Advances in Applied Mathematics 应用数学进展, 2022, 11(11), 7729-7737
Published Online November 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.1111817

Hans汉斯

k-归纳模型检测结果的认证器

刘　帅，魏峰玉，黄怡桐，江建国*

辽宁师范大学数学学院，辽宁　大连

收稿日期：2022年10月7日；录用日期：2022年11月1日；发布日期：2022年11月10日

摘　要

基于*k*-归纳的模型检测可以验证许多重要的系统，但在具体的实现过程中仍可能存在缺陷，所以对其结果进行认证是非常必要的。目前最有效的方法是利用*k*-证据电路的概念扩展给定的模型检测问题，然后使用认证器进行判定，但是在部分情况下，QBF的验证大大增加了认证时间。为了解决这一问题，本文提出对复位检测的算法进行优化，创建了一个新的复位条件并添加了带映射的原始电路来减小QBF。并且本文还使用C语言实现了认证器。实验结果表明：利用优化算法生成的QBF有效减少了认证时间，利用C语言实现的认证器也为之后的研究提供了有力工具。

关键词

***k*-归纳，模型检测，认证器，*k*-证据电路**

A Certifier for *k*-Induction Model Checking Results

Shuai Liu, Fengyu Wei, Yitong Huang, Jianguo Jiang*

School of Mathematics, Liaoning Normal University, Dalian Liaoning

Received: Oct. 7th, 2022; accepted: Nov. 1st, 2022; published: Nov. 10th, 2022

Abstract

Model checking based on *k*-induction can verify many important systems, but there may still be defects in the implementation process, so it is very necessary to verify the results. At present, the most effective method is to extend the given model checking problem by using the concept of *k*-witness circuit, and then use the certifier to determine. However, in some cases, the verification of QBF greatly increases the certification time. In order to solve this problem, this paper proposes

*通讯作者。

文章引用: 刘帅, 魏峰玉, 黄怡桐, 江建国. *k*-归纳模型检测结果的认证器[J]. 应用数学进展, 2022, 11(11): 7729-7737.
DOI: 10.12677/aam.2022.1111817

to optimize the reset checking algorithm, create a new reset condition and add the original circuit with mapping to reduce QBF. And this paper also uses C language to achieve the certification. The experimental results show that the QBF generated by the optimization algorithm effectively reduces the certification time, and the certifier implemented by C language also provides a powerful tool for future research.

Keywords

***k*-Induction, Model Checking, Certifier, *k*-Witness Circuit**

1. 引言

随着科学技术的不断发展，计算机系统的规模和功能得到了不断扩充，因此，对计算机系统正确性的验证日益重要。目前，计算机系统的验证主要采用模拟法和形式化验证。其中，形式化验证采用数学化的方法证明计算机系统的正确性，相对于严重依赖测试向量的模拟法来说，更能保证验证的完备性。

形式化验证主要分为两大类：定理证明和模型检测。其中，模型检测是指通过显式状态搜索或隐式不动点计算来验证有穷状态并发系统的命题性质。相比于定理证明而言，模型检测的自动化程度更高，所以模型检测的应用更为广泛。

为了完全自动化的证明系统的正确性，验证算法是研究的核心内容。通过验证算法，将需要验证的问题转化为逻辑公式，再借助于自动化的逻辑推理工具证明逻辑公式的正确性。事实上，验证算法有很多，例如谓词抽象、路径抽象、*k*-归纳[1]等等。

k-归纳可以将模型检测简化为一系列 SAT 问题。基于 *k*-归纳的模型检测可以通过对状态机进行安全可靠的分析，从而验证许多重要的系统。但在具体的实现过程中，开发人员在处理异常方面拥有很大的自由度，这就意味着即使规范已经被证明，但仍可能存在缺陷。所以为了弥补可能存在的缺陷，即提高模型检测结果的可信度，于是提出对基于 *k*-归纳的模型检测结果进行认证。

