3º Trabalho Prático

Métodos de Programação I
LESI/LMCC
Universidade do Minho

Ano Lectivo de 2002/2003

1 Preâmbulo

Este trabalho deve ser realizado por grupos com um máximo de três alunos. O relatório do trabalho deve ser submetido electronicamente até ao dia 10 de Janeiro de 2003, de acordo com as instruções divulgadas na página da disciplina (WWW). Simultaneamente, deverá ser realizada a entrega de uma cópia em papel na Recepção do *Departamento de Informática* (ext. 4430).

2 O problema

Neste trabalho pretende-se explorar a capacidade expressiva dos tipos indutivos em Haskell para representarem situações da vida real. O tema escolhido para esse efeito é o da composição e formatação de um jornal electrónico, com notícias, imagens, etc., a ser publicado em formato HTML calculado a partir de uma sua descrição abstracta em Haskell.

Tal como é habitual nos trabalhos desta disciplina, fornece-se de seguida um "kit" para arranque do projecto. Este enunciado concluir-se-á com sugestões para valorização do trabalho a realizar.

3 Material fornecido como "kit" para o projecto

Entre na página da disciplina e descarregue, descomprimindo-o (*eg.* via `unzip`), o ficheiro *`mpi0203mp.zip`* que contém o respectivo material pedagógico. Para além de outros ficheiros relevantes para a disciplina, deverá obter:

1. `mpi0203t3.lhs` - trata-se do ficheiro que está a ler neste momento, escrito em "*literate* HASKELL". Isto significa que:

   - se o carregar no HUGS, este interpretador carregará o código HASKELL nele contido e interpretá-lo-á. Sugestão: invoque o HUGS e experimente `:l mpi0203t3.lhs`. Uma vez iniciada a sessão, escreva

     ```
     Main> printJournal example
     ```

     Deverá obter a resposta

     ```
     Output HTML written into file 'journal.html'
     ```

     Poderá encontrar o ficheiro `journal.html` na sua directoria de trabalho, onde a aplicação o gravou. Se o abrir no seu "browser" preferido, verificará que se trata de um "arremedo" de página de um jornal electrónico.

   - se o processar via LaTeX (ou PDFLaTeX) obterá este mesmo documento em *PDF*. Sugestão: experimente

     ```
     latex mpi0203t3.lhs
     dvips -o mpi0203t3.ps mpi0203t3
     ps2pdf mpi0203t3.ps
     ```

ou, mais simplesmente,

```
pdflatex mpi0203t3.lhs
```

Se não está habituado a LaTeX: em LINUX, faz parte da distribuição standard e é só experimentar; em Windows, sugere-se a instalação de *MiKTeX* (`http://www.miktex.org/`).

2. `mpi0203.sty` - trata-se de um ficheiro importado por `mpi0203t3.lhs`, contendo funções de processamento de texto expressas em sintaxe LaTeX (como poderá ver, é normal programar *funcionalmente* em LaTeX).
3. `mpi0203t3.pdf` - trata-se do resultado do processamento de `mpi0203t3.lhs`, fornecido para sua conveniência, caso não tenha LaTeX acessível.
4. `journal.html` - o ficheiro acima referido, fornecido para sua conveniência, caso não consiga — por qualquer motivo — produzi-lo.
5. Vários ficheiros `*.jpg` - imagens usadas na composição do jornal exemplo.

O trabalho consiste em editar o ficheiro `mpi0203t3.lhs`, acrescentando-lhe não só texto seu (por exemplo, preâmbulo, conclusões, detalhes de execução, pistas para trabalho futuro, etc.) mas todo o código HASKELL que vai desenvolver, introduzindo-o pela ordem que lhe parecer mais natural.

