1 `curry-doc`: A Documentation Generator for Curry Programs

CurryDoc is a tool in the Curry distribution that generates the documentation for a Curry program (i.e., the main module and all its imported modules) in HTML format. The generated HTML pages contain information about all data types and functions exported by a module as well as links between the different entities. Furthermore, some information about the definitional status of functions (like rigid, flexible, external, complete, or overlapping definitions) are provided and combined with documentation comments provided by the programmer.

1.1 Installation

The current implementation of CurryDoc is 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

A *documentation comment* starts at the beginning of a line with "`--- `" (also in literate programs!). All documentation comments immediately before a definition of a datatype or (top-level) function are kept together.[1] The documentation comments for the complete module occur before the first "module" or "import" line in the module. The comments can also contain several special tags. These tags must be the first thing on its line (in the documentation comment) and continues until the next tag is encountered or until the end of the comment. The following tags are recognized:

`@author` *comment*
: Specifies the author of a module (only reasonable in module comments).

`@version` *comment*
: Specifies the version of a module (only reasonable in module comments).

`@cons` *id comment*
: A comment for the constructor *id* of a datatype (only reasonable in datatype comments).

`@param` *id comment*
: A comment for function parameter *id* (only reasonable in function comments). Due to pattern matching, this need not be the name of a parameter given in the declaration of the function but all parameters for this functions must be commented in left-to-right order (if they are commented at all).

[1] The documentation tool recognizes this association from the first identifier in a program line. If one wants to add a documentation comment to the definition of a function which is an infix operator, the first line of the operator definition should be a type definition, otherwise the documentation comment is not recognized.

`@return` *comment*

A comment for the return value of a function (only reasonable in function comments).

The comment of a documented entity can be any string in *Markdown syntax*[2] (the currently supported set of elements is described in the Curry package `markdown`.[3] 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 parameter "`--nomarkdown`".

The comments can also contain markups in HTML format so that special characters like "`<`" must be quoted (e.g., "`&lt;`"). However, header tags like `<h1>` should not be used since the structuring is generated by CurryDoc. In addition to Markdown or HTML markups, 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:

```
--- This is an
--- example module.
--- @author Michael Hanus
--- @version 0.1

module Example where

--- The function `conc` concatenates two lists.
--- @param xs - the first list
--- @param ys - the second list
--- @return a list containing all elements of `xs` and `ys`
conc []     ys = ys
conc (x:xs) ys = x : conc xs ys
-- this comment will not 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.
--- @param xs - the given input list
```

[2] `http://en.wikipedia.org/wiki/Markdown`

[3] `https://cpm.curry-lang.org/pkgs/markdown.html`

```
--- @return last element of the input list
last xs | conc ys [x] =:= xs  = x   where x,ys free

--- This data type defines _polymorphic_ trees.
--- @cons Leaf - a leaf of the tree
--- @cons Node - an inner node of the tree
data Tree a = Leaf a | Node [Tree a]
```

1.3 Generating Documentation

To generate the documentation, 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`.

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.