Exame de Linguagens de Programação II

25 Janeiro 2005

I [7]

Considere a linguagem PCF definida pela seguinte gramática

```
Expr::= {fun} Id {->} Expr |
        {succ} Expr |
        {pred} Expr |
        Expr {(} Expr {)}|
        {letrec} Id {=} Expr {in} {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 `succ` (`pred`) representa a operação successor (predecessor) 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 argumento `Id` e corpo `Expr`. Por exemplo, a expressão

```
fun x -> x
```

representa a função identidade. A construção `letrec` permite introduzir definições recursivas. 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 PCF.

1. [1] Explique como é possível representar funções de múltiplos argumentos na linguagem PCF, que apenas possui abstracções unárias da forma `fun x -> E`.

2. [2]Eis um exemplo de programa (incompleto) nesta linguagem

   ```
   letrec add = ...
     in letrec f = fun x -> add(x)(x)
       in letrec g = fun u -> u(2)
         in g(f)
   ```

   Que valor produz este programa, supondo que fornecemos uma definição correcta para a função `add`, de modo a que esta represente a adição de números inteiros? Complete o programa acima, substituindo `...` por uma expressão adequada da linguagem PCF, que permita implementar a adição.

3. [1] Descreva como representaria a sintaxe abstracta da linguagem PCF, usando uma hierarquia de classes na linguagem Java.

4. [3] Especifique em cada uma das classes que apresentou em 1 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 PCF, ou assinale um erro de interpretação.

II [4]

Considere o seguinte programa escrito numa linguagem de tipo `Pascal`:

```
program PPP ;
        var x, y : integer ;
        procedure P ;
                var x : integer ;
                begin x := 10 ; y := y + 1 end ;
        procedure Q(var y:integer) ;
                begin x := y ; P end ;
begin x := 0 ; y := 1 ; Q(x) ; write(x) ; write(y) end.
```

1. [1] Qual o "output" deste programa se a regra de resolução de nomes for "estática"? E se for "dinâmica"?
2. [2] Traduza este programa na linguagem básica definida no trabalho prático 4. Apresente o estado do ambiente no momento em que está a ser executado o comando `y:=y+1`.
3. [1] Para implementar um interpretador para a linguagem `C` não é preciso representar funções por fechos, bastando guardar para cada função apenas os seus parâmetros e o seu corpo. Porquê?

III [5]

No trabalho realizado nas aulas práticas, a linguagem de objectos OhOhFCT! é interpretada por tradução numa linguagem mais básica, que permite definir registos, e funções de ordem superior.

Explique como introduzir na linguagem OhOhFCT! um novo tipo de dados para representar *listas de elementos*, do género disponível nas linguagem OCAML.

Esta extensão à linguagem acarreta a extensão da sua categoria sintáctica das expressões para conter as quatro construções seguintes:

```
Expr::= ...
      | []
      | E1 :: E2
      | hd(E)
      | tl(E)
      | empty(E)
```

O literal `[]` tem como valor a lista vazia, enquanto que a expressão `E1 :: E2` tem como valor a lista obtida quando se coloca o valor da expressão `E1` á cabeça lista resultante de avaliar a expressão `E2`. Se a expressão `E` produzir uma lista não vazia como valor, avaliar `hd(E)` produz como resultado o seu primeiro elemento, e `tl(E)` produz como valor a lista de todos os seus elementos que não o primeiro. O predicado `empty(E)` devolve true se e só se a expressão `E` avalia para uma lista vazia.

Explique como alteraria o seu interpretador de OhOhFCT! para que este suportasse estas construções *por tradução na própria linguagem OhOhFCT! do trabalho prático.*