NuSMV Doxygen documentation Documentation

Release 0.0

Alessandro Mariotti

October 06, 2014

Contents

Contents:

1 Documenting a NuSMV header file

Here is a significative example of how any NuSMV header file should look like, whether it declares a pseudo-class or it contains normal code:

```
/* ---------------------------------------------------------------------------

  This file is part of NuSMV version 2. Copyright (C) 2014 by
  FBK-irst.

  NuSMV version 2 is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2 of the License, or (at your option) any later version.

  NuSMV version 2 is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
```

```c
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA.

  For more information on NuSMV see <http://nusmv.fbk.eu>
  or email to <nusmv-users@fbk.eu>.
  Please report bugs to <nusmv-users@fbk.eu>.

  To contact the NuSMV development board, email to <nusmv@fbk.eu>.

-----------------------------------------------------------------------------*/

/*!
  \author Alessandro Mariotti
  \brief Provides functionalities for foo

  This is the full description. It can go multiline. It supports
  <i>HTML</i> and `Markdown` (From doxygen >= 1.8). I would
  suggest using the `Markdown` format, since more readable and
  easier to learn.

  Provides functionalities for foo, which are here described in a
  detailed way.

  If Foo is a NuSMV pseudo-class, describe the class
  functionalities here.

  \todo: Support for foo2 is missing

*/

#ifndef __FOO_H__
#define __FOO_H__

/*!
  \struct Foo
  \brief The Foo structure does foo

  Foo struct long description. Specifically for structs (this is the
  only case), the use of the \struct command is suggested for better
  documentation generation.
*/
typedef struct Foo_TAG* Foo_ptr;

/*!
  \brief Checks the given Foo instance

  Checks the given Foo instance. Check only checks whether the given
  value is NULL or not.
*/
#define FOO_CHECK_INSTANCE(x)  \
        (nusmv_assert(FOO(x) != FOO(NULL)))

/*!
  \brief A very nice enumeration
  Long description here
*/
typedef enum FooEnum_TAG {
        VAL1 = 1, /*!< Docs for VAL1 */
        VAL2 = 2, /*!< Docs for VAL2 */
        VAL3 = 3 /*!< Docs for VAL3 */
} FooEnumType;

/*!
  \brief A function that does something with the enviroment and returns
    a Foo instance

  A longer description about the function that does something with the
  enviroment and returns a Foo instance.

  Call this function is this way (this will be shown as code
  snippet, since separated from the text with an empty line and
  indented by 4 spaces):

      Foo_ptr foo = Foo_do_something(env, 2, strings);

  Parameter strings is freed by the Foo_do_something function, therefore we
  add the <b>takes_mem</b> command

  The return value memory is handled internally, so it must not be
  freed by the caller. In this case, we add the <b>keeps_mem</b>
  command.

  \todo Missing description about Foo Fighters.

  \param env The environment
  \param param An integer parameter
  \param strings The input list of strings \takes_mem
  \return A Foo instance. \keeps_mem

  \todo Improve this documentation
  \sa Foo_do_something_2 (this is for See-Also)
  \se The given NuSMVEnv_ptr instance is changed (this is for Side-Effect)
*/
Foo* Foo_do_something(NuSMVEnv_ptr env, int param, char** strings);

#endif /* __FOO_H__ */
```

Here is a significative example of how to rightly describe a method, it's mandatory to use the directive *methodof* in order to correctly associate "methods" to the corresponding classes in doxygen documentation.

```
/*!
   \methodof ClassName
   \brief Short description

   Longer description...
*/
 <type> ClassName_method_name(ClassName* self, ...);
```

2 Documenting a NuSMV source file

The documentation rules for header files apply also for the source files. Altough source file members documentation is optional, the copyright header and the first doxygen entry, where author and description of the file are described, are mandatory.

Currently, doxygen documentation within a source file is not exported in the generated documentation. For exporting such documentation, please put it in a *.doxy file, which will be automatically read by doxygen.

