Exame de Interpretação e Compilação de Linguagens de Programação

Época Normal 2007-2008 — 9 de Janeiro de 2008

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

Considere a linguagem de programação *FunPair* cuja sintaxe abstracta é descrita pela seguinte gramática:

$$
\begin{array}{rcl}
Program & ::= & \mathsf{var}\ Id_1, ..., Id_n\ \mathsf{begin}\ Command\ \mathsf{end} \\
Command & ::= & Id := Expression \mid Command; Command \mid Program \\
Expression & ::= & Id \\
& \mid & Num \\
& \mid & Expression + Expression \\
& \mid & \mathsf{Function}\ Id_1, ..., Id_n\ \mathsf{var}\ Id'_1, ..., Id'_m\ \mathsf{begin}\ Command\ \mathsf{return}\ Expression \\
& \mid & Id(Expression_1, ..., Expression_n) \\
& \mid & (Expression, Expression) \\
& \mid & \mathsf{fst}\ Expression \\
& \mid & \mathsf{snd}\ Expression
\end{array}
$$

As construções da linguagem são o identificador (Id), os números inteiros (Num), a adição de inteiros ($... + ...$), a abstracção funcional (Function...), a aplicação funcional ($Id(...)$), a construção de pares ordenados ($(..., ...)$), a selecção do primeiro elemento de um par ordenado (fst ...), e a selecção do segundo elemento de um par ordenado (snd ...). Os programas são sequências de declarações de variáveis seguidos de um bloco de comandos (var ... begin ... end). Os comandos podem ser afectações ($Id := ...$), sequências de comandos ou programas. O âmbito das declarações de variáveis é o comando do corpo no caso de um programa e o comando do corpo e a expressão de retorno no caso das funções. As declarações de variáveis locais às funções e o comando do corpo das funções e programas podem ser vazios, nesse caso omitem-se as palavras chave var e/ou begin. Observe o seguinte exemplo da linguagem *FunPair*:

$$
\begin{array}{l}
\mathsf{var}\ sum, res\ \mathsf{begin} \\
\quad sum :=\ \mathsf{Function}\ x, y\ \mathsf{var}\ res\ \mathsf{begin}\ res := x + y\ \mathsf{return}\ res; \\
\quad res := sum(10, 20) \\
\mathsf{end}
\end{array}
$$

1. [3 valores] Descreva o conjunto de classes Java, e respectivas variáveis de instância, necessários para implementar o interpretador da linguagem *FunPair*. Indique quais os membros de dados de cada classe através do nome e parâmetros do seu construtor.

2. [5 valores] Implemente, para cada uma das classes da resposta à pergunta 1, um método que interprete as construções da linguagem *FunPair* seguindo uma estratégia de avaliação dos argumentos *call-by-value* e resolução estática de nomes. **Indique detalhadamente** qual a assinatura completa do método e quais as hierarquias de interfaces e classes necessárias.

(continua na página seguinte)

3. [3 valores] Considere o seguinte programa na linguagem *FunPair*:

```
var a, b, c begin
  a := 1;
  b := Function x return x + 1;
  c := Function x var e begin e := x + a return b(e);
  var b, d begin
    a := 2;
    b := Function x return c(x + 1);
    d := c(1)
  end
end
```

   (a) Usando **resolução estática de nomes** diga justificando se o programa termina e quais os valores das variáveis após a execução do comando $d := c(1)$.

   (b) Usando **resolução estática de nomes** indique justificando qual o conteúdo da memória e qual o ambiente de avaliação da expressão $x + a$.

   (c) Usando **resolução dinâmica de nomes** diga justificando se o programa termina e quais os valores das variáveis após a execução do comando $d := c(1)$.

4. [4 valores] Estenda a linguagem *FunPair* com a construção this que denota o par corrente. No exemplo seguinte, o par atribuído à variável p contém duas funções. No corpo da segunda função, obtido pela expressão (snd p), é referido o primeiro elemento do par, através da expressão (fst this). **Indique justificadamente quais as modificações** necessárias à semântica apresentadas na resposta à pergunta 2 para avaliar a linguagem extendida.

```
var p, g, x begin
  p := (Function x return x + 1, Function x var f begin f := fst this return x + f(x))
  g := snd p
  x := g(1)
end
```

5. [5 valores] Defina a linguagem *TyFunPair*, que é uma extensão tipificada da linguagem *FunPair* em que os programas bem tipificados não têm os seguintes erros de execução:

   - Utilização de uma variável não definida.
   - Chamada de uma função sobre um valor que não uma função.
   - Passagem de um número de argumentos diferente do número de parâmetros.
   - Operações aritméticas sobre valores que não são inteiros (pares ou funções).
   - Selecção de um membro (primeiro ou segundo) de um valor que não é um par.

   (a) Indique as alterações à sintaxe da linguagem base (*FunPair*) e defina as classes, interfaces, e métodos necessários para a correcta tipificação de programas *TyFunPair*. **Justifique** como é que o sistema de tipos evita os erros acima referidos.

   Apresente exemplos e justifique se as afirmações seguintes são verdadeiras ou falsas.

   (b) "Todos os programas *TyFunPair* onde não ocorrem os erros de execução acima referidos estão bem tipificados."

   (c) "Podem ocorrer erros de execução em programas *TyFunPair* bem tipificados."