4 Análise do "kit" do projecto

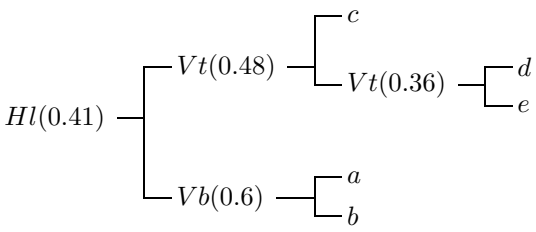
Comece por analisar o código Haskell dado em anexo (pág.3), que é fornecido como primeira abordagem aos requisitos do trabalho. Nele encontra o tipo $Sheet$ (="página de jornal") que é baseado num tipo indutivo X, cuja estrutura de base é semelhante a

$$X \quad \cong \quad A + B \times X^2$$

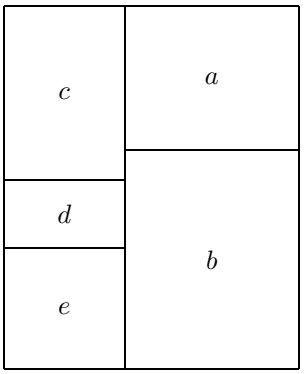
e que exprime a partição de um rectângulo (a página tipográfica) em vários subrectângulos (as caixas tipográficas a encher com texto ou imagens), segundo um processo de partição binária, na horizontal ou na vertical. O tipo $Mode$ especifica quatro variantes de partição. O seu argumento deverá ser um número de 0 a 1, indicando a fracção da altura (ou da largura) em que o rectângulo é dividido, a saber:

- `Hr i` — partição horizontal, medindo i a partir da direita
- `Hl i` — partição horizontal, medindo i a partir da esquerda
- `Vt i` — partição vertical, medindo i a partir do topo
- `Vb i` — partição vertical, medindo i a partir da base

Por exemplo, a partição dada na figura da página 3 corresponde à partição de um rectângulo 780×960 de acordo com a seguinte árvore de partições:



As caixas delineadas por uma partição (como a dada acima) correspondem a folhas da árvore de partição e podem conter texto ou imagens. É o que se verifica no objecto `example` da secção A.5 que, processado por `sheet2html` (secção A.3) vem a produzir o ficheiro `journal.html`.



5 O que se pretende

Infere-se da análise da secção anterior que o código Haskell fornecido como "kit" para arranque deste trabalho não está suficientemente estruturado em termos dos combinadores `cata`, `ana`, `hylo` estudados nesta disciplina. O que se pretende é, então:

1. A construção de uma biblioteca "pointfree" [1] com base na qual o processamento ("pointwise") já disponível possa ser redefinido.
2. A evolução da biblioteca anterior para uma outra que permita partições n-árias (para *qualquer* n finito) e não apenas binárias [2].
3. Uma parametrização adequada da biblioteca anterior por forma a acomodar não apenas texto corrido e imagens, mas também "links" e "fonts".
4. Funcionalidade adicional para ampliação/redução de caixas tipográficas, mudanças de "font", etc.

6 Sugestões para Valorização

Qualquer contribuição na direcção das questões seguintes:

1. Como controlar "fonts", tamanhos de imagens e impedir a sua distorção?
2. Como produzir texto formatado?
3. Quão prática é esta maneira de produzir um "layout" de jornal? Que outras estruturas indutivas podem ser usadas para o mesmo efeito? [3]
4. Sofisticação gráfica: elaborar o HTML produzido em termos de "design" por forma a aproximar o resultado de `printJournal` de um jornal electrónico real.

A Anexo

O código Haskell fornecido neste anexo é baseado no módulo

```
import IO
```

[1] A desenvolver de forma análoga a outras bibliotecas que conhece (*eg.* `BTree.hs`, etc).

[2] Repare que é a falta desta capacidade expressiva que origina, no "kit" actual, a definição das funções auxiliares da secção A.4, por exemplo.

[3] Havendo tempo, pode ser útil a consulta da referência *CAD Tool Extension for Formal Building Description Language. Advances in Engineering Software*, 29(7-9):571–586, 1998.