Here is a significative example of how a NuSMV source file should look like:

```
/* ---------------------------------------------------------------------------

   This file is part of NuSMV version 2.  Copyright (C) 2014 by
   FBK-irst.

   NuSMV version 2 is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2 of the License, or (at your option) any later version.

   NuSMV version 2 is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
   Lesser General Public License for more details.

   You should have received a copy of the GNU Lesser General Public
   License along with this library; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA.

   For more information on NuSMV see <http://nusmv.fbk.eu>
   or email to <nusmv-users@fbk.eu>.
   Please report bugs to <nusmv-users@fbk.eu>.

   To contact the NuSMV development board, email to <nusmv@fbk.eu>.

-----------------------------------------------------------------------------*/

/*!
  \author Alessandro Mariotti
  \brief Implementation of Foo

  A long description about the Foo implementation
*/

#include "nusmv/core/Foo.h"

/*---------------------------------------------------------------------------*/
/* Type declarations                                                         */
/*---------------------------------------------------------------------------*/

typedef struct Foo_TAG
{
  INHERITS_FROM(EnvObject);

  NodeMgr_ptr nodes; /*!< Used for something about Foo */

} Foo;
```

```
/*---------------------------------------------------------------------------*/
/* Static function prototypes                                                */
/*---------------------------------------------------------------------------*/

static int
foo_do_something_new(const Foo_ptr foo);

/*---------------------------------------------------------------------------*/
/* Definition of exported functions                                          */
/*---------------------------------------------------------------------------*/

Foo* Foo_do_something(NuSMVEnv_ptr env, int param, char** strings)
{
   /* Do a lot of stuff */
}

/*--------------------------------------------------------------------------*/
/* Definition of static functions                                           */
/*--------------------------------------------------------------------------*/

/*!
  \brief Does something new with Foo

  \param foo the Foo instance
  \return an integer that has something to do with the given foo instance
*/
static int foo_do_something_new(const Foo_ptr foo)
{
        /* Do a lot of stuff */
}
```

3 Documenting a NuSMV package

With the introdution of the doxygen documentation generation, it is now possible to write proper top-level documentation for *NuSMV* and *NuSMV* addons packages.

Packages documentation can be placed in any directory that is under the doxygen documentation path (see the doxygen configuration file for details), but **must** have the *.doxy* extension.

One single *.doxy* documentation file can contain documentation of one or more packages and sub-packages, but it is mandatory to specify the belonging package directory with the \dir <path> command.

An example for the node package and one of it's sub-packages would look like

File `NuSMV2/NuSMV/doc/doxygen/packages/node/node.doxy`:

```
/*!
  \dir nusmv/core/node
  \brief This is the node package

  This package contains a lot of nodes..
*/

/*!
  \dir nusmv/core/node/printers
  This sub-package contains a lot of node printers..
*/
```

However, these can be split-up onto two files:

File `NuSMV2/NuSMV/doc/doxygen/packages/node/node.doxy`:

```
/*!
  \dir nusmv/core/node
  \brief This is the node package

  This package contains a lot of nodes..
*/
```

File `NuSMV2/NuSMV/doc/doxygen/packages/node/printers/printers.doxy`:

```
/*!
  \dir nusmv/core/node/printers

  This sub-package contains a lot of node printers..
*/
```

3.1 TO BE DISCUSSED

Package documentation files can be placed everywhere in the source tree. However, I would suggest using one of the two following rules:

1. Place them within an ad-hoc directory, as listed below:

Code	Docs directory
NuSMV	NuSMV2/NuSMV/doc/doxygen/packages
addons	NuSMV2/addons/doc/doxygen/packages

2. Place them directly in the package directory:

Pkg	Docs directory
node	NuSMV2/NuSMV/nusmv/core/node
simp	NuSMV2/addons/src/addons/simp

