Paradigmas de Linguagens de Programação
Exame Escrito

Centro de Informática - UFPE
31 de julho de 2003

Questão 1 [1,0] Defina uma função ***merge*** de alta ordem (usando uma notação semelhante à LF2) que recebe como parâmetros duas listas de mesmo tamanho, uma função e um predicado (função *booleana*) cujo tipo do argumento é o produto cartesiano dos elementos das duas listas. O retorno da função deve ser uma lista onde cada elemento é o resultado da aplicação da primeira função a elementos de mesma posição na lista original, desde que o par original torne o predicado verdadeiro (***true***). Por exemplo,

merge *[a,b,c] [x,y,z] f p = [f(a,x), f(c,z)]*

assumindo que *p(a,x) = p(c,z) = true* e *p(b,y) = false*.

Questão 2 [1,0] Quais os requisitos necessários à caracterização de uma linguagem de programação? O que é um paradigma de programação?

Questão 3 [1,0] Explique o que é referência pendente (*dangling reference*) e o que pode causá-la.

Questão 4 [1,0] Defina os conceitos de *binding* e escopo de um identificador, explique a diferença entre escopo estático e dinâmico, e o que é um *buraco* (**hole**) em um escopo.

Questão 5 [1,0] Explique o que é o princípio da abstração (*abstraction principle*) e qual a sua importância.

Questão 6 [1,0] Orientação a objetos oferece polimorfismo via subtipos. Explique esta facilidade e compare com polimorfismo real (paramétrico).

Questão 7 [4,0] Na linguagem funcional ML, além de variáveis lógicas (associadas a um valor fixo) há variáveis associadas a referências. Enquanto a associação entre estas últimas variáveis a referências é fixa, a associação entre a referência e o valor propriamente dito pode mudar através de comandos de atribuição. Considere o exemplo

```
let var x = 1, ref y = 2 in y:=x + y; y:=y + 1 return x + y
```

Estenda LE2 com a seguinte estrutura de bloco

```
let dec in [comando] return expressao
```

onde dec deve permitir declarações de variáveis (como antes) e de referências e a cláusula (opcional) comando permite a composição seqüencial de atribuições a referências.

a) Apresente a BNF ou um modelo UML para a nova linguagem e implemente as novas classes ou classes que precisaram ser modificadas (considerar apenas avaliação e ignorar verificação de tipos). Quanto ao ambiente de execução, basta definir a nova interface e explicar como seria implementado, mas não precisa implementar a classe **ContextoExecucao**.
b) Considerando que a mesma extensão foi feita para LF1, qual o impacto, do ponto de vista do paradigma funcional: quais as vantagens/desvantagens da extensão?
c) Seria fácil implementar um provador de teoremas para a linguagem estendida, através, por exemplo, de uma extensão do provador apresentado em sala? Explique sua resposta.

Apêndice

BNF da LE2

```
Programa ::= Expressao
        Expressao ::= Valor | ExpUnaria | ExpBinaria | ExpDeclaracao | Id
        Valor ::= ValorConcreto
        ValorConcreto ::= ValorInteiro | ValorBooleano | ValorString
        ExpUnaria ::= "-" Expressao | "not" Expressao | "length" Expressao
        ExpBinaria ::= Expressao "+" Expressao
                | Expressao "-" Expressao
                | Expressao "and" Expressao
                | Expressao "or" Expressao
                | Expressao "==" Expressao
                | Expressao "++" Expressao
        ExpDeclaracao ::= "let" DecVariavel "in" Expressao
        DecVariavel ::= "var" Id "=" Expressao | DecVariavel "," DecVariavel
```

Classes auxiliares

Tipo
Ambiente
AmbienteCompilacao
AmbienteExecucao
ContextoCompilacao
ContextoExecucao
StackHandler
VariavelJaDeclaradaException
VariavelNaoDeclaradaException
IdentificadorJaDeclaradoException
IdentificadorNaoDeclaradoException

Parte do código da classe ExpDeclaracao

```
public class ExpDeclaracao implements Expressao{

      List seqdecVariavel;
      Expressao expressao;

      public ExpDeclaracao(List declarations, Expressao expressaoArg){
            seqdecVariavel = declarations;
            expressao = expressaoArg;
      }

      public Valor avaliar(AmbienteExecucao ambiente) throws
                              VariavelNaoDeclaradaException,VariavelJaDeclaradaException {
            ambiente.incrementa();
            Map resolvedValues = resolveValueBindings(ambiente);
            includeValueBindings(ambiente, resolvedValues);
            Valor result = expressao.avaliar(ambiente);
            ambiente.restaura();
            return result;
      }
```