Paradigmas de Linguagens Computacionais
Exame Escrito

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

20 de novembro de 2014

Questão 1 Considere a seguinte definição do tipo `Valor`, representando valores resultantes da interpretação de programas escritos em uma linguagem simples que manipula booleanos, inteiros e funções:

```
data Valor = Num Int | Bool Bool | Erro | Fun (Valor -> Valor)
```

Defina em Haskell as funções a seguir, que trabalham com ambientes e memórias de interpretadores como

```
[("fat",Fun fatorial),(">0",Fun maior0),("+2",Fun adiciona2)],
```

ou seja, listas que representam mapeamentos de elementos de um tipo em elementos de outro tipo:

1. `overwrite`, que recebe um par `(identificador,valor)` e uma lista de pares `m`, retornando uma lista que representa o mapeamento que contém o par recebido mais os pares de `m` que não têm `identificador` como primeiro elemento;
2. `lookup`, que recebe um elemento `identificador` e uma lista de pares `m`, retornando `Erro` caso `identificador` não seja mapeado por `m`, e `valor` caso `m` mapeie `identificador` em `valor`.

Indique e explique o tipo das funções definidas. □

Questão 2 Considerando a classe

```
class Eq a where (==) ::  a -> a -> Bool
```

de Haskell, defina o tipo `Valor` da questão anterior como sendo uma instância de `Eq`. Comparações entre elementos de construtores diferentes de `Valor` devem dar falso, assim como comparações entre duas funções. □

Questão 3 Assumindo as definições a seguir para linguagens de expressões booleanas e inteiras, defina em Haskell o tipo de dados `Comando`, representando comandos de uma linguagem de programação simples com atribuição (de expressão inteira a uma variável), condicional de comandos (if-else onde a condição é uma expressão booleana), `while` (a condição é uma expressão booleana), e composição sequencial de dois comandos.

```
data ExpB = V | F | NEG ExpB | OU ExpB ExpB | VarB Id | AplB Operador ExpI

data ExpI = Lit Int | VarI Id | Apl Operador ExpI
```

Por exemplo, um programa nessa linguagem seria

```
x = 5; y = fat x; if (odd y) then z = x else z = y; while (>0 z) {x = +2 x; z = -10 z}
```

onde atribuições são representadas por `=`, composições sequenciais por `;`, e operações como `fat`, `>0` e `-10` são pré-definidas na linguagem. Defina também o tipo `Operador` como `String`. □

Questão 4 Com base na definição do tipo `Comando`, defina em Haskell a função `int` (de interpretar), que recebe um elemento desse tipo e dois mapeamentos de *strings* em valores (um representando o ambiente com as funções pré-definidas, e o outro a memória com as variáveis e seus valores). Como resultado, `interpretar` retorna um par formado por um valor (elemento do tipo `Valor` da Questão 1), representando o resultado da interpretação do comando, e um mapeamento representando a nova memória após a interpretação. Use funções definidas nas questões anteriores, e considere que `intEB :: [(Operador,Valor)] -> ExpB -> [(String,Valor)] -> Valor` existe e interpreta expressões booleanas, mas defina a função `intEI` que interpreta expressões inteiras. □

BOA SORTE E BOAS "FÉRIAS"!