LaTeX2e 用 Haskell スタイル

Manuel M. T. Chakravarty
Institute of Information Sciences and Electronics
University of Tsukuba, Japan

chak@cse.unsw.edu.au
www.cse.unsw.edu.au/~chak

Version 1.0e (for Version 1.0e of the style file)

概要

ソースコードをそのまま貼り付けて verbatim よりはマシに整形する目的には listings 環境 (と、Haskell のための追加設定) が便利である。(ソースコードそのままではなく) 編集コストをかけて Haskell ソースをより美しく組版する環境を `haskell.sty` が提供する。

1 誰のためのもの?

大量のコード断片を `verbatim` 環境で整形するのは見苦しいし、添え字を使ったりコメントを (等幅でない) プロポーショナルフォントで書いたりするのが難しい。一方、例えば TeX の数式モードを使うのは、複数文字の名前を正しくカーニングしたり、関数適用の意味の名前の並びを整形するのにするのに追加のマクロを必要とする。この `haskell` スタイルは LaTeX [Lam94] 中で Haskell [H+92, P+97] プログラムを簡単に整形する環境とマクロを提供する。このスタイルは Haskell 用に特化しているが[*1]、ML や Nesl といった他の関数型言語にとっても有用であろう。

有名な *map* 関数は次のように整形される。

$$
\begin{array}{lll}
map & :: (\alpha \to \beta) \to [\alpha] \to [\beta] & \text{— 型宣言 type assertion} \\
map\ f\ [] & = [] & \text{— [] は空リスト} \\
map\ f\ (x : xs) & = f\ x\ :\ map\ f\ xs & \text{— :は中置きリスト構成子}
\end{array}
$$

この定義の元で *map* (+1) [1, 2, 3] を評価すると [2, 3, 4] になる。

上の例のソースを以下に示す。

[*1] Joe:specifically geared towards Haskell

```
有名な\<map\>関数は次のように整形される。
%
\begin{haskell*}
  map          &::& (\alpha\to\beta)\to[\alpha]\to[\beta]
  &\hscom{型宣言 type assertion}\\
  map f []     &=& []
  &\hscom{\<[]\> は空リスト}\\
  map f (x:xs) &=& f x : map f xs
  &\hscom{\<:\> は中置きリスト構成子}
\end{haskell*}
%
この定義の元で\<map (+1) [1, 2, 3]\>を評価すると\<[2, 3, 4]\>になる。
```

利点はたくさんある。[*2] Haskell 文中で`\alpha`のような数式モードのコマンドが使用可能である。にもかかわらず、空白文字は式の並び (つまり Haskell の関数適用) を表すのに使え、また複数文字の名前も正しく整形される。`$ffoo$`と`\<ffoo\>`を整形した $ffoo$ と *ffoo* を比べてみて欲しい。

以降の例について、そのソースは付録でまとめて示す。

2 基本的な使用法

2.1 Haskell モード

`haskell` 環境とコマンド`\<` から `\>` の間で Haskell モードに入る。予約語は`\hskwd` により **bold face** で整形できる。`haskell` 環境の中では、桁位置は桁合わせ文字`&`で定義される。通常は関係記号を桁合わせ文字で囲む必要はなく、($\mathcal{AMS}$-LaTeX の `align` 環境のように) 関係記号の右側[*3]だけに桁合わせ文字を置けば充分である。しかし、型宣言の :: とそれに対応する等式の = のように、大きさの違う関係記号を揃えたいような場合には、アスタリスク付きの`haskell*`環境を使う。これは一つめと二つめの桁合わせ文字に囲まれた内容 (material) を中揃えする。

2.2 コメント

一行コメントを整形するには、1 引数コマンド`\hscom` を使う。

[*2] There are a number of points worth noting.

[*3] サンプルを見るに、`&`が先、関係記号が後になっていて逆である。

hscom

```
data Maybe α = Nothing    — 結果なし
             | Just α     — 普通に α 型の値
```

2 行めはコメントの中でさらに Haskell モードに入っている例である。これは、コメント中に名前を書いたり、プログラム断片を引用したりするとき、書体をプログラムと同じにできるという意味で特に重要である。

アルゴリズムの詳細を省略して、形式的でない説明で代用させたい場合などには`\hsinf` を使う。

hsinf

```
main = do
         config ← ⟨ 設定ファイルを読み込む ⟩
         result ← ⟨config に従って入力を処理 ⟩
         putStr result
```

2.3 数式モードからの機能

LaTeX の数式モードの記号、添え字、上付文字などのあらゆる機能が Haskell モードでも使える。これには TeX の関係記号も含む、例えば $\geq$ や $\neq$ など、また型変数に使うギリシャ文字も使える。

