Exame de Linguagens de Programação II

12 Janeiro 2005

I [6]

Considere a linguagem FUN definida pela seguinte gramática

```
Expr ::= {fun} Id {->} Expr |
       {self} |
       Expr {+} Expr |
       Expr {-} Expr |
       Expr {@} Expr |
       {if} Exp {then} Exp {else} Exp
       Num |
       Id |
       {() Expr {}}
```

onde os símbolos terminais estão representados entre { } e Num e Id representam respectivamente os literais numéricos (ex: 0, 98) e os identificadores (ex, i, xpto, luis). Todos os programas nesta linguagem são expressões que denotam um valor inteiro. O símbolo + (-) representa a operação de adição (subtração) de inteiros.

Uma expressão da forma `if Exp1 then Exp2 else Exp3` calcula o valor de `Exp2` se o valor de `Exp1` é diferente de zero, e o valor de `Exp3` caso contrário.

Uma expressão da forma `fun Id -> Expr` representa a função de inteiros em inteiros de argumento `Id` e corpo `Expr`. Por exemplo, a expressão

```
fun x -> x+1
```

representa a função sucessor. As funções expressas por `fun Id -> Exp` podem ser recursivas, pois dentro de `Exp`, a própria função pode ser referida através do nome especial `self`. Por exemplo, a expressão

```
fun x -> if x then x + self@(x-1) else 0
```

representa a função que dado `n` calcula $0 + 1 + 2 + \dots + n$.

Uma expressão da forma

```
Expr1@Expr2
```

representa aplicação da função `Expr1` ao argumento `Expr2`. O objectivo deste exercício é desenvolver um pequeno interpretador para a linguagem FUN.

1. [1] Eis um exemplo de programa nesta linguagem

```
((fun x -> x+x)@10)+((fun y -> ((fun z->y+z)@1))@2)
```

Que valor produz este programa?

2. [1] Descreva como representaria a sintaxe abstracta da linguagem FUN, usando uma hierarquia de classes na linguagem Java.
3. [2] Defina a interface de uma classe Java `Ambiente` que implemente as operações necessárias para manter o valor associado aos identificadores durante a interpretação dos programas FUN. Explique a funcionalidade de cada uma dessas operações.
4. [2] Especifique em cada uma das classes que apresentou em **2**, um método

```
int evaluate(a: Environment)
```

de tal modo que, dada a uma AST guardada na variável `exp`, a chamada

```
exp.evaluate(new Environment())
```

produza como resultado o seu valor enquanto expressão de FUN, ou assinale um erro de interpretação.

II [4]

Considere o seguinte programa escrito numa linguagem de tipo Pascal

```
program P123(output) ;
var a : integer ;
  procedure P1 ;
    procedure P2 ;
      var a : integer ;
        procedure P3 ;
          begin
            a := 3 ;
            P1
          end ;
        begin (* P2 *)
          a := 2 ;
          P3
        end ;
      begin (* P1 *)
        if a = 0 then begin a := 1 ; P2 end
        else writeln(a)
        end ;
    end ;
begin
  a := 0 ;
  P1
end.
```

1. [2] Qual o “output” do programa se a regra de resolução de nomes for o escopo estático ? E se a regra de resolução de nomes for o escopo dinâmico ? Se os resultados forem distintos, explique porquê.
2. [1] Apresente um **pequeno** programa em Pascal, sintacticamente correcto (que compila), mas que se fosse executado sob uma disciplina de escopo dinâmico provocaria um erro de execução por tentar alterar o valor de uma constante.
3. [1] Numa linguagem com escopo dinâmico não é possível efectuar em geral uma verificação estática de tipos. Porquê ?

III [6]

1. [2] No trabalho realizado nas aulas práticas, a linguagem de objectos OhOhFCT! é interpretada por tradução numa linguagem mais básica, que contém funções como entidades de primeira classe. Indique e explique em que situações são usadas funções no interpretador de OhOhFCT!.
2. Considere a linguagem FUN do Grupo I.
 - (a) [2] Que erros de execução podem ocorrer num programa FUN?
 - (b) [2] Defina, usando um conjunto de regras, um sistema de tipos para a linguagem FUN que garanta estaticamente que os erros identificados em (a) nunca ocorrerão em tempo de execução.