4 Documenting a NuSMV shell command

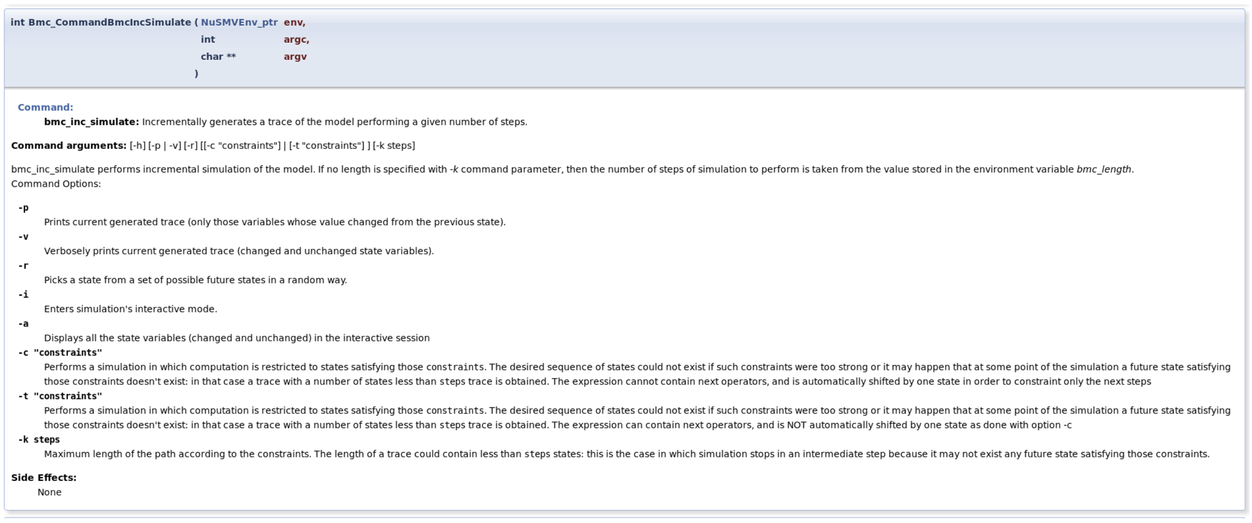
Here is a significative example of how a NuSMV shell commands documentation should look like.

Part of file `NuSMV2/NuSMV/shell/bmc/bmcCmd.h`:

```
/*!
  \command{bmc_inc_simulate} Incrementally generates a trace of the model
  performing a given number of steps.

  \command_args{\[-h\] \[-p | -v\] \[-r\]
  [\[-c "constraints"\] | \[-t "constraints"\] ] \[-k steps\]}

  bmc_inc_simulate performs incremental simulation
  of the model. If no length is specified with <i>-k</i> command
  parameter, then the number of steps of simulation to perform is
  taken from the value stored in the environment variable
  <i>bmc_length</i>.<BR>

  \command_opts{

    \opt{p, Prints current generated trace (only those variables
            whose value changed from the previous state)}

    \opt{v, Verbosely prints current generated trace (changed and
             unchanged state variables)}

    \opt{r, Picks a state from a set of possible future states in
            a random way.}

    \opt{i, Enters simulation's interactive mode.}

    \opt{a, Displays all the state variables (changed and
            unchanged) in the interactive session}

    \opt{c "constraints", Performs a simulation in which
         computation is restricted to states satisfying those
         <tt>constraints</tt>. The desired sequence of states
         could not exist if such constraints were too strong or it
         may happen that at some point of the simulation a future
         state satisfying those constraints doesn't exist: in that
         case a trace with a number of states less than
         <tt>steps</tt> trace is obtained. The expression cannot
         contain next operators\, and is automatically shifted by
         one state in order to constraint only the next steps}

    \opt{t "constraints", Performs a simulation in which
         computation is restricted to states satisfying those
         <tt>constraints</tt>. The desired sequence of states
         could not exist if such constraints were too strong or it
         may happen that at some point of the simulation a future
         state satisfying those constraints doesn't exist: in that
         case a trace with a number of states less than
         <tt>steps</tt> trace is obtained.  The expression can
         contain next operators\, and is NOT automatically shifted
         by one state as done with option -c}

    \opt{k steps, Maximum length of the path according to the
        constraints.  The length of a trace could contain less
        than <tt>steps</tt> states: this is the case in which
        simulation stops in an intermediate step because it may
        not exist any future state satisfying those constraints.
        </dl>}
  }

  \se None
*/
int Bmc_CommandBmcSimulate (NuSMVEnv_ptr env, int argc, char** argv);
```

Using the proper documentation tags for documenting commands results in a better organized an better looking documentation: Each command properly documented ends up in the **Commands** related page, and is therefore more readable and easier to find

`Generated documentation of the example above`



int Bmc_CommandBmcIncSimulate (NuSMVEnv_ptr env,
int argc,
char ** argv
)

Command:
bmc_inc_simulate: Incrementally generates a trace of the model performing a given number of steps.

