if (x < 0) then y := -x
    else z := 1
```

- (a) Recorrendo à semântica de avaliação (*big-step*) simule a execução do programa a partir do estado inicial s em que $s x = -1$.
(b) Diga, se o programa acima é *equivalente* a algum dos dois seguintes. Apresente em cada caso uma prova (se forem iguais) ou um contra-exemplo (se não forem).

<pre>y := x; if (y < 0) then y := -y else z := 1;</pre>	<pre>if (x == 0) then z := 1 else if (x < 0) then y := -x else y := x;</pre>
--	---

4. Pretende-se definir uma semântica natural para a avaliação de expressões.

- (a) Especifique o sistema de transição para a avaliação de expressões aritméticas.
(b) Prove que o significado de uma expressão aritmética dado por esta nova definição é o mesmo do que o dado por \mathcal{A} .
(c) Especifique o sistema de transição para a avaliação de expressões booleanas.

5. Imagine agora que pretendemos acrescentar operadores com efeitos laterais (como o $++x$ e o $x++$ do C) à linguagem de expressões aritméticas.

$$\mathbf{Aexp} \ni a ::= n \mid x \mid ++x \mid x++ \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2$$

- (a) Proponha uma semântica natural que permita capturar o efeito da avaliação destas novas expressões aritméticas.
- (b) Defina uma semântica natural expressões booleanas.
- (c) Com base na semântica que definiu, indique o valor das expressões $(++x) * x$ e $(x++) * x = 0$ no estado s em que $sx = 0$.
- (d) Que consequências isto acarreta na semântica dos programas?
- (e) Proponha um conjunto de regras de avaliação que captem corretamente o comportamento destes novos programas.

6. Pretende-se estender a linguagem **While**, acrescentando-lhe uma nova forma de ciclo, inspirado no comando com a mesma sintaxe na linguagem C, e com um comportamento igual.

$$\mathbf{Stm} \ni C ::= \dots \mid \text{for } (C_1, b, C_3) \text{ do } C_2$$

- (a) Especifique formalmente o comportamento deste novo ciclo, escrevendo regras apropriadas para a *semântica natural* da linguagem. As regras não devem fazer referência a outros ciclos.
- (b) Recorrendo à semântica de natural e tendo em conta as regras que propôs, prove que o comando `while b do $\{C_2; C_3\}$` e o comando `for(skip , b , C_3) do C_2` são *equivalentes*. Isto é, mostre que para qualquer $s, s_f \in \mathbf{State}$,

$$\langle \text{while } b \text{ do } \{C_2; C_3\}, s \rangle \rightarrow s_f \quad sse \quad \langle \text{for}(\text{skip}, b, C_3) \text{ do } C_2, s \rangle \rightarrow s_f$$

A prova das duas implicações são por indução na derivação do antecedente.

- (c) Prove agora a *equivalência* dos dois comandos seguintes:

$$\text{for}(C_1, b, C_3) \text{ do } C_2 \quad e \quad C_1; \text{while } b \text{ do } \{C_2; C_3\}$$

Terá que demonstrar que para qualquer $s, s_f \in \mathbf{State}$,

- i. $\langle \text{for}(C_1, b, C_3) \text{ do } C_2, s \rangle \rightarrow s_f \Rightarrow \langle C_1; \text{while } b \text{ do } \{C_2; C_3\}, s \rangle \rightarrow s_f$
- ii. $\langle C_1; \text{while } b \text{ do } \{C_2; C_3\}, s \rangle \rightarrow s_f \Rightarrow \langle \text{for}(C_1, b, C_3) \text{ do } C_2, s \rangle \rightarrow s_f$

A prova de (i) faz-se por indução na derivação de $\langle \text{for}(C_1, b, C_3) \text{ do } C_2, s \rangle \rightarrow s_f$.

A prova de (ii) não é feita por indução. Usa o lema demonstrado na alínea anterior.

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

$$\mathbf{Stm} \ni C ::= \dots \mid \text{repeat } C_1 \text{ until } b$$

A descrição informal da semântica deste comando é a seguinte:

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

- (a) Especifique formalmente o comportamento deste novo ciclo, escrevendo regras apropriadas para a *semântica natural* da linguagem. As regras não devem fazer referência a outros ciclos.
- (b) Tendo em conta as regras que propôs, prove a *equivalência* entre dois comandos seguintes:

`repeat C until b` e `C ; while ¬b do C`

8. Pretende-se construir em Haskell um programa que simule a avaliação de programas.

- (a) Comece por simular a avaliação de expressões. Para isso:
 - i. Defina tipos para as categorias sintáticas **Aexp** e **Bexp**.
 - ii. Defina as operações de substituição para expressões **Aexp** e **Bexp**.
 - iii. Defina as funções semânticas \mathcal{A} e \mathcal{B} .
- (b) Implemente agora um programa que simule a avaliação de programas, num dado estado, de acordo com a semântica natural (*big-step*).
 - i. Defina o tipo para a categoria sintáctica dos comandos, **Stm**.
 - ii. Defina a função auxiliar de “update” do estado.
 - iii. Defina a função `evalNS` que simula a relação de avaliação dos comandos.
 - iv. Defina alguns exemplos de programas e estados e teste a relação de avaliação que definiu.