为了认证基于 *k*-归纳的模型检测结果，人们花费了大量时间提出了各种方法[2]-[8]，但是有些方法无法直接适用于普通形式的 *k*-归纳，有些方法产生的 *k*-归纳特定证书不提供归纳不变式，还有些方法产生的认证证书是指数级的。所以，目前来说 Emily Yu 等人[9]所介绍的认证方法是最有效的，他们利用 *k*-证据电路的概念，扩展给定的模型检测问题，得到一个更大电路的简单归纳不变式作为原始电路的 *k*-归纳证明，将认证简化为固定数量的 SAT 问题和一个 QBF 问题，然后利用独立的认证机构进行判定，从而得到基于 *k*-归纳的模型检测结果的认证结果。

Emily Yu 等人所研发的认证器 Certifiger [10]独立于任何模型检测器，工作效率很高，但还是有一些不足之处。他们使用 python 语言编写认证器代码，使认证器整体有两种编程语言，故语法结构较为松散随便，非常不利于后期的调试和开发。

因此，本文致力于在 Emily Yu 等人的基础上利用 C 语言[11]重新实现基于 *k*-归纳的模型检测结果的认证器。由于认证器调用的所有组件都由 C 语言进行编译，所以本文利用 C 语言编写代码从而统一认证器的编程语言，使认证器更加完整，为之后的使用、维护、调试、开发和优化都提供了便利。

2. 基本概念

设 $B(V)$ 是布尔变量 V 上的布尔表达式集，给定两个布尔表达式 $f(V),g(V)\in B(V)$，如果它们有相同的模型，则称它们为等价物，写为 $f(V)\equiv g(V)$。用“$\simeq$”表示语法等价[12]，“$\to$”表示语法蕴涵，“$\Rightarrow$”表示语义蕴涵，以及等式“=”。在本文中，只关注简单的安全属性[5]。

定义 2.1. 电路 $C=(I,L,R,F,P)$ 的定义如下：

1) I 是布尔输入变量集；

2) L 是布尔锁存变量集；

3) $R=\{r_l(L)\,|\,l\in L\}$ 是复位函数公式集；

4) $F=\{f_l(I,L)\,|\,l\in L\}$ 是转换函数集，对于每个锁存器 $l\in L$，都有对应的转换函 $f_l(I,L)\in B(I,L)$；

5) $P(I,L)\in B(I,L)$ 是编码(好状态)属性的公式。

定义 2.2. 对于展开深度 $m\in\mathrm{N}$，长度为 m 的电路 C 的展开定义为 $U_m=\bigwedge_{i\in[0,m)}\left(L_{i+1}\simeq F(I_i,L_i)\right)$。

定义 2.3. 给定电路 $C=(I,L,R,F,P)$ 和 $C'=(I',L',R',F',P')$，C'组合扩展 C，若 $I=I'$ 和 $L\subseteq L'$。

定义 2.4. 给定电路 $C=(I,L,R,F,P)$ 和 $C'=(I',L',R',F',P')$，当 C'组合扩展 C 时，则 C'组合仿真 C，如果以下条件成立：

1) $f_l(I,L)\equiv f'_l(I,L')$；

2) $P'(I,L')\Rightarrow P(I,L)$；

3) $R(L)\Rightarrow\exists(L'\setminus L)R'(L')$。

定义 2.4. 1) 称为转换检测，定义 2.4. 2)称为属性检测，定义 2.4. 3)称为复位检测。

定义 2.5. 一个公式 ϕ 为电路 C 的归纳不变式，如果 ϕ 满足以下条件：

1) $R(L)\Rightarrow\phi(I,L)$；

2) $\phi(I,L)\Rightarrow P(I,L)$；

3) $U_1\wedge\phi(I_0,L_0)\Rightarrow\phi(I_1,L_1)$。

定义 2.5. 1) 称为初始性检测，定义 2.5. 2) 称为一致性检测，定义 2.5. 3) 称为连续性检测。

定义 2.6 给定一个性质为 P 的电路 C，定义公式 $S_k=\bigwedge_{i\in[0,k)}P(I_i,L_i)$。当且仅当以下两个条件成立时，$P$ 在 C 中称为 k-归纳的：

1) $U_{k-1}\wedge R(L_0)\Rightarrow S_k$；

2) $U_k\wedge S_k\Rightarrow P(I_k,L_k)$。

当 $\phi(I,L)\equiv P(I,L)$ 时，1-归纳不变式等价于归纳不变式。

3. k-证据电路

在下面的定义中，用 L^i 中的上标 i 表示锁存器 L 在空间方向上的副本，其中 $l^i\in L^i$ 是 $l\in L$ 的相应副本，输入也是如此。初始化位 B 取决于各自的初始化状态。