2.4 補足

細かいコマンドがいくつか undocumented なので、付録 A も参照のこと。

2.5 位置揃え Aligned Material

関数本体が大きくなると、定義を視覚的に構造化するために部分的に位置揃えする必要がある。コマンド`\hsalign` は任意の桁数での位置揃えを行う。Haskell 標準ライブラリの *transpose* 関数を考えよう。

hsalign

```
transpose :: [[a]] → [[a]]
transpose = foldr
              (λ xs xss → zipWith (:) xs (xss ++ repeat []))
              []
```

桁揃えが必要になるいくつかの構文要素について、`\hsalign` を内部で使ったマクロがあらかじめ用意してある。例えば、**let** 式を整形する`\hslet` コマンドがある。

hslet

```
foo a = let
          x = 1
          y = 2
        in
        x + y
```

if-then-else 式を整形する`\hsif` マクロもある。下の例では桁揃えの入れ子を行っている。

hsif

```
foo a = if a == 0 then
          let
            x = a + 1
          in
          x
        else
          a
```

桁揃えが大きくなりすぎて、改行によって外側の桁揃えを破棄する場合がある。これを整形するには`\hsbody` を使う。

hsbody

```
calculateNextPos                    :: Pos → Mover → State → [Pos]
calculateNextPos oldPos move state =
  map(λ (lowerHalf, upperHalf) → (upperHalf, lowerHalf)    — この行
      (compare oldPos (move state))
```

\hsbody コマンドの定番の使い方は **where** 節であるが、さらにそれをマクロにした\hswhere コマンドがある。

hswhere

$$
\begin{aligned}
&\mathit{partition} \quad :: (a \to \mathit{Bool}) \to [a] \to ([a],[a])\\
&\mathit{partition}\ p\ xs = \mathit{foldr}\ \mathit{select}\ ([],[])\ xs\\
&\quad \textbf{where}\\
&\qquad \mathit{select}\ x\ (ts,fs) \mid p\ x \quad\quad\ = (x:ts,fs)\\
&\qquad\qquad\qquad\quad \mid \mathit{otherwise} = (ts,\ x:fs)
\end{aligned}
$$

3 上級の機能

複雑な文書を整形するには、Haskell モードのより微妙な点を理解する必要がある。一般に、技巧的に Haskell モードを使おうとする、または独自のコマンドを定義しようとするとき、Haskell モードでは空白が重要であることに留意が必要である。すなわち、Haskell モードは TEX の \obeyspaces を使う。これが TEX の挙動を微妙に変化させる。

3.1 コマンドの定義

Haskell モードの中で使うコマンドは、\newcommand の代わりに\hscommand で定義する。\hscommand の引数は\newcommand と同様である。(ただし、\hscommand コマンドに* 版はない。) \hscommand の動作は少し異なる、例えば、続けて書くことが関数適用を表すとするために、コマンド定義の本体での空白を処理する点に関して。

3.2 『真の』数式モードに入る

Haskell モードは、数式モードの特殊な文字や下付き添え字などが使えるという意味で、数式モードによく似ている。しかし、純粋な数式モードに入る方がよい場合がある。Haskell の途中で、LATEX 標準の数式モードに入るには、コマンド \(と \) を使う。これはマクロを定義するとき[*4]に有用であり、例えば、マクロ\hscom はこれを使って定義されている。

3.3 変換規則で使う

変換規則の中などでは、let-in 構造のインデントを揃えるといったような複雑な式を記述したい。これは位置揃えのためのオプション引数を指定することで解決できる。例えば\hslet[c]{···} の

[*4] \(の前に \relax を置く必要がある場合がある。マクロが&の後ろで使われる場合である。

ように。

position

$$\begin{array}{l}\textbf{let}\\ \quad x = e\\ \textbf{in}\\ C[x]\end{array} \Longrightarrow C[e]$$

標準では、ボックスは垂直方向には最初の行の baseline で揃えられる。(オプション `t` に対応する。) `c` を指定すると中央揃え、`b` では最終行の baseline で揃えられる。

3.4 TEX の数式パーザ

TEX の数式モードは数式の構造をある程度理解する能力を持つ。分析された構造は数式の要素間に挿入される空白に影響する。次の 2 つの例を比較してみよう。

`3+4` $\Longrightarrow 3+4$

`3{+}4` $\Longrightarrow 3+4$

前者 TEX は + が二項演算子で 3 と 4 に適用されていると理解している。しかし後者はそうではない。詳細は *The TEXbook* [Knu86] にある。

4 Copyleft

参考文献

[H+92] P. Hudak et al. Haskell special issue. *ACM SIGPLAN Notices*, 27(5), May 1992.

[Knu86] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1986.

[Lam94] Leslie Lamport. *LATEX: a Document Preparation System*. Addison-Wesley, second edition, 1994.

[P+97] John Peterson et al. Haskell 1.4: a non-strict, purely functional language. Research report, Yale University, April 1997.

