

Paradigmas de Linguagens Computacionais

Exame Escrito

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

5 de fevereiro de 2013

Questão 1 Defina em Haskell as seguintes funções que trabalham com listas que representam mapeamentos de elementos de um tipo (como chave) em elementos de outro tipo (como valor):

1. **add**, que recebe um par (**chave**,**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 **chave** como primeiro elemento;
2. **domain**, que recebe uma lista de pares **m** e retorna uma lista com todos os primeiros elementos dos pares em **m**;
3. **image**, que recebe uma lista de pares **m** e retorna uma lista com todos os segundos elementos dos pares em **m**;
4. **apply**, que recebe um elemento **chave** e uma lista de pares **m**, retornando **Nothing** caso **chave** não seja mapeada por **m**, e **Just valor** caso **m** mapeie **chave** em **valor**.

Assuma a declaração do seguinte tipo: `data Maybe a = Nothing | Just a`. Pode também assumir que as listas recebidas não mapeiam a mesma chave em mais de um valor. Use compreensão de listas para definir **domain** e **image**. Indique e explique o tipo das funções definidas. □

Questão 2 Defina em Haskell o tipo de dados **Listas**, que representa uma linguagem de manipulação de listas de inteiros. Por exemplo, a interpretação do programa

```
X:=[2,4,9,1]; Y:=X; Z=[ e * 2 | e <- Y, e < 3]; Z
```

dá como resultado a lista `[4,2]`. Como ilustrado, os elementos de **Listas** podem ser listas constantes como `[2,4,9,1]`, atribuições representadas pelo símbolo `:=` e envolvendo uma *string* e um elemento de **Listas**, variáveis como o **Z** do final do programa e o **X** usado na segunda atribuição, composições sequenciais representadas pelo símbolo `;` e envolvendo dois elementos de **Listas**, e compreensões que envolvem um elemento de **Expressão** (como `e * 2`), um elemento de **Predicado** (como `e < 3`), um elemento de **Listas** (no exemplo a variável **Y**, podendo ser qualquer outro elemento), e uma *string* representando cada elemento da lista sendo manipulada (no exemplo `e`). Assuma que os tipos **Expressão** e **Predicado** estão definidos. □

Questão 3 Com base na definição do tipo **Listas**, defina em Haskell a função **avaliar**, que recebe um elemento desse tipo e um mapeamento de *strings* em listas (representando a memória, as variáveis de **Listas** e seus valores). Como resultado, **avaliar** retorna um par formado por um elemento de **Maybe**(**[Int]**), representando o resultado da avaliação do elemento de **Listas**, e um mapeamento representando a nova memória após a avaliação. Pode usar funções definidas nas questões anteriores. Assuma a existência de funções **avaliarP** e **avaliarE** que avaliam predicados e expressões, respectivamente. O tipo da primeira é **Predicado** \rightarrow **String** \rightarrow **[Int]** \rightarrow **[Int]**. O tipo da segunda é similar, recebendo **Expressão** como primeiro argumento. O segundo argumento é a *string* que representa cada elemento da lista manipulada (`e`, no exemplo da questão anterior). □