定义 3.1. 给定电路 $C=(I,L,R,F,P)$，和 $k\in\mathrm{N}^+$，C 的 k-证据电路 $\boxed{C'=(I',L',R',F',P')}$ 定义如下：

1) $\boxed{I'}=I$ 为了简单可见，我们也称 I' 为 X^{k-1}；

2) $\boxed{L'}=X^0\cup\cdots\cup X^{k-2}\cup L^0\cup\cdots\cup L^{k-1}\cup B$：

a) X^i 是原始输入的副本，对于所有的 $i\in[0,k-2]$；

b) L^i 是原始锁存器的副本，对于所有的 $i\in[0,k-1]$；

c) $B=\left\{b^0,\cdots,b^{k-1}\right\}$ 是初始化位的集合；

3) 重置函数 $\boxed{R'}=\left\{r_l'\left(L'\right)\middle|l\in L'\right\}$ 定义如下：

a) 对于 $x\in X^0\cup\cdots\cup X^{k-2}$，$r_x'=x$；

b) 对于 $i\in[1,k-1)$，$u^i=R\left(L^i\right)\vee u^{i+1}$，且 $u^{k-1}=R\left(L^{k-1}\right)$；

c) 对于 $l\in L^0$，$r_l'=ite\left(u^1,l,r_l\left(L^0\right)\right)$；

d) 对于 $i\in[1,k)$，$r_{l^i}'=ite\left(u^i,l^i,f_{l^i}\left(X^{i-1},L^{i-1}\right)\right)$；

e) $r_{b^{k-1}}'=\mathrm{T}$；

f) $r_{b^0}'=\neg u^1$；

g) 对于 $i\in[1,k-1)$，$r_{b^i}'=b^{i-1}\vee\left(R\left(L^i\right)\wedge\neg u^{i+1}\right)$；

4) $\boxed{F'}=\left\{f_l'\left(I',L'\right)\middle|l\in L'\right\}$ 被定义如下：

a) 对于 $i\in[0,k-1)$，$f_{x^i}'\left(I',L'\right)=x^{i+1}$；

b) 对于 $l\in L^{k-1}$，$f_l'\left(I',L'\right)=f_l\left(X^{k-1},L^{k-1}\right)$；

c) 对于 $i\in[0,k-1)$，$f_{l^i}'\left(I',L'\right)=l^{i+1}$；

d) 对于 $i\in[0,k-1)$，$f_{b^i}'\left(I',L'\right)=b^{i+1}$ 且 $f_{b^{k-1}}'\left(I',L'\right)=b^{k-1}$；

5) 性质 $\boxed{P'}$ 被定义为 $P'\left(I',L'\right)=\bigwedge\limits_{i\in[0,4]}p_i\left(I',L'\right)$：

a)对于 $i\in[0,k-1)$，$h^i=\left(L^{i+1}\simeq F\left(X^i,L^i\right)\right)$；

b) $p_0\left(I',L'\right)=\bigwedge\limits_{i\in[0,k-1)}\left(b^i\to b^{i+1}\right)$；

c) $p_1\left(I',L'\right)=\bigwedge\limits_{i\in[0,k-1)}\left(b^i\to h^i\right)$；

d) $p_2\left(I',L'\right)=\bigwedge\limits_{i\in[0,k)}\left(b^i\to P\left(X^i,L^i\right)\right)$；

e) $p_3\left(I',L'\right)=\bigwedge\limits_{i\in[1,k)}\left(\left(\neg b^{i-1}\wedge b^i\right)\to R\left(L^i\right)\right)$；

f) $p_4\left(I',L'\right)=b^{k-1}$。

定理 3.1. 电路 C 组合仿真其 k-证据电路。

定理 3.2. 给定电路 C，一个固定的 $k\in\mathrm{N}^+$，以及它的 k-证据电路 C'，P 在 C 中是 k-归纳的，当且仅当 P'在 C'中是 1-归纳的。

