1 CurryDoc: A Documentation Generator for Curry Programs

CurryDoc is a tool to generate the documentation for a Curry program (i.e., the main module and all its imported modules) in HTML format. It can also be used with the Curry Package Manager CPM (see below) to generate the documentation of a Curry package. The generated HTML pages contain information about all data types, type classes and operations exported by a module as well as links between the different entities. Furthermore, some information about the definitional status of operations (like rigid, flexible, external, complete, or overlapping definitions) are provided and combined with documentation comments provided by the programmer.

1.1 Installation

The implementation of CurryDoc is contained in a package managed by the Curry Package Manager CPM. Thus, to install the newest version of CurryDoc, use the following commands:

```
> cypm update
> cypm install currydoc
```

This downloads the newest package, compiles it, and places the executable `curry-doc` into the directory `$HOME/.cpm/bin`. Hence it is recommended to add this directory to your path in order to execute CurryDoc as described below.

1.2 Documentation Comments

The documentation syntax follows Haddock (see `https://www.haskell.org/haddock/doc/html/index.html`).[1]

A *documentation comment* starts with "`-- | `" or "`-- ^`" (also in literate programs!).[2] The former style can be used to write document comments preceding a declaration and the latter for comments following a declaration. Nested comments are also supported with "`{- | `" or "`{- ^`" Other comments are also considered as documentation comments, if they are on a line directly below another documentation comment.

The documentation comments for the complete module occur before the first "module" or "import" line in the module. The module comments can also contain several special tags. These tags must be the first thing on its line (in the documentation comment) and continues until a line has at most the same degree of indentation or until the end of the comment. No tag must occur in the module comments, but any occurring tag as to be in the specified order. The following tags are recognized:

`Description:` *comment*

Specifies a short description of a module

`Category:` *comment*

Specifies the category of a module

[1]Note that older versions of CurryDoc (< 5.0.0) use a slightly different syntax which is still supported but should not be used.

[2]"`--- `" can be used instead of "`-- | `" for backwards compatibility

`Author:` *comment*
: Specifies the author of a module

`Version:` *comment*
: Specifies the version of a module

All text following the tags can be used for a longer module description.

The ordering of the final documentation can be controlled via the export list in the module header. The user can insert section headings via comments of the form "`-- *`" or "`{- *` ". The number of "`*`" controls if it is a subsection, subsubsection, etc.

The comment of a documented entity can be any string in *Markdown syntax*[3]. The currently supported set of elements is described in the Curry package `markdown`.[4] For instance, it can contain Markdown annotations for emphasizing elements (e.g., `_verb_`), strong elements (e.g., `**important**`), code elements (e.g., `` `3+4` ``), code blocks (lines prefixed by four blanks), unordered lists (lines prefixed by " `*` "), ordered lists (lines prefixed by blanks followed by a digit and a dot), quotations (lines prefixed by "`>` "), and web links of the form "`<http://...>`" or "`[link text](http://...)`". If the Markdown syntax should not be used, one could run CurryDoc with the option "`--nomarkdown`".

The comments cannot contain markups in HTML format anymore, due to the possibility of XSS-Attacks. In addition to Markdown, one can also mark *references to names* of operations or data types in Curry programs which are translated into links inside the generated HTML documentation. Such references have to be enclosed in single quotes. For instance, the text `'conc'` refers to the Curry operation `conc` inside the current module whereas the text `'Prelude.reverse'` refers to the operation `reverse` of the module `Prelude`. If one wants to write single quotes without this specific meaning, one can escape them with a backslash:

```
-- | This is a comment without a \'reference\'.
```

To simplify the writing of documentation comments, such escaping is only necessary for single words, i.e., if the text inside quotes has not the syntax of an identifier, the escaping can be omitted, as in

```
-- | This isn't a reference.
```

The following example text shows a Curry program with some documentation comments:

```
{- |
    Description: A simple example for CurryDoc
    Author     : Michael Hanus
    Version    : 0.1

    This is an
    example module.
-}
module Example (
  -- * Tree type
  Tree(..),
  -- * Operations on lists
```

[3] `http://en.wikipedia.org/wiki/Markdown`
[4] `https://cpm.curry-lang.org/pkgs/markdown.html`

```
    last,
    -- ** Operations repeated from the prelude
    conc
   ) where

  -- | The function `conc` concatenates two lists.
  --   It is identical to the function 'Prelude.++'.
  conc :: [a] -- ^ The first list
       → [a] -- ^ The second list
       → [a] -- ^ A list containing all elements of the parameters
  conc (x:xs) ys = x : conc xs ys
  conc []     ys = ys
  -- ^ This comment will also be included in the documentation

  -- | The function `last` computes the last element of a given list.
  --   It is based on the operation 'conc' to concatenate two lists.
  last :: Data a
       => [a] -- ^ The given input list
       → a   -- ^ The last element of the input list
  last xs | conc ys [x] =:= xs  = x   where x,ys free

  -- | This data type defines _polymorphic_ trees.
  data Tree a = Leaf a -- ^ A leaf of the tree
              | Node [Tree a] -- ^ An inner node of the tree
```

1.3 Generating Documentation for Curry Modules

To generate the documentation of the module `Example` shown above, execute the command

```
curry-doc Example
```

This command creates the directory `DOC_Example` (if it does not exist) and puts all HTML documentation files for the main program module `Example` and all its imported modules in this directory together with a main index file `index.html`. If one prefers another directory for the documentation files, one can also execute the command

```
curry-doc docdir Example
```

where `docdir` is the directory for the documentation files.

In order to generate the common documentation for large collections of Curry modules (e.g., the libraries contained in the Curry distribution), one can call `curry-doc` with the following options:

`curry-doc --noindexhtml docdir Mod`
: This command generates the documentation for module `Mod` in the directory `docdir` without the index pages (i.e., main index page and index pages for all functions and constructors defined in `Mod` and its imported modules).

`curry-doc --onlyindexhtml docdir Mod1 Mod2 ...Mod`n
: This command generates only the index pages (i.e., a main index page and index pages for

all functions and constructors defined in the modules `Mod1`, `Mod2`,...,`Mod`n and their imported modules) in the directory `docdir`.

1.4 Generating Documentation for Curry Packages

CurryDoc is also used by the Curry Package Manager CPM to generate the documentation of a Curry package. Inside a Curry package, one can execute the command

```
> cypm doc
```

to generate the documentation of all modules exported by this package. A detailed description of this command and its options can be found in the manual of CPM.

References

[1] M. Hanus. CurryDoc: A Documentation Tool for Declarative Programs. In *Proc. of the 11th International Workshop on Functional and (Constraint) Logic Programming (WFLP 2002)*, pages 225-228. Research Report UDMI/18/2002/RR, University of Udine, 2002.