Interpretação e Compilação de Linguagens de Programação

Exame de Época Normal 2008-2009 — 14 de Janeiro de 2009

Notas: O exame pode ser realizado com consulta de materiais em suporte de papel trazidos pelo aluno. O exame tem a duração de 3h.

1. Considere a linguagem *LAZY* cuja sintaxe concreta é definida pela seguinte gramática:

$$
\begin{array}{rcl}
E & ::= & \mathit{Num} \\
 & | & E\langle + \rangle E \\
 & | & \mathit{Id} \\
 & | & \langle \texttt{var} \rangle\ \mathit{Id}\ \langle = \rangle\ E\ \langle \texttt{in} \rangle\ E \\
 & | & \langle \texttt{const} \rangle\ \mathit{Id}\ \langle = \rangle\ E\ \langle \texttt{in} \rangle\ E \\
 & | & \mathit{Id}\ \langle := \rangle\ E \\
 & | & \langle \texttt{fun} \rangle\ \langle () \rangle\ [\langle \texttt{const} \rangle]\ \mathit{Id}_1, \cdots, [\langle \texttt{const} \rangle]\ \mathit{Id}_n\ \langle () \rangle\ \langle \{ \rangle\ E\ \langle \} \rangle \\
 & | & E\ \langle () \rangle\ E, \cdots, E\ \langle () \rangle \\
 & | & \langle \texttt{if} \rangle\ \langle () \rangle\ E\ \langle () \rangle\ E\ \langle \texttt{else} \rangle\ E
\end{array}
$$

As palavras chave da linguagem são indicadas na gramática delimitados por $\langle\ \rangle$. As construções da linguagem são os números inteiros (*Num*) e as operações habituais sobre inteiros (aqui representados pela operação de adição), os identificadores (*Id*), a declaração de variáveis (`var ...`), a declaração de constantes (`const ...`), a afectação, a definição e chamada de funções e a expressão condicional. A declaração dos parâmetros de funções **pode** ser modificada através da palavra chave `const` o que quer dizer que o valor dos parâmetros `const` não pode ser modificado. Um parâmetro não `const` comporta-se como uma variável local. A expressão condicional tem o comportamento esperado com o valor 0 (zero) para a condição a ser interpretado como falso e qualquer outro valor inteiro como verdadeiro. Na linguagem *LAZY* as expressões são avaliadas de uma forma *lazy* (passagem de parâmetros por necessidade). Note que a declaração de constantes e variáveis força a avaliação da expressão inicializadora antes do corpo da declaração.

(a) **Descreva** a sintaxe abstracta da linguagem *LAZY* usando um conjunto de classes Java. Escreva apenas a assinatura de um constructor de cada classe com o seu nome e variáveis de instância.

(b) **Defina** a semântica da linguagem *LAZY* através da implementação de um método de cada uma das classes `ASTId`, `ASTVar`, `ASTConst`, `ASTFun`, `ASTCall` que interprete a construção correspondente. **Explique** sucintamente quais os parâmetros e valores de retorno da função `evaluate`. Note que a linguagem segue a estratégia de avaliação de expressões por necessidade e resolução estática de nomes.

(c) **Enumere** quais os erros de execução que podem ocorrer num programa *LAZY*.

2. Considere o programa escrito na linguagem *LAZY*

```
var x = 0 in
var y = 0 in
const i = fun (const c, const t, const e) { c(t,e) } in
const tt = fun (const t, const e) { t } in
const ff = fun (const t, const e) { e } in
const g = fun (const f) { f(f) } in
const h = fun (const f) { 1 } in
  i( ff, x := x+g(g), y := y+g(h))
```

   (a) **Indique** qual o ambiente de avaliação da expressão `e` no corpo da função `f`.

   (b) **Indique** o valor das variáveis `x` e `y` após a execução do programa.

   (c) **Indique** o resultado quando o programa é interpretado com passagem de parâmetros por valor.

   (d) **Escreva** um exemplo de um programa da linguagem *LAZY* que apresente resultados diferentes se interpretado com passagem de parâmetros por valor ou passagem de parâmetros por necessidade.

3. **Enumere** quais são as características mínimas e essenciais de uma linguagem de programação para que se possam registar fenómenos de *aliasing*. **Justifique** usando pequenos exemplos de programas.

4. Considere a linguagem *TAZY*, uma extensão tipificada da linguagem *LAZY* com anotações de tipos na declaração dos parâmetros das funções.

   (a) **Indique** quais os tipos necessários para tipificar a linguagem *TAZY*.

   (b) **Defina** um sistema de tipos da linguagem *TAZY* que evite pelo menos as afectações a constantes ou parâmetros constantes.

   (c) **Explique sucintamente** como estenderia o sistema de tipos da alínea anterior para incluir funções puras (funções que não modificam o estado do programa). As funções declaradas como puras são etiquetadas com a palavra reservada `const` conforme o seguinte exemplo:

   ```
   const fun( Int x ) : Int { x + 1 }
   ```

   Note que o sistema de tipos rejeitaria o programa

   ```
   var x = 1 in const fun() : Int { x := 1 }
   ```