

NuSMV-Project-01

本仓库是《软件工程理论基础》项目一代码和报告。

`./proj1.pdf` 是项目要求，本仓库完成了项目要求中的三个题目。

【广告】除了 NuSMV 之外，软件工程有另一个形式化验证工具 [TLA+](#)。该工具由图灵奖获得者 [Leslie Lamport](#) 开发（ $LaTeX$ 中的 La 就是指 Lamport）。我的 b 站主页搬运了 Lamport 本人录制的教程 [The TLA+ Video Course](#)，仅供学习。

1. NuSMV 的安装和配置

在 [NuSMV - Downloads](#) 下载对应版本，推荐直接下载右侧红框中对应的二进制文件。

NuSMV v2 is open source software and it is distributed under [LGPL v2.1 license](#). For more information about the availability and the use of NuSMV v2 please send an e-mail to nusmv@fbk.eu. Before downloading NuSMV v2 please carefully read the [LGPL license](#). NuSMV v1.1 can be used, free of charge, only by academic institutions and only for their internal research and educational purposes.

Source Code		
Version	License	Download
2.7.0	LGPL v2.1	NuSMV-2.7.0.tarxz NuSMV-2.7.0.tarxz.sha256sum
2.6.0	LGPL v2.1	NuSMV-2.6.0.targz
2.5.4	LGPL v2.1	NuSMV-2.5.4.targz
2.5.3	LGPL v2.1	NuSMV-2.5.3.targz
2.5.2	LGPL v2.1	NuSMV-2.5.2.targz
2.5.1	LGPL v2.1	NuSMV-2.5.1.targz
2.5.0	LGPL v2.1	NuSMV-2.5.0.targz
2.4.3	LGPL v2.1	NuSMV-2.4.3.targz
2.4.2	LGPL v2.1	NuSMV-2.4.2.targz
2.4.1	LGPL v2.1	NuSMV-2.4.1.targz

Binaries		
Version	License	Downloads
2.7.0	LGPL v2.1	NuSMV-2.7.0-linux64.tarxz ⓘ NuSMV-2.7.0-linux64.tarxz.sha256sum ⓘ NuSMV-2.7.0-macos-universal.tarxz ⓘ NuSMV-2.7.0-macos-universal.tarxz.sha256sum ⓘ NuSMV-2.7.0-win64.zip ⓘ NuSMV-2.7.0-win64.zip.sha256sum ⓘ
2.6.0	LGPL v2.1	NuSMV-2.6.0-linux32.targz ⓘ NuSMV-2.6.0-linux64.targz ⓘ NuSMV-2.6.0-macosx64.targz ⓘ NuSMV-2.6.0-win32.targz ⓘ NuSMV-2.6.0-win64.targz ⓘ

以 Windows 系统为例，下载到本地后解压到任意目录。

File Explorer path: 此电脑 > Data (D:) > Seafile > Ventoy-Installation > Windows-Installation > NuSMV-2.7.0-win64

名称	修改日期	类型	大小
bin	2025-04-08 21:43	文件夹	
include	2025-04-08 21:43	文件夹	
lib	2025-04-08 21:43	文件夹	

将 `NuSMV-2.7.0-win64/bin` 的路径添加到系统环境变量后，即可在命令行使用 NuSMV 命令。

```

Windows PowerShell
版权所有 (C) Microsoft Corporation. 保留所有权利。

安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows

PS D:\Seafire\Ventoy-Installation\Windows-Installation\NuSMV-2.7.0-win64> NuSMV
*** This is NuSMV 2.7.0 (compiled on Thu Oct 24 17:56:00 2024)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2024, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

Input file is (null). You must set the input file before.

Aborting batch mode

NuSMV terminated by a signal

Aborting batch mode
PS D:\Seafire\Ventoy-Installation\Windows-Installation\NuSMV-2.7.0-win64>

