Enunciado

Atenção! Este enunciado ainda é provisório, podendo sofrer algumas alterações nos próximos dias. Poderá ser considerado definitivo quando este aviso for retirado.

Uma máquina de Turing pode ser vista como um modelo abstracto de um computador, importante para o estudo teórico da computabilidade, complexidade, etc. Este modelo foi desenvolvido pelo famoso matemático britânico Alan Turing nos anos 30. Na Wikipédia pode encontrar uma boa descrição deste formalismo. Existem vários simuladores na Web para estas máquinas, como por exemplo este. Este projecto é constituído por uma série de tarefas relacionadas com máquinas de Turing, a resolver usando a linguagem de programação Haskell.

Tarefa 1

A primeira tarefa consiste em desenvolver um simulador de uma máquina de Turing. Pretende-se um programa que, dado uma tabela de acção representada num ficheiro e a fita inicial, determine a fita resultante. Por exemplo, dado a tabela

```
1,1->2, ,D
2,1->2,1,D
2, ->3, ,D
3, ->4,1,E
3,1->3,1,D
4,1->4,1,E
4, ->5, ,E
5,1->5,1,E
5, ->1,1,D
```

e a fita `1111` o simulador deverá dar como resultado a fita `1111 1111`, dado que o objectivo deste programa é duplicar a sequência de uns presente na fita. Assume-se que inicialmente o cabeçote está posicionado no um mais à esquerda, e que a máquina está também no estado 1. O símbolo especial *branco* é aqui representado pelo espaço. Cada linha da tabela de acção tem a forma `s,a->t,b,d`, devendo ser entendida como: *se a máquina estiver no estado* `s` *e o símbolo lido pelo cabeçote for* `a`*, então escreva o símbolo* `b`*, mova o cabeçote na direcção* `d` *e mude o estado para* `t`. O simulador também deve permitir fazer a simulação passo a passo, mostrando os estados intermédios da fita e da máquina.

Tarefa 2

A segunda tarefa consiste em desenvolver um visualizador para tabelas de acção. Pretende-se que as tabelas sejam visualizadas como grafos direccionado, sendo os vértices usados para representar os diferentes estados da máquina e as arestas para representar as transições entre esses estados. Esta tarefa deverá ser implementada recorrendo ao Graphviz, um software para visualização de grafos. Mais concretamente, pretende-se um programa que traduza um ficheiro com uma tabela de acção num ficheiro no formato DOT, para posterior visualização com a ferramenta `dot` do Graphviz. Fica ao critério de cada grupo definir o aspecto final dos grafos.

Tarefa 3

A terceira tarefa consiste em desenvolver um optimizador de tabelas de acção. Considere, por exemplo, a seguinte tabela:

```
1,1->2, ,D
2,1->2,1,D
2, ->3, ,D
3, ->4,1,E
3,1->3,1,D
4,1->4,1,E
4, ->5, ,E
5,1->5,1,E
5, ->8,1,D
6,1->7,1,D
7,1->6, ,E
8,1->9, ,D
9,1->9,1,D
9, ->3, ,D
```

Esta tabela é equivalente à apresentada acima, ou seja, quando executada produz exactamente o mesmo efeito: duplicar a sequência de uns na fita. No entanto, consome muito mais espaço que a anterior. Nesta tarefa pretende-se um programa que dada uma tabela de acção escrita num ficheiro determine uma tabela equivalente mais pequena. Depois de identificar o tipo de optimizações que podem ser feitos, devem desenvolver algoritmos que, na medida do possível, as implementem.

Tarefa 4

A última tarefa consiste em desenvolver um pequeno compilador para máquinas de Turing. Dada uma função com uma variável definida usando somas e constantes positivas, o compilador deve devolver uma tabela de acção que, dada uma fita com um número representado em unário, calcule o valor dessa função para esse número. Por exemplo, dada a função `f(x)=(x+2)+x` e a fita `111`, a tabela de acção deverá calcular o valor `11111111`. Note que o corpo da função pode ser representado facilmente em Haskell usando o seguinte tipo de dados, para representar expressões aritméticas:

```
data Exp = Var | Const Int | Soma Exp Exp
```

A função acima seria representada pelo valor `Soma (Soma Var (Const 2)) Var`. Opcionalmente, podem melhorar o compilador por forma a processar expressões envolvendo números negativos e subtracções.

Relatório

Deverá também ser escrito um pequeno relatório em LaTeX descrevendo os tipos de dados, funções (e respectivos algoritmos) usados para resolver cada uma das tarefas. O relatório deverá ser no formato `article`, com tamanho de letra `10pt`, tamanho de página `a4wide`, e ter no máximo 8 páginas. Devido ao espaço limitado, não é recomendável incluir todo o código no relatório: apenas devem incluir os extractos que julguem mais interessantes.

Logística

O projecto deverá ser realizado em grupos de 3 alunos, e apresentado aos monitores na semana de 11 a 15 de Janeiro de 2010. Para a apresentação deverão trazer o código desenvolvido e o relatório impresso (idealmente em frente-e-verso). Mais detalhes sobre o processo de submissão serão anunciados aqui brevemente.

As cotações máximas que podem ser obtidas em cada uma das tarefas é a seguinte:

Tarefa	Cotação
Tarefa 1	7
Tarefa 2	3
Tarefa 3	3
Tarefa 4	3
Relatório	4

As notas dos elementos do grupo podem ser diferentes, de acordo com a sua prestação durante a apresentação. Caso se detectem cópias entre os trabalhos, todos os membros dos grupos envolvidos ficarão imediatamente reprovados.

FAQ

Esta secção será actualizada regularmente com as dúvidas mais frequentes que nos forem colocadas.