Trabalho Prático nº1

Métodos de Programação I

2005/2006

1 Introdução

Este é o enunciado do primeiro trabalho de Métodos de Programação I que se destina a aplicar o conceito de *Monad* na programação em *Haskell*. O tema do trabalho é a construção de uma aplicação que permita jogar o puzzle *Su-Doku* que é publicado em vários jornais diários (ver `http://www.sudoku.com` para uma apresentação das regras e de estratégias para a sua resolução).

Convém referir que os monads em programas como o que se propõe surgem de duas formas distintas:

- para certa funcionalidade, o próprio desenho da linguagem *Haskell* força-nos a utilização de determinados monads (e.g. monad `IO` para realizar operações de entrada/saída).
- a utilização dos monads pode-se ainda justificar com vista a estruturar o programa, permitindo a construção de um programa mais claro e mais fácil de manter.

Na avaliação deste trabalho serão levadas em linha de conta estas duas vertentes da utilização dos Monads.

Divide-se a apresentação do trabalho em duas tarefas:

- Construção de um editor do jogo *Su-Doku*: um programa que matém a informação referente ao tabuleiro de jogo e que pode ser preenchido pelo utilizador. Esta componente do projecto terá um peso de 12 valores em 20.
- Codificação de estratégias de resolução: o programa deve suportar um conjunto de comandos que se destinam a auxiliar o jogador na resolução do jogo. No limite, deve resolver completamente o puzzle.

2 Editor do jogo

Pretende-se realizar uma aplicação em Haskell que permita a um utilizador resolver um puzzle *Su-Doku*. Para tal codifica-se um interpretador que aceite

os seguintes comandos[1]:

read – lê ficheiro com a informação de um tabuleiro *Su-Doku*. Este ficheiro deverá conter 9 linhas, cada uma das quais com 9 caracteres (digitos de 1 a 9 ou o espaço que marca uma casa vazia). Um possível tabuleiro poderá ser:

5		8	2		4	6		1
9		6				3		2
3			7		2			8
		4		6		7		
2			8		5			4
6		5				4		3
1		2	3		7	5		6

show – visualiza o estado actual do tabuleiro.

set – permite realizar uma jogada no tabuleiro actual. Deve ler as coordenadas de uma posição e o valor a inserir. Deve falhar se a jogada conduzir a um tabuleiro inválido (e.g. com dígitos repetidos numa linha, etc.).

undo – permite voltar ao estado anterior à última jogada.

quit – sai do interpretador.

3 Estratégias de solução

Com vista a auxiliar o utilizador na resolução do puzzle, o programa dispõe ainda de um conjunto de comandos que se destinam a fornecer informação útil à resolução do jogo e a permitir alguma forma de mecanização.

cands – mostra conjunto de alternativas de jogadas admissíveis para uma dada posição (jogadas que não conduzem a um tabuleiro inválido). Estes são designados por *conjuntos candidatos* de uma dada posição.

showCands – mostra todos os conjuntos candidatos das posições livres.

simplify – realiza todas as jogadas que não pressupõe escolhas (e que, portanto, não comprometem a realização do puzzle). Note que neste ponto é importante o "grau de inteligência" que se incorpora no programa: um programa que só codifique estratégias muito básicas pode não ser capaz de prosseguir com a resolução de puzzle que dispõe de uma única solução.

autosolve – resolve o puzzle de forma automática.

[1] Pode considerar ainda outros comandos para incorporar funcionalidade adicional.

4 Valorização

Para notas superiores a 17 valores deve realizar alguma das seguintes extensões do trabalho prático:

1. Generalizar o programa para permitir a resolução da variante dos puzzles Su-Doku designados por *Samurai* (`http://sudoku.top-notch.co.uk/gattai5.asp`).
2. Construir uma interface gráfica para o jogo. Sugere-se para o efeito a utilização da biblioteca *WxHaskell* (`http://wxhaskell.sourceforge.net/`).