```

使用下列命令可以运行所需的 NuSMV 源文件。

```
1 NuSMV -df filename.smv
```

在 VSCode 中安装 coderunner 插件, 并在配置文件中添加 `".smv": "cd $dir && NuSMV -df $fileName"` 即可使用快捷键 `Ctrl+Alt+N` 运行 `.smv` 文件。

```

README.md M  bubble_sort.smv U  bubble_sort_c.smv U  Settings  settings.json 9  formula_a.smv U
C: > Users > 37756 > AppData > Roaming > Code > User > {} settings.json > {} code-runner.executorMapByFileExtension > .smv
74  "code-runner.executorMapByFileExtension": {
95      ".csproj": "dotnet run --project",
96      ".fsproj": "dotnet run --project",
97      ".lisp": "sbcl --script",
98      ".kit": "kitc --run",
99      ".v": "v run",
100     ".vsh": "v run",
101     ".sass": "sass --style expanded",
102     ".cu": "cd $dir && nvcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",
103     ".smv": "cd $dir && NuSMV -df $fileName"
104 },
105 "C_Cpp.default.cppStandard": "c++17",
106 "C_Cpp.default.cStandard": "c11",
107 "code-runner.runInTerminal": true,
108 "tabnine.experimentalAutoImports": true,
109 "latex-workshop.view.pdf.viewer": "tab",
110 "python.languageServer": "Default",
111 "workbench.editorAssociations": {
112     "text/plain": "code-runner.executorMapByFileExtension"

```

2. 公式等价性判断

结论: a, c 和 d 中两个公式不等价, b 中两个公式等价。

`./formula/fomula_a.smv` 给出了 a 的一个反例:

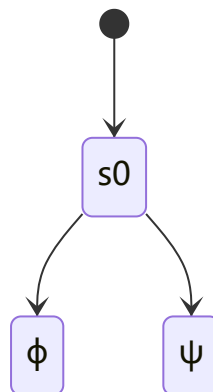


在上述反例中:

```

1 CTLSPEC EF(phi); -- 结果为 TRUE
2 CTLSPEC EG(phi); -- 结果为 FALSE
  
```

`./formula/fomula_c.smv` 给出了 c 的一个反例:

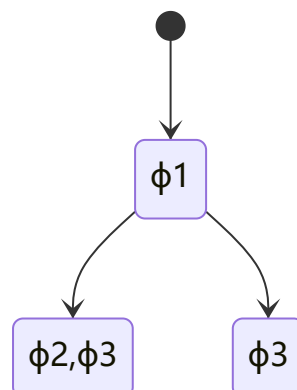


在上述反例中:

```

1 CTLSPEC AF(phi); -- 结果为 TRUE
2 CTLSPEC AF(psi); -- 结果为 FALSE
  
```

`./formula/fomula_d.smv` 给出了 d 的一个反例:



在上述反例中：

```
1 CTLSPEC A[phi1 U A[phi2 U phi3]]; -- 结果为 TRUE
2 CTLSPEC A[A[phi1 U phi2] U phi3]; -- 结果为 FALSE
```

3. 冒泡排序建模

`./bubble_sort/bubble_sort.smv` 使用 SMV 代码建模冒泡排序算法，并验证是否完成排序。

基本思路是，根据 `j` 的取值来实现交换逻辑，例如下面代码片段。值得注意的是，NuSMV 是同步赋值，因此 `a0` 和 `a1` 的更新基于当前状态的值，下面写法中的变量交换是原子的。

```

1  -- 交换逻辑
2  next(a0) := case
3    -- 比较 a[j] 和 a[j+1]，若需要交换则更新
4    j = 0 & a0 > a1 : a1;  -- 交换 a0 和 a1
5    TRUE : a0;
6  esac;
7
8  next(a1) := case
9    j = 0 & a0 > a1 : a0;  -- 交换 a0 和 a1
10   j = 1 & a1 > a2 : a2;  -- 交换 a1 和 a2
11   TRUE : a1;
12  esac;
13
14  next(a2) := case
15    j = 1 & a1 > a2 : a1;  -- 交换 a1 和 a2
16    TRUE : a2;
17  esac;

```

使用以下规约验证排序结果：

```

1  CTLSPEC AG( (i = 2 & j = 0 & !swapped) -> (a0 <= a1 & a1 <= a2) ) -- 结果为 TRUE

```

`./bubble_sort/bubble_sort_c.smv` 是改进的算法，使用枚举避免整数域运算，并验证了算法的正确性。由于题目限制了变量的取值范围为 `0..7`，因此可以用三个比特（布尔值）来表达一个整数，进而用布尔（逻辑）运算来代替整数运算。

```

1  DEFINE
2    -- 定义比较逻辑（使用位比较代替数值比较）
3    greater_a0_a1 := (a0_b2 & !a1_b2) |
4                     ((a0_b2 = a1_b2) & (a0_b1 & !a1_b1)) |
5                     ((a0_b2 = a1_b2) & (a0_b1 = a1_b1) & (a0_b0 & !a1_b0));
6
7    greater_a1_a2 := (a1_b2 & !a2_b2) |
8                     ((a1_b2 = a2_b2) & (a1_b1 & !a2_b1)) |
9                     ((a1_b2 = a2_b2) & (a1_b1 = a2_b1) & (a1_b0 & !a2_b0));

