

Micro- \LaTeX em Haskell

LEI — LI1/1112

Dez. 2011

Apresenta-se de seguida uma pequena biblioteca em Haskell que representa nesta linguagem a estrutura básica, muito simplificada, da linguagem \LaTeX para preparação de texto.

Módulo e dependências

```
module TeX where
    -- This library provides for LaTeX outputs, which can be written to
    -- files. Only a (very!) tiny subset of LaTeX is considered.
import Data.List
```

Tipos básicos

```
data TeX =
    Cmd TeXCmd
  | NoArg String
  | INT Int
  | TabRow [TeX]
  | STR String
  | Opt{t :: TeX}
  | TeXs [TeX] deriving Show
data TeXCmd = TeXCmd{op :: String, str :: [TeX]} deriving Show
```

Estruturas básicas

```
latex :: String → TeX → TeX
latex c t = TeXs [documentclass c, document t]
documentclass :: String → TeX
documentclass c = cmd "documentclass" [STR c]
document :: TeX → TeX
```

```

document = (envir "document")
section :: String → TeX
section s = cmd "section" [STR s]
itemize :: [TeX] → TeX
itemize = (envir "itemize") · TeXs · pref (NoArg "item")
quote :: TeX → TeX
quote = envir "quote"
description :: [TeX] → TeX
description = (envir "description") · TeXs
enumerate :: [TeX] → TeX
enumerate = (envir "itemize") · TeXs · pref (NoArg "item")
tabular :: String → [TeX] → TeX
tabular cs = envir1 "tabular" cs · TeXs
descitem :: TeX → TeX → TeX
descitem l t = TeXs [cmd "item" [Opt l], t]
descitemstr :: String → String → TeX
descitemstr l t = TeXs [cmd "item" [Opt (STR l)], STR t]

```

Serialização

Funções de conversão para texto:

```

texTeX2txt :: TeX → [Char]
texTeX2txt (Cmd c) = texTeXCmd2txt c
texTeX2txt (NoArg s) = escape s ++ " "
texTeX2txt (TabRow r) = concat (intersperse "&" (map texTeX2txt r))
texTeX2txt (STR c) = texTeXstring c
texTeX2txt (TeXs x) = concat (map texTeX2txt x)
texTeX2txt (Opt a) = texTeX2txt a
texTeX2txt (INT i) = show i
texTeXCmd2txt :: TeXCmd → [Char]
texTeXCmd2txt (TeXCmd o l) =
  (escape o) ++ concat (map arg l) where
    arg (Opt t) = sqbraces (texTeX2txt t)
    arg t = braces (texTeX2txt t)
texTeXstring :: [Char] → [Char]
texTeXstring = concat · map texTeXchar where
  texTeXchar ' _ ' = "\\ _"
  texTeXchar c = [c]

```

Abreviaturas e funções auxiliares

```

cmd :: String → [TeX] → TeX
cmd o x = Cmd (TeXCmd o x)

string :: String → TeX → TeX
string s t = TeXs [STR s, t]

envir :: String → TeX → TeX
envir n x = TeXs
  [ cmd "begin" [STR n]
  , TeXs [x]
  , cmd "end" [STR n]
  ]

envir1 :: String → String → TeX → TeX
envir1 n p x = TeXs
  [ cmd "begin" [STR n, STR p]
  , TeXs [x]
  , cmd "end" [STR n]
  ]

pref :: a → [a] → [a]
pref p l = concat [[p, x] | x ← l]

verb :: [Char] → [Char]
verb s = "\\verb!" ++ s ++ "!"

braces :: [Char] → [Char]
braces s = "{" ++ s ++ "}"

sqbraces :: [Char] → [Char]
sqbraces s = "[" ++ s ++ "]"

escape :: [Char] → [Char]
escape s = "\\n\\\\" ++ s

strTabRow :: [String] → TeX
strTabRow = TabRow · map STR

hline = NoArg "hline"

newl = NoArg "\\\"

nlhline = TeXs [newl, hline]

wrap a as = [a] ++ (intersperse a as) ++ [a]

```

Com estas funções auxiliares é mais fácil escrever expressões que representam estruturas \LaTeX em Haskell. Por exemplo, o texto em \LaTeX

```

\documentclass{article}
\begin{document}
\begin{tabular}{|c|c|}
\hline um&dois
\\

```

```

\hline onze&doze
\\
\hline
\end{tabular}
\end{document}

```

correspondente à produção da tabela

um	dois
onze	doze

pode obter-se correndo, no GHCi, a expressão

```
(writeFile "a.tex" · texTeX2txt · latex "article") exemplo
```

onde

```

exemplo =
  tabular " | c | c | "
    [ hline
    , strTabRow ["um", "dois"]
    , newl, hline
    , strTabRow ["onze", "doze"]
    , newl, hline
    ]

```

em lugar de se ter que escrever a expressão

```

TeXs [Cmd (TeXCmd {o = "begin", str = [STR "tabular", STR "|c|c|"]}), TeXs
[TeXs [NoArg "hline", TabRow {r = [STR "um", STR "dois"]}, NoArg "\\ ",
NoArg "hline", TabRow {r = [STR "onze", STR "doze"]},
NoArg "\\ ", NoArg "hline"]],
Cmd (TeXCmd {o = "end", str = [STR "tabular"]})]

```

que corresponde à representação interna de *exemplo* nos tipos básicos dados.