Command arguments: [-h] [-p | -v] [-r] [[-c "constraints"] | [-t "constraints"]] [-k steps]

bmc_inc_simulate performs incremental simulation of the model. If no length is specified with *-k* command parameter, then the number of steps of simulation to perform is taken from the value stored in the environment variable *bmc_length*.
Command Options:

`-p`
Prints current generated trace (only those variables whose value changed from the previous state).

`-v`
Verbosely prints current generated trace (changed and unchanged state variables).

`-r`
Picks a state from a set of possible future states in a random way.

`-i`
Enters simulation's interactive mode.

`-a`
Displays all the state variables (changed and unchanged) in the interactive session

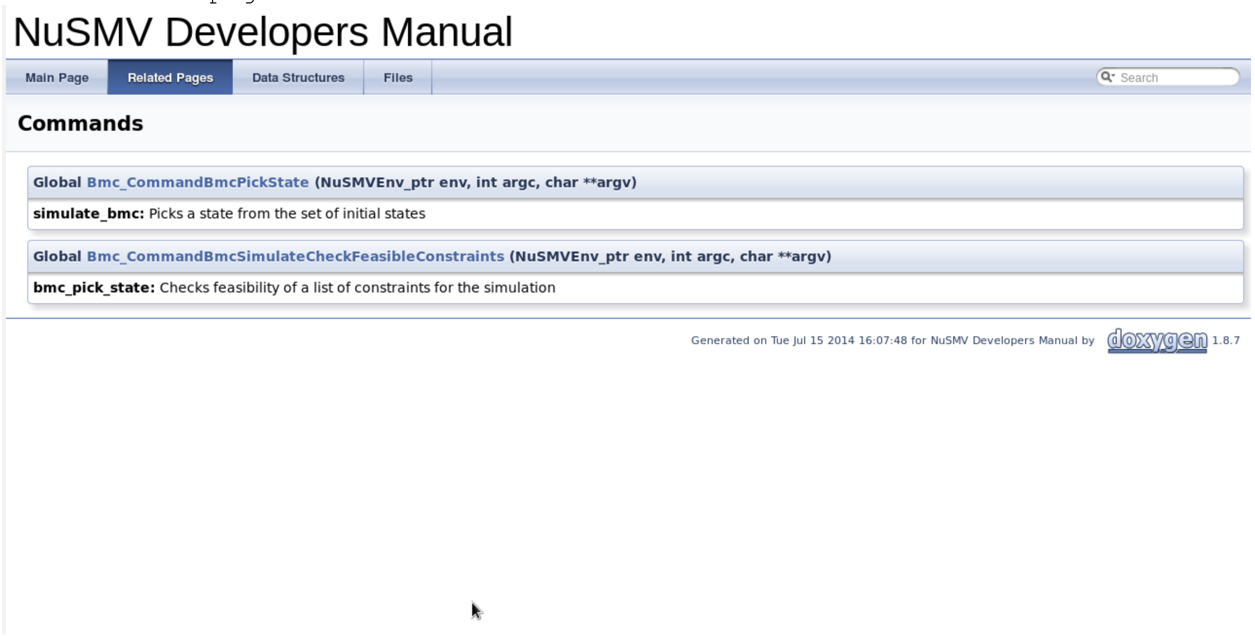
`-c "constraints"`
Performs a simulation in which computation is restricted to states satisfying those `constraints`. The desired sequence of states could not exist if such constraints were too strong or it may happen that at some point of the simulation a future state satisfying those constraints doesn't exist: in that case a trace with a number of states less than `steps` trace is obtained. The expression cannot contain next operators, and is automatically shifted by one state in order to constraint only the next steps

`-t "constraints"`
Performs a simulation in which computation is restricted to states satisfying those `constraints`. The desired sequence of states could not exist if such constraints were too strong or it may happen that at some point of the simulation a future state satisfying those constraints doesn't exist: in that case a trace with a number of states less than `steps` trace is obtained. The expression can contain next operators, and is NOT automatically shifted by one state as done with option -c

`-k steps`
Maximum length of the path according to the constraints. The length of a trace could contain less than `steps` states: this is the case in which simulation stops in an intermediate step because it may not exist any future state satisfying those constraints.

