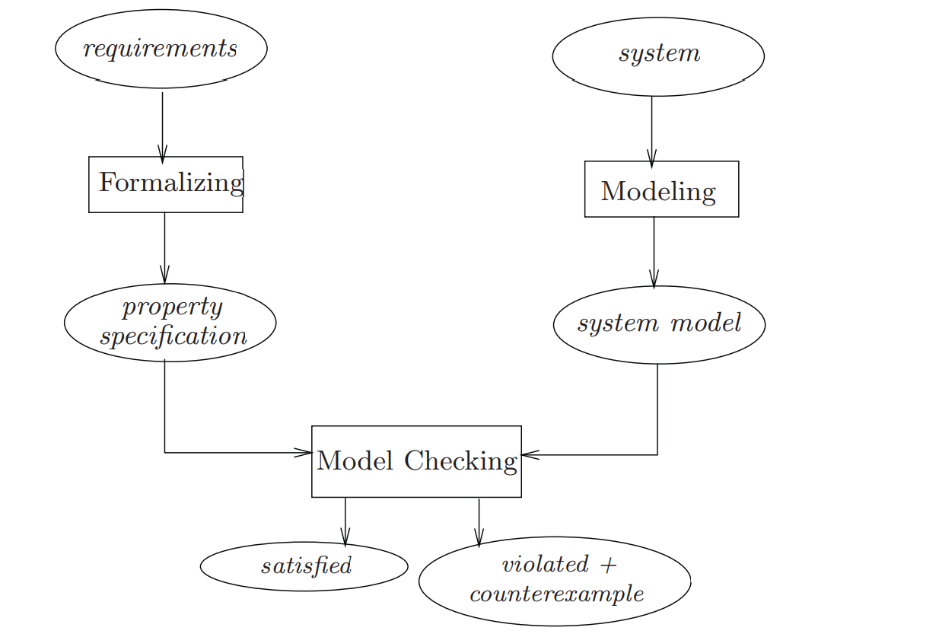
**NuSMV使用教程**

**摘要**：本教程简要介绍**模型检测**（Model Checking）技术，并提供对模型检测工具NuSMV的相关教程，包括安装方式、其接受的输入语言格式及使用实例。

# 简要介绍

**模型检测**（Model Checking）是一种验证技术，它以蛮力搜索的方式遍历系统所有可能的状态。通过这种方式，可以证明给定的系统模型确实满足某个特性或者违反某个特性。目前模型检测最大的挑战是状态空间爆炸，最新的模型检测工具可以通过显式的状态空间枚举处理大约108到109个状态的状态空间，如果使用巧妙构造的算法和特定的数据结构，可以针对特定问题处理更大的状态空间（1020个甚至更多状态）。模型检测最大的优势是能够毫无遗漏的发现系统所有的错误，比如模拟、仿真和测试未发现的细微错误。

图 1 模型检测示意图



2001年，基于SMV(Symbolic Model Verifier)，Carnegie Mellon University(CMU)和Istituto per la Ricerca Scientifica e Techolgica(IRST)联合开发出模型验证器NuSMV，它主要是针对SMV 2.4.4版本的重新实现和扩展，重新定义了软件架构并加入了一些新特性。NuSMV目前已发展到2.6.0版本。具体来说，NuSMV从三个方面扩展了SMV：

* 功能上，除了可以验证用CTL描述的规范外，还可以验证用LTL描述的规范；不仅实现了经典的基于BDD的符号模型检测技术外，还整合了基于SAT的有界模型验证技术(BMC)；提供了一个类似于Unix的shell的接口，方便用户使用。
  + 相对于SMV，NuSMV定义了一个良好的软件系统架构，实现也更加模块化和开放，容易删除、替换或添加模块。例如，可以使用商用的zchaff包提供更加高效的有界模型验证技术。
  + NuSMV源码的注释、文档化更加完整，比SMV更加容易读和便于修改。这归因于NuSMV的一个目标是提供一个模型检测的通用平台，所以在编码上考虑到未来的扩展和修改。

**可用资源**：

1. NuSMV工具网址：<https://nusmv.fbk.eu/index.html>
2. NuSMV用户手册：<https://nusmv.fbk.eu/NuSMV/userman/v26/nusmv.pdf>
3. NuSMV官方教程：<https://nusmv.fbk.eu/NuSMV/tutorial/v26/tutorial.pdf>

