Haskell for LaTeX2e

Manuel M. T. Chakravarty
School of Computer Science and Engineering
University of New South Wales, Australia

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

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

1 What's it good for?

Setting large pieces of code in an `verbatim` is ugly and makes it difficult to uses subscripts or set comments in a proportional type face. On the other hand, the use of, for example, TeX's math mode requires the use of additional macros to achieve proper kerning in multi-letter identifiers and to typeset application by juxtaposition. The `haskell` style provides environments and macros that simplify setting Haskell [H+92, P+97] programs in LaTeX [Lam94]. While the style is specifically geared towards Haskell, it should also be useful for other functional languages like ML or Nesl.

The famous *map* function can be set as follows:

$$
\begin{array}{lll}
map & :: (\alpha \to \beta) \to [\alpha] \to [\beta] & \text{-- type assertion} \\
map\ f\ [] & = [] & \text{-- [] is the empty list} \\
map\ f\ (x : xs) & = f\ x\ :\ map\ f\ xs & \text{-- : is the infix list constr.}
\end{array}
$$

Under this definition, *map* (+1) [1, 2, 3] evaluates to [2, 3, 4].

The previous example was set using the following input:

```
The famous \<map\> function can be set as follows:
%
\begin{haskell*}
  map          &::& (\alpha\to\beta)\to[\alpha]\to[\beta]
  &\hscom{type assertion}\\
  map f []     &=& []
  &\hscom{\<[]\> is the empty list}\\
  map f (x:xs) &=& f x : map f xs
  &\hscom{\<:\> is the infix list constr.}
\end{haskell*}
%
Under this definition, \<map (+1) [1, 2, 3]\>
evaluates to \<[2, 3, 4]\>.
```

There are a number of points worth noting. In a Haskell phrase, math mode commands, such as `\alpha` are available, but nevertheless, space characters can be used to denote juxtaposition of expressions (i.e., application in Haskell) and multi-letter identifiers are set properly. Compare $ffoo$ and *ffoo*, set by `$ffoo$` and `\<ffoo\>`, respectively.

The examples in the following sections do not provide the LaTeX sources used to typeset the examples in the formatted version. Please look at the source file to see the code.

2 Basic Usage

2.1 Haskell Mode

The environment `haskell` as well as the commands `\<` and `\>` enter Haskell mode. Keywords can be set in **bold face** using `\hskwd`. Within a `haskell` environment, columns are defined using the alignment character `&`. Usually, it is not necessary to wrap relations into alignment characters, but instead a single alignment to the right of the relational symbol suffices (just like AMSLaTeX's `align` environment). However, sometimes we want to align relational symbols of different size (i.e., the :: of a type assertion and the = of the corresponding equations). In this case, the star form `haskell*` should be used, which centers the material enclosed between the first and second alignment character.

2.2 Comments

The unary command `\hscom` is used to set single line comments.

```
data Maybe α = Nothing    -- No result
             | Just α     -- Just a value of type α
```

The second line shows that in a comment, you can again enter Haskell mode. This is particularly important when identifiers appear in comments or when you want to cite some program fragement, as the font choice in these cases should be identical to the fonts in the displayed program.

Sometimes, we like to omit some details of an algorithm and instead give an informal description. This is achieved with `\hsinf`.

```
main = do
         config ← ⟨read the configuration file⟩
         result ← ⟨process the input according to config⟩
         putStr result
```

2.3 Features from Math Mode

You can use all sorts of symbols, subscripts, and superscripts from LaTeX's math mode in Haskell mode. This includes TeX's relational symbols, for example, ≥ or ≠, and greek characters for type variables.

2.4 Aligned Material

Larger function bodies usually require the use of aligned subphrases to visually structure the definition. The command `\hsalign` generates an alignment with an arbitrary number of columns. Consider the function *transpose* from Haskell's standard library:

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

For several constructs that require alignments, there are predefined macros internally using `\hsalign`. For example for **let** expressions, we have `\hslet`:

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

Another macros is `\hsif` for if-then-else expression. In the following, example, we see that aligned subphrases can be nested:

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

Aligned material frequently gets big enough to require to break the surrounding alignment. This achieved with `\hsbody`:

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

A canned use of `\hsbody` is provided by `\hswhere` for **where** clauses:

```
partition       :: (a → Bool) → [a] → ([a], [a])
partition p xs = foldr select ([], []) xs
  where
    select x (ts, fs) | p x         = (x : ts, fs)
                      | otherwise = (ts, x : fs)
```

3 Advanced Features

If you need to set complex documents, you may want to know some of the more subtle points of the Haskell mode. Generally, whenever you attempt to use Haskell mode in sophistcated ways or try to define your own commands, keep in mind that spaces matter in Haskell mode, i.e., Haskell mode uses TEX's `\obeyspaces`. This changes TEX's behaviour in subtle ways.

3.1 Defining Commands

Commands that are used in Haskell mode should be defined with `\hscommand` instead of `\newcommand`. The arguments expected by `\hscommand` are the same as those for `\newcommand` (however, there is no `*`-version of `\hscommand`). `\hscommand` works slightly different, for example, with respect to processing spaces in the body of the command definition, to allow application by juxtaposition.

3.2 Entering "Real" Math Mode

Haskell mode is rather similar to math mode, to allow using special characters, subscripts, and so forth. However, sometimes, it is desirable to enter plain math

mode. Within a Haskell phrase, the commands `\(` and `\)` may be used to enter LaTeX's standard math mode. This is often helpful, when defining macros[1] and, for example, the macro `\hscom` is defined this way.

3.3 Use in Transformation Rules

For example in transformation rules, we want complex expressions, such as let-in constructs to be vertically centered. This is achieved by specifying an optional argument giving the alignment, for example, `\hslet[c]{···}`.

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

By default boxes are vertically aligned by the baseline of the first line (corresponding to the option `t`). Using `c` the alignment is in the center and with `b` it is at the baseline of the last line.

3.4 TeX's Formula Parser

TeX is capable of understanding some of the structure of formulae in math mode. The parsed structure influences the space inserted between the elements of a formula. Compare

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

In the first case, TeX groks that `+` is a binary operator applied to 3 and 4, but not in the second. Details are provided in *The TeXbook* [Knu86].

4 BSD3

The Haskell style is available under a BSD3 license.

References

[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.

[1] Sometimes it may be necessary to prefix such a use of `\(` by `\relax`, when the macro is used behind a `&`.