A.1 Tipos de dados

```
data X a b i = Leaf (Unit a b) | Node (Mode i) (X a b i) (X a b i) deriving Show

data Sheet a b i = Rect (Frame i) (X a b i) deriving Show

data Mode i = Hr i | Hl i | Vt i | Vb i deriving Show

data Frame i = Frame i i deriving Show

-- leafs info

data Unit a b = Image a
              | Text b deriving Show
```

A.2 Funções da API (IO)

A API (=“Application Program Interface”) deste módulo consiste nas funções seguintes:

```
printJournal :: Sheet String String Double -> IO ()
printJournal = write . sheet2html

write :: String  -> IO ()
write s = do writeFile "journal.html" s
             putStr "Output HTML written into file ‘journal.html’"
```

A.3 Geração de HTML

```
sheet2html (Rect (Frame w h) y) = htmlbegin ++ x2html y (w,h) ++ htmlend

x2html :: X String String Double -> (Double, Double) -> String
x2html (Leaf (Image img)) (w,h)= "<img src=\"" ++
                                 img ++
                                 "\" width  =\"" ++
                                 show(w) ++
                                 "\" height =\"" ++
                                 show(h) ++
                                 "\">"
x2html (Leaf (Text txt)) _ = txt
x2html (Node (Vt i) x1 x2) (w,h) = "<table cellpadding=\"0\" width=\"" ++
                                        show(w) ++
                                        "\" height =\"" ++
                                        show(h) ++
                                        "\">\n<tr>\n<td width=\"" ++
                                        show(w) ++
                                        "\" height =\"" ++
                                        show(h*i) ++
                                        "\">\n" ++
                                        x2html x1 (w, h*i) ++
                                        "\n</td>\n</tr>" ++
                                        "\n<tr>\n<td width=\"" ++
                                        show(w) ++
```

```
                                        "\" height =\"" ++
                                        show(h*(1-i)) ++
                                        "\">\n" ++
                                        x2html x2 (w, h*(1-i)) ++
                                        "\n</td>\n</tr>\n</table>"
x2html (Node (Hl i) x1 x2) (w,h) = "<table cellpadding=\"0\" width=\"" ++
                                        show(w) ++
                                        "\" height =\"" ++
                                        show(h) ++
                                        "\">\n<tr>\n<td width=\"" ++
                                        show(w*i) ++
                                        "\" height =\"" ++
                                        show(h) ++
                                        "\">\n" ++
                                        x2html x1 (w*i, h) ++
                                        "\n</td>\n<td width=\"" ++
                                        show(w*(1-i)) ++
                                        "\" height =\"" ++
                                        show(h) ++
                                        "\">\n" ++
                                        x2html x2 (w*(1-i), h) ++
                                        "\n</td>\n</tr>\n</table>"
x2html (Node (Vb i) x1 x2) m = x2html (Node (Vt (1 - i)) x1 x2) m
x2html (Node (Hr i) x1 x2) m = x2html (Node (Hl (1 - i)) x1 x2) m

htmlbegin = "<html>\n" ++
            "<head>\n" ++
            "<title>MP-I/0203: Sheet2HTML</title>\n" ++
            "</head>\n\n" ++
            "<body BGCOLOR=\"#F4EFD8\">\n" ++
            "<div align=\"center\">\n"

htmlend = "\n</div>\n</body>"
```

A.4 Funções auxiliares

```
twoVtImg a b = Node (Vt 0.5) (Leaf (Image a)) (Leaf (Image b))

fourInArow a b c d =
        Node (Hl 0.5)
          (Node (Hl 0.5) (Leaf (Text a)) (Leaf (Text b)))
          (Node (Hl 0.5) (Leaf (Text c)) (Leaf (Text d)))
```

A.5 Dados para teste

```
example :: (Fractional i) => Sheet String String i
example =
   Rect (Frame 420 450)
    (Node (Vt 0.01)
      (Node (Vt 0.1)
         (Leaf (Image "dn-logo.jpg"))
```

```
   (fourInArow "5?-feira, 12 de Dezembro 2002" "Ano 1" "Nr. 1" "Suplemento MP-I/02
(Node (Vt 0.55)
    (Node (Hl 0.65)
        (Leaf (Text
           "PORTO - A chegada do Metro. A Linha Azul vai dar inicio ao metro na re
        (Leaf (Image
            "mapa_metro.jpg")))
    (Node (Hl 0.32)
        (twoVtImg
            "manuel_oliveira_rt.jpg"
            "manoel_de_oliveira_adn.jpg")
        (Leaf (Text
            "MANOEL DE OLIVEIRA - ‘‘Patriarca’’ do cinema português celebra 94 anc
```