

INF05516 - Semântica Formal - Prova II - 2009/1

Nome: _____

Número: _____

Instruções:

- Todas as questões são relativas a linguagem L3 sem subtipos, exceto quando explicitamente mencionado.
- A duração da prova é de 100 minutos.
- As respostas devem ser dadas **a caneta e somente** nos espaços designados
- O que for escrito fora dos espaços para as soluções **não** será levado em conta na correção
- As questões regulares totalizam 10 pontos.
- As duas questões extras conferem pontos adicionais.

	Pontos
1	
2	
3	
4	
5	
6	
Extra 1	
Extra 2	
Total	

1. Considere a linguagem cuja sintaxe abstrata é dada pela gramática abaixo:

expressões $e ::= \square \mid \nabla \mid e_1 \asymp e_2$

valores $v ::= \square \mid \nabla$

tipos $T ::= \text{Quadrado} \mid \text{Triangulo}$

A semântica operacional *small step* dessa linguagem é dada pelas seguintes regras:

$$\frac{}{\square \asymp \nabla \rightarrow \nabla} \quad (\text{AX1})$$

$$\frac{}{\square \asymp \square \rightarrow \square} \quad (\text{AX2})$$

$$\frac{e_1 \rightarrow e'_1}{e_1 \asymp e_2 \rightarrow e'_1 \asymp e_2} \quad (\text{RGR1})$$

$$\frac{e_2 \rightarrow e'_2}{\square \asymp e_2 \rightarrow \square \asymp e'_2} \quad (\text{RGR2})$$

(a) (0,5pt) Diga o que acontece com a avaliação da expressão $(\square \asymp \square) \asymp (\square \asymp \nabla)$

- ☒ Termina com o valor ∇
- ☐ Termina com a expressão _____, que não é valor
- ☐ A avaliação da expressão nunca termina

(b) (0,5pt) Idem para a expressão $(\square \asymp \nabla) \asymp (\nabla \asymp \square)$

- ☐ Termina com o valor _____
- ☒ Termina com a expressão $\square \asymp (\nabla \asymp \square)$, que não é valor
- ☐ A avaliação da expressão nunca termina

(c) (1pt) Considere o seguinte sistema de tipos para essa linguagem:

$\square : \text{Quadrado} \quad \nabla : \text{Triangulo}$

$$\frac{e_1 : T_1 \quad e_2 : T_2}{e_1 \asymp e_2 : T_2}$$

Esse sistema de tipos é seguro em relação a semântica operacional ? Justifique a sua resposta.

O sistema de tipos não é seguro pois a propriedade do progresso de expressões bem tipadas não é verdadeira. Considere por exemplo a expressão $\square \asymp (\nabla \asymp \square)$. Pelo sistema de tipos ela é considerada como sendo do tipo Quadrado, porém ela não é um valor da linguagem e não há regra na semântica operacional que a permita progredir na sua avaliação.

2. Diga se a expressões abaixo são bem tipadas ou não. Em caso positivo, escreva o tipo da expressão. Considere o sistema da tipos de L3 **sem subtipos**:

(a) (0,5pt) $[x : \text{bool} \rightarrow (\text{bool} \rightarrow \text{int}), y : \text{unit} \rightarrow \text{bool}] \vdash \text{fn } z : \text{unit} \Rightarrow x (y z) : ?$

☐ Não
☒ Sim Tipo: $\text{unit} \rightarrow \text{bool} \rightarrow \text{int}$

(b) (0,5pt) $[x : \text{int} \rightarrow \text{int}] \vdash x (\text{fn } z : (\text{int} \rightarrow \text{int}) \rightarrow \text{int} \Rightarrow z x) : ?$

☒ Não
☐ Sim Tipo: _____

3. (0,5pt) Diga se a expressão $\{p = \{p = \{p = \{p = 3\}\}\}\}$ pode ser do tipo $\{ \}$ em L3 **com subtipos**:

☐ Não
☒ Sim

4. (0,5pt) Diga se a expressões abaixo é bem tipada ou não. Em caso positivo, escreva o tipo da expressão. Considere o sistema da tipos de L3 **com subtipos**:

$\emptyset \vdash \text{fn } f : \{p : \text{int}, q : \text{int}\} \rightarrow \text{int} \Rightarrow (f \{q = 3, p = 2\}) + (f \{p = 4\}) : ?$

☒ Não
☐ Sim Tipo: _____

5. Considere as seguintes expressões:

```
A ≡ fn x : bool =>
  try
    if x then
      raise
    else
      fn y : bool => true
  with
    (fn z : _____ => z) raise
```

$B \equiv A \text{ false}$

$C \equiv B \text{ true}$

- (a) (0,5pt) Que tipo deve ter a variável z para que o termo A seja bem tipado? $\text{bool} \rightarrow \text{bool}$
 (b) (0,5pt) Qual o tipo da expressão A ? $\text{bool} \rightarrow \text{bool} \rightarrow \text{bool}$
 (c) (0,5pt) Qual o resultado da avaliação da expressão B ? $\text{fn } y : \text{bool} \Rightarrow \text{true}$
 (d) (0,5pt) Qual o resultado da avaliação da expressão C ? true