# NuSMV的安装

**推荐第2种，方便快捷！！**

1、从源码安装（仅展示GNU/Linux系统），系统要求：GNU/Linux，比如Ubuntu。（以下例程基于Ubuntu 20.04 LTS amd64）

# 安装依赖

sudo apt install gcc g++ flex bison cmake tar gzip libxml2 libreadline6-dev doxygen texlive texmaker

# 在Ubuntu上进行编译

wget http://nusmv.fbk.eu/distrib/NuSMV-2.6.0.tar.gz

tar zxvf NuSMV-2.6.0.tar.gz

cd NuSMV-2.6.0/NuSMV

mkdir build

cd build

cmake ..

gedit code/nusmv/shell/cmd/cmdHelp.c

# 修改58行为: ”int command\_number;”

gedit doc/prog-man/cmake\_install.cmake

# 删除49行内容: ”/html”

gedit ../../cudd-2.4.1.1/util/pipefork.c

# 修改43行为: ”#if (defined \_\_linux\_\_) || (defined \_\_hpux) || (defined \_\_osf\_\_) || (defined \_IBMR2) || (defined \_\_SVR4) || (defined \_\_CYGWIN32\_\_) || (defined \_\_MINGW32\_\_)”

gedit ../../MiniSat/MiniSat\_v37dc6c6\_nusmv.patch

# 修改679行为: ”+extern "C" void MiniSat\_Delete(MiniSat\_ptr ms)”

gedit ../../NuSMV/cmake/combine\_grammar.py

# 修改41行为: ”for key in sorted(d, reverse=True):”

make

sudo make install

使用方式：

NuSMV /usr/local/share/nusmv/examples/smv-dist/counter.smv

注：上述安装过程将NuSMV安装至目录/usr/local中，其中示例文件目录为/usr/local/share/nusmv/examples。

2、下载二进制文件、解压运行即可。

1）GNU/Linux系统，比如Ubuntu：

# GNU/Linux libc6 (686) 32-bit

wget https://nusmv.fbk.eu/distrib/NuSMV-2.6.0-linux32.tar.gz

tar zxvf NuSMV-2.6.0-linux32.tar.gz

sudo cp -R NuSMV-2.6.0-Linux/\* /usr/local/

# GNU/Linux libc6 (x86) 64-bit

wget https://nusmv.fbk.eu/distrib/NuSMV-2.6.0-linux64.tar.gz

tar zxvf NuSMV-2.6.0-linux64.tar.gz

sudo cp -R NuSMV-2.6.0-Linux/\* /usr/local/

使用方式：

NuSMV /usr/local/share/nusmv/examples/smv-dist/counter.smv

注：上述安装过程将NuSMV安装至目录/usr/local中，其中示例文件目录为/usr/local/share/nusmv/examples。

2）Windows系统：

# Windows archive 32-bit (586)

# 下载地址 https://nusmv.fbk.eu/distrib/NuSMV-2.6.0-win32.tar.gz

# 解压缩包，比如至C:\Program Files (x86)\NuSMV-2.6.0-win32

# 然后配置PATH，编辑path，添加C:\Program Files (x86)\NuSMV-2.6.0-win32\bin

# 使用方式，打开cmd键入： NuSMV "C:\Program Files (x86)\NuSMV-2.6.0-win32\share\nusmv\examples\smv-dist\counter.smv"

注：上述安装过程将NuSMV安装至目录C:\Program Files (x86)\NuSMV-2.6.0-win32中，其中示例文件目录为C:\Program Files (x86)\NuSMV-2.6.0-win32\share\nusmv\examples。

注：上述安装过程将NuSMV安装至目录C:\Program Files (x86)\NuSMV-2.6.0-win64中，其中示例文件目录为C:\Program Files (x86)\NuSMV-2.6.0-win64\share\nusmv\examples。

# Windows archive 64-bit (x86)

# 下载地址 https://nusmv.fbk.eu/distrib/NuSMV-2.6.0-win64.tar.gz

# 解压缩包，比如至C:\Program Files (x86)\NuSMV-2.6.0-win64

# 然后配置PATH，编辑path，添加C:\Program Files (x86)\NuSMV-2.6.0-win64\bin

# 使用方式，打开cmd键入： NuSMV "C:\Program Files (x86)\NuSMV-2.6.0-win64\share\nusmv\examples\smv-dist\counter.smv"

3）MacOS系统：

# MacOSX Darwin (x86) 64-bit

# 下载地址https://nusmv.fbk.eu/distrib/NuSMV-2.6.0-macosx64.tar.gz

# 其余步骤请参考1）

# 实例介绍

NuSMV用smv输入语言（规定的一种文本格式）来描述Kripke结构和待验证的规范。在NuSMV中Kripke结构常称为Finite State Machine（FSM）。其输入语言中，表达式和语句类似于C语言。NuSMV有两个重要的表达式：init表达式和next表达式。