Side Effects:
None

`The Commands page`



NuSMV Developers Manual

Commands

Global Bmc_CommandBmcPickState (NuSMVEnv_ptr env, int argc, char **argv)

simulate_bmc: Picks a state from the set of initial states

Global Bmc_CommandBmcSimulateCheckFeasibleConstraints (NuSMVEnv_ptr env, int argc, char **argv)

bmc_pick_state: Checks feasibility of a list of constraints for the simulation

Generated on Tue Jul 15 2014 16:07:48 for NuSMV Developers Manual by doxygen 1.8.7

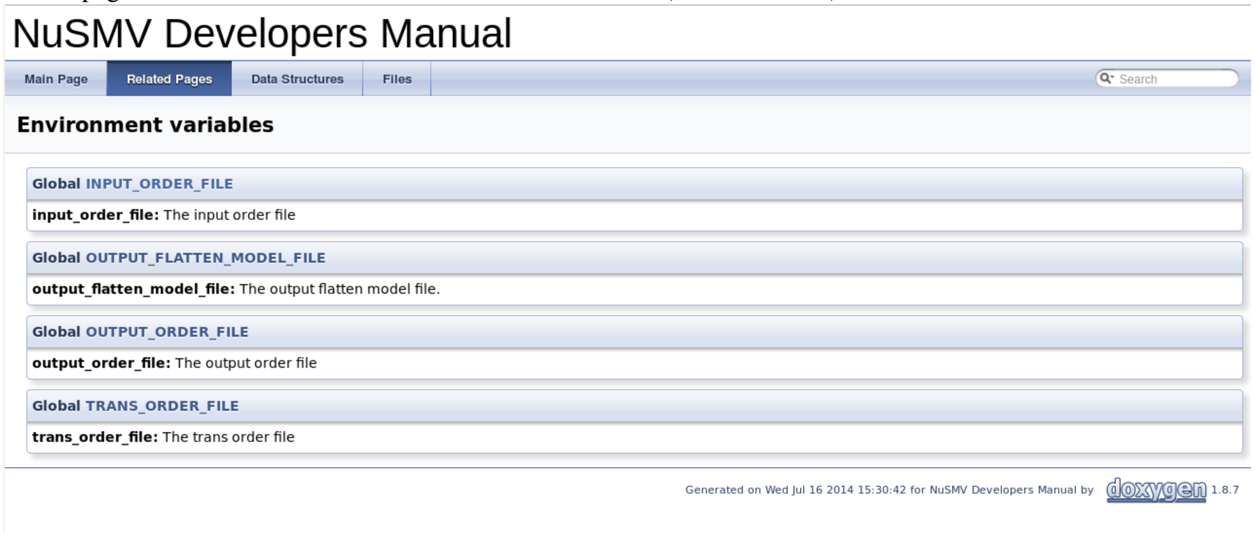
5 Documenting a NuSMV environment variable

Here is a significative example of how a NuSMV environment variables documentation should look like. Place the environment documentation above the definition of the name of the variable:

Part of file `NuSMV2/NuSMV/core/opt/opt.h`:

```
/*!
  \env_var{input_order_file} The input order file

  Longer description

*/
#define INPUT_ORDER_FILE  "input_order_file"
```

Using the proper documentation tags for documenting environment variables results in a better organized an better looking documentation: Each environment variable properly documented ends up in the **Environment variables** related page, and is therefore more readable and easier to find (See screenshot)



NuSMV Developers Manual

Environment variables

Global INPUT_ORDER_FILE

input_order_file: The input order file

Global OUTPUT_FLATTEN_MODEL_FILE

output_flatten_model_file: The output flatten model file.

Global OUTPUT_ORDER_FILE

output_order_file: The output order file

Global TRANS_ORDER_FILE

trans_order_file: The trans order file

Generated on Wed Jul 16 2014 15:30:42 for NuSMV Developers Manual by doxygen 1.8.7