```

使用以下规约验证排序结果：

```

1  CTLSPEC AG( (i = 2 & j = 0 & !swapped) -> (!greater_a0_a1 & !greater_a1_a2) ) -- 结果为 TRUE

```

4. Raft 算法建模

`./raft/raft.smv` 使用 NuSMV 对 Raft 算法进行建模，并验证了其正确性。

对每个定时器制定相同的计时规则：

```

1  -- 定时器更新逻辑，每次增加 1，模拟时间流逝
2  next(timer1) := case
3      node1 = Leader: 0; -- 如果当前是 Leader，定时器归零
4      node1 = Follower: min(timer1 + 1, 10); -- Follower 每次增加 1，但不超过 10
5      node1 = Candidate: min(timer1 + 1, 10); -- Candidate 每次增加 1，但不超过 10
6  esac;
7
8  next(timer2) := case
9      node2 = Leader: 0; -- 如果当前是 Leader，定时器归零
10     node2 = Follower: min(timer2 + 1, 10); -- Follower 每次增加 1，但不超过 10
11     node2 = Candidate: min(timer2 + 1, 10); -- Candidate 每次增加 1，但不超过 10
12  esac;
13
14  next(timer3) := case
15      node3 = Leader: 0; -- 如果当前是 Leader，定时器归零
16      node3 = Follower: min(timer3 + 1, 10); -- Follower 每次增加 1，但不超过 10
17      node3 = Candidate: min(timer3 + 1, 10); -- Candidate 每次增加 1，但不超过 10
18  esac;

```

根据 Raft 协议规定，描述节点之间的状态转移关系：

```

1  -- 状态转移：若定时器超时，则成为 Candidate，若选举成功，则成为 Leader
2  next(node1) := case
3      timer1 >= 5: Candidate; -- 如果定时器超过 5，则成为 Candidate
4      node1 = Candidate & timer1 < 5: Follower; -- 如果处于 Candidate 状态，未超时则变回
Follower
5      node1 = Leader: Leader; -- 如果是 Leader，则保持 Leader
6      TRUE: node1; -- 否则保持当前状态
7  esac;
8
9  next(node2) := case
10     timer2 >= 5: Candidate; -- 如果定时器超过 5，则成为 Candidate
11     node2 = Candidate & timer2 < 5: Follower; -- 如果处于 Candidate 状态，未超时则变回
Follower
12     node2 = Leader: Leader; -- 如果是 Leader，则保持 Leader
13     TRUE: node2; -- 否则保持当前状态
14  esac;
15
16  next(node3) := case
17     timer3 >= 5: Candidate; -- 如果定时器超过 5，则成为 Candidate
18     node3 = Candidate & timer3 < 5: Follower; -- 如果处于 Candidate 状态，未超时则变回
Follower
19     node3 = Leader: Leader; -- 如果是 Leader，则保持 Leader
20     TRUE: node3; -- 否则保持当前状态
21  esac;

```

本项目实现了三个节点的 Raft 协议，可以轻松扩展到多个节点。

使用以下规约验证协议一致性和安全性：

```
1  -- 规约 1: 三个节点都可能成为 Leader
2  CTLSPEC AG (node1 = Leader -> EF (node2 = Leader | node3 = Leader)) -- 结果为 TRUE
3
4  -- 规约 2: 不会出现多个 Leader
5  CTLSPEC AG (node1 = Leader & node2 = Leader -> FALSE) -- 结果为 TRUE
6  CTLSPEC AG (node1 = Leader & node3 = Leader -> FALSE) -- 结果为 TRUE
7  CTLSPEC AG (node2 = Leader & node3 = Leader -> FALSE) -- 结果为 TRUE
8
9  -- 规约 3: 可能有多个节点同时想成为 Leader
10 CTLSPEC EF (node1 = Candidate & node2 = Candidate) -- 结果为 TRUE
11 CTLSPEC EF (node2 = Candidate & node3 = Candidate) -- 结果为 TRUE
12 CTLSPEC EF (node1 = Candidate & node3 = Candidate) -- 结果为 TRUE
13 CTLSPEC EF (node1 = Candidate & node2 = Candidate & node3 = Candidate) -- 结果为 TRUE
```