* init表达式用于描述初始状态；
* next表达式用于描述转移关系。

其程序（比如counter.smv）常被称为smv程序，由模块（MODULE）构成。模块由模块名和模块定义组成，模块定义又由形参（parameter）和主体（body）部分组成。模块主体部分分为三类：Variables部分、Constraint部分和Specification部分。

* Variables部分用于描述Kripke模型的状态集；
* Constraint部分用于描述Kripke模型的转移关系和对模型的一些限制；
* Specification部分用来描述待验证规范。

另smv程序至少要有一个称为main的模块，且main模块不能有形参。可以使用多个模块描述FSM，然后组合成一个整体的FSM。

一个典型的smv程序的结构如下：

MODULE main //至少要有一个main模块，为系统建模

VAR

… //状态变量声明

ASSIGN

… //初始状态和转移关系的声明

SPEC(或LTLSPEC、CTLSPEC) //规范定义，可选

… //使用CTL或LTL描述特验证的系统规范

MODULE submodule //各个子模块的定义，可选

… //同main模快

1）Variables部分有两大类：

* State Varibles（状态的赋值表示具体的某个状态）；
* Input Varibles（通过标记关系来表示状态）。

分别以关键字VAR或IVAR表示。Variables的类型仅为boolean、integer、enum、word、array以及set类型。

2）Constraints的种类有assign、trans、init、invar、fairness等，分别以关键字ASSIGN、TRANS、INIT、INVAR、FAIRNESS表示。

为了更加方便地描述FSM，NuSMV还引入了DEFINE。DEFINE定义的符号的可看成是一个宏。

3）Specification部分可以使用CTL公式，也可以使用LTL公式。

以下为counter.smv源程序的示例：

MODULE main

VAR

bit0: counter\_cell(TRUE);

bit1: counter\_cell(bit0.carry\_out);

bit2: counter\_cell(bit1.carry\_out);

SPEC

AG AF bit2.carry\_out

MODULE counter\_cell(carry\_in)

VAR

value : boolean;

ASSIGN

init(value) := FALSE;

next(value) := value xor carry\_in;

DEFINE

carry\_out := value & carry\_in;

该程序为3位二进制计数器电路的模型。以下简要分析：

由main模块可知，调用了3次counter\_cell模块，所以整体模块拥有3个boolean变量bit0.value, bit1.value, bit2.value。ASSIGN语句中的init指定初始状态为（bit2.value, bit1.value, bit0.value）=（0，0，0）。ASSIGN语句中的next指定下一状态：

* bit0.value = bit0.valueTRUE = bit0.value；
* bit1.value = bit1.valuebit0.carry\_out = bit1.value(bit0.value & TRUE) = bit1.valuebit0.value；
* bit2.value = bit2.valuebit1.carry\_out = bit2.value(bit1.value & bit0.carry\_out) = bit2.value(bit1.value & (bit0.value & TRUE)) = bit2.value(bit1.value & bit0.value)。

由上述转移关系可知，（bit2.value, bit1.value, bit0.value）：（0，0，0）（0，0，1）（0，1，0）……（1，1，1）（0，0，0）。因此该程序为3位二进制计数器电路的模型。

要验证的规范为CTL规范：AG AF bit2.carry\_out，即AG ( AF bit2.value & bit1.value & bit0.value )，该规范含义：对CTL计算树中所有路径，路径中所有节点，该节点的所有后续路径，路径中存在一个节点使得该节点满足bit2.value = bit1.value = bit0.value = TRUE。从计数器的模型中，容易想象该规范是满足的（计数器从000一直增到111，再变为000，以此循环下去）。

事实上，含有多个模块程序可以转化为仅含一个main模块的程序。以下为示例：

MODULE main

VAR

a: subModule1;

b: subModule2;

c: 10..20;

MODULE subModule1

VAR

value : boolean;

b: {start, end};

MODULE subModule2

VAR

value : 0..7;

ASSIGN

next(value) := 2;

MODULE main

VAR

a.value : boolean;

a.b: {start, end};

b.value : 0..7;

c: 10..20;

ASSIGN

next(b.value) := 2;

注：上述程序中b.value表示0-7之间的整数，包含0、7。更多示例查看share/nusmv/examples目录。