6. Defina as **regras de tipo** e as regras da **semântica operacional** *small step* para uma extensão da linguagem L3 com listas e operações sobre listas dada pela gramática abaixo:

$$\begin{aligned} e &::= \dots \mid nil \mid e_1 :: e_2 \mid hd\ e \mid tl\ e \\ v &::= \dots \mid nil \mid v_1 :: v_2 \end{aligned}$$

$$T ::= \dots \mid T\ list$$

Listas são coleções ordenadas de dados do mesmo tipo. A expressão `nil` é a lista vazia. A expressão `e1 :: e2` é uma lista onde `e1` é o primeiro elemento da lista e `e2` é o restante da lista. Uma lista cujos elementos são todos valores é um valor. A expressão `hd e` retorna o primeiro elemento da lista `e`, ou retorna `raise` caso a lista `e` seja vazia. A expressão `tl e` retorna a lista resultante da eliminação do primeiro elemento da lista `e`, ou `raise` caso `e` seja uma lista vazia. Exemplos de listas não vazias (já completamente avaliadas) são: `1 :: (3 :: (7 :: (0 :: nil)))` e `true :: (true :: (false :: nil))`. A expressão `hd 1 :: (3 :: (7 :: (0 :: nil)))` retorna o elemento 1 e a expressão `tl 1 :: (3 :: (7 :: (0 :: nil)))` retorna a lista `3 :: (7 :: (0 :: nil))`. A lista `1 :: (3 :: (7 :: (0 :: nil)))` é do tipo `int list` e a lista `true :: (true :: (false :: nil))` é do tipo `bool list`. A lista vazia `nil` pode ser de qualquer tipo lista.

Regras de tipo: (2pts)

$$\Gamma \vdash nil : T\ list$$

$$\frac{\Gamma \vdash e_1 : T \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 :: e_2 : T\ list}$$

$$\frac{\Gamma \vdash e : T\ list}{\Gamma \vdash hd\ e : T}$$

$$\frac{\Gamma \vdash e : T\ list}{\Gamma \vdash tl\ e : T\ list}$$

Regras da semântica operacional: (2pts)

$$\frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{e_1 :: e_2, \sigma \longrightarrow e'_1 :: e_2, \sigma'}$$

$$\frac{e_2, \sigma \longrightarrow e'_2, \sigma'}{v :: e_2, \sigma \longrightarrow v :: e'_2, \sigma'}$$

$$raise :: e_2, \sigma \longrightarrow raise, \sigma \quad v :: raise, \sigma \longrightarrow raise, \sigma$$

$$\frac{e, \sigma \longrightarrow e', \sigma'}{hd\ e, \sigma \longrightarrow hd\ e', \sigma'}$$

$$hd\ v_1 :: v_2, \sigma \longrightarrow v_1, \sigma \quad hd\ nil, \sigma \longrightarrow raise, \sigma \quad hd\ raise, \sigma \longrightarrow raise, \sigma$$

$$\frac{e, \sigma \longrightarrow e', \sigma'}{tl\ e, \sigma \longrightarrow tl\ e', \sigma'}$$

$$tl\ v_1 :: v_2, \sigma \longrightarrow v_2, \sigma \quad tl\ nil, \sigma \longrightarrow raise, \sigma \quad tl\ raise, \sigma \longrightarrow raise, \sigma$$

Questão extra 1: (1pt) Os métodos de uma classe podem ser **reusados** para definir novas classes chamadas subclasses. Por exemplo, supondo que já tenhamos definido uma classe *counterClass*, podemos definir uma classe *it resetCounterClass* de contadores com *reset* da seguinte forma:

```

1. resetCounterClass =
2.   fn r : CounterRep  $\Rightarrow$ 
3.     let super = counterClass r in
4.       {get = super.get
5.        inc = super.inc
6.        reset = fn _ : unit  $\Rightarrow$  r.x := 1}

```

A cópia explícita da maioria dos campos da superclasse no registro da subclasse ainda é inconveniente (linhas 4 e 5). Como está evita-se repetir todo o código dos métodos da superclasse na subclasse, mas mesmo assim requer muita digitação. Para programas OO maiores será útil dispormos de uma construção como

```
super with {reset = fn _: unit => r.x := 1}
```

(no lugar das linhas 4,5 e 6) representando um registro como `super` mas com o campo `reset` redefinido. Defina a sintaxe, semântica operacional e regra de tipo para essa nova construção.

1. Defina a sintaxe abstrata dessa nova expressão
2. Defina a semântica operacional dessa expressão
3. Defina uma regra de tipo para a expressão

Questão extra 2: (1pt) Prove que o sistema de tipos definido na questão 1(c) é seguro em relação a semântica operacional da linguagem.