翻訳と蛇足の追加：大久保 ohkubo@ist.aichi-pu.ac.jp

付録 A　コマンド一覧

コマンド	引数	機能
\< \>	–	インライン Haskell モード
haskell, haskell*	–	ブロック Haskell 環境
\hscom	1	コメント
\hskwd	1	予約語
\hsinf	1	非形式的説明
\hschar	1	文字リテラル 'c'
\hsstr	1	文字列リテラル "hoge"
\hsfrom	0	左向き矢印 ← ちなみに右向きは \to
\hsalign	1	位置揃え
\hsnoalign	1	位置揃えしないで出力
\hsbody	1	インデント位置を 1 段目に強制的に設定
\hscommand	1	コマンド定義
\hslet	2	let 式
\hsif	3	if 式
\hscase	2	case 分岐
\hswhere	1	where 節
\hsdo	1	do ブロック
\hsapp	0	リストの連結 ++
\hsifix	1	関数を中置演算子にする 1 `*foo*` 2
\hsus	0	パターンマッチのワイルドカード _
\hstchar	1	Hutton 本風文字リテラル 'c'
\hststr	1	Hutton 本風文字列リテラル "hoge **ほげ**"
\hsaligned	0	\hsalign とその派生コマンドの終了時の改行を補正 (暫定版)
\vs	0	Visual Space '␣'
\bs	1	Backslash "\n" 引数なしでも使える
\hsbind	0	⋙=
\hsbin	0	⋙
\hseq	0	等しい ==

\hschar, \hsstr は使わないとイタリックで出力されるので利用するよう修正するほうがよい。otf パッケージと相性が悪いかもしれない。

\hslet, \hsif, \hscase, \hsdo はもう一つオプションパラメータを先頭にとる。その 1 引数

めは\hsalign のオプションパラメータになる。これは 3.3 節にあるように高さ方向の位置揃えを t, b, c で指定する。

付録 B　サンプルのソース

■hscom

```
\begin{haskell*}
  \hskwd{data} Maybe \alpha
  & = & Nothing      &\hscom{結果なし}\\
  & | & Just \alpha  &\hscom{普通に\<\alpha\>型の値}
\end{haskell*}
```

■hsinf

```
\begin{haskell}
  main &= \hsdo{
    config \hsfrom \hsinf{設定ファイルを読み込む}\\
    result \hsfrom \hsinf{\<config\>に従って入力を処理}\\
    putStr result
    }
\end{haskell}
```

■hsalign

```
\begin{haskell*}
transpose &::& [[a]]\to [[a]]\\
transpose &=&
\hsalign{
  foldr\\
  \quad(\lambda xs xss\to zipWith (:) xs (xss\hsapp repeat []))\\
  \quad[]
  }
\end{haskell*}
```

■hslet

```
\begin{haskell}
  foo a = \hslet{
```

```
    x &= 1\\
    y &= 2
    }{%
    x+y
    }
\end{haskell}
```

■hsif

```
\begin{haskell}
  foo a = \hsif{a == 0}{%
    \hslet{%
      x = a + 1
      }{%
      x}
    }{%
    a}
\end{haskell}
```

■hsbody

```
\begin{haskell*}
  calculateNextPos &::& Pos\to Mover\to State\to [Pos]\\
  calculateNextPos oldPos move state &=&\relax
  \hsbody{%
    map &(\lambda (lowerHalf, upperHalf)\to (upperHalf, lowerHalf) \hscom{この行}\\
        &(compare oldPos (move state))
    }
\end{haskell*}
```

■hswhere

```
\begin{haskell*}
partition       &::& (a\to Bool)\to [a]\to ([a],[a])\\
partition p xs  &=& foldr select ([],[]) xs
\hswhere{%
  select x (ts,fs) &\mid p x       &= (x:ts,fs)\\
```

```
                    &\mid otherwise &= (ts, x:fs)
  }
\end{haskell*}
```

■position

```
\begin{haskell}
  \hslet[c]{%
    x &= e
    }{C[x]}
  \Longrightarrow
  C[e]
\end{haskell}
```

付録 C　問題

- オリジナルのままだと \< というコマンドが多重定義になって怒られる。そのため \renewcommand に変更している。これは tabular 環境のコマンド。
- 一部の記号が courier で出力されない。アンダーライン文字の横幅が短いため、もう少し強調するように\hsus を用意している。結果を PDF 出力した後でテキストモードでコピーペーストしたときにアンダーライン文字が出ない点が不便。

  \verb　　:　{ }　[ ]　_

  haskell style　:　{ }　[ ]　　-, _ (アンダーライン文字, \hsus)
- \hsalign とその派生コマンドでインデントを行った後、次の行との間隔が狭くなる。これを補正するために\hsaligned を定義したが、現状は逆に間隔が広がりすぎるので暫定版としてある。(\hsnoalign で解決できる？)