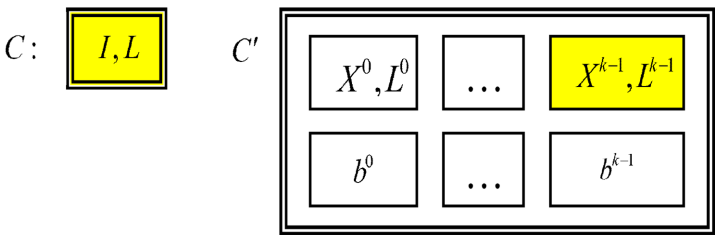

Figure 1. The structure of input and latch variables in C and C'
图 1. C 和 C'中输入和锁存变量的结构

图 1 显示了原始电路及其 k-证据电路的可变结构的比较，标记为黄色的区域(左框和位于右侧的右上框)是由同一组变量组成。

4. 认证器

4.1. 认证原理

基于 k-归纳的模型检测结果的认证原理是首先利用 k-证据电路的概念，得到给定电路 C 的 k-证据电路 C'，由定理 3.1 可知 C'组合仿真，然后根据定理 3.2 可知，当 P 在 C 中是 k-归纳的，当且仅当 P'在 C'中是 1-归纳的，故 P'是 C'的归纳不变式。所以为了得到认证结果，对归纳不变式 P'的初始性检测、连续性检测和一致性检测逐一判定，而其中为了保证认证的正确性，对组合仿真概念的转换检测、属性检测和复位检测也进行判定，如果组合仿真和归纳不变式的六个检测都验证成功，那么这意味着电路 C'组合仿真 C，并且 P'是 C'的归纳不变式，故基于 k-归纳的模型检测结果的认证是成功的。

认证器是为了符合上述认证原理而且能够证实基于 k-归纳的模型检测结果的正确性而设计的。故认证器应具备以下功能：

1) 可以读取输入文件；
2) 可以选择任意组件进行操作；
3) 能够返回任意组件运行所生成的文件；
4) 对返回的文件进行操作；
5) 可以得到认证是否正确。

根据需求分析，需要实现一个由多个组件组成的认证器，该认证器将包含一个 AIGER 格式[13]的电路和一个由开源的基于 k-归纳的模型检测器 Mcaiger [14]所提供的 k 值作为输入，输出一个积极的模型检测结果的认证结果。

4.2. 优化算法

本文对认证器组合仿真的复位检测提出了优化算法。由于在部分情况下，QBF 的求解过程比其他公式复杂很多，使整个认证过程花费了过多的时间、空间和计算量。所以为了解决这个问题，本文创建一个新的复位条件并添加了带映射的原电路，利用 C 和 C'的复位值和其负值动态的为电路 C 添加与门直到添加结束，并将带映射的电路放入到指针变量中。这种算法大大减少了验证复位检测时所需的时间、空间和计算量。

算法 1 给出了复位检测的优化算法。此算法主要包括以下几个步骤：1) 将 C 的与门与 C 的输入个数、与 C 的输入个数与 C 的锁存个数的和与 C 的总个数分别进行比较，再向 ci 中添加带映射的电路；2) 对锁存个数进行比较，若锁存个数为 0，则将终止符 1 写入到输出文件中；3) 若锁存个数不为 0，则将 ci 作为与门的和值，将 C 的锁存器的复位值或者其负值作为与门的左值，并将 C'的锁存器的复位值负值或者其原值作为与门的右值，通过循环变量的增加动态地增加做与门的次数，而锁存器变量个数的有限性保证了循环过程最终得以终止；4) 将得到的复位条件写入到输出文件中，以便提供给求解器进行计算。若计算结果为不可满足的，则复位检测条件认证成功，否则认证失败。

与原来的算法相比较而言，本文所提出的优化算法直接根据 C 的锁存器的复位值作与门，不需要像原来算法对 C 的输入、锁存和与门以及 C'的输入、锁存还有与门等全部复制 $k-1$次，进而出现重复遍历的情况。本文所提出的算法提高了认证器的性能，使整个认证过程更加清晰明了。

算法 1：复位检测优化算法

输入：AIGER 格式的电路 C 与 C'

输出：复位检测的输出文件 file

1. $ci \leftarrow$ 添加带映射的电路
2. **if** C 的锁存个数=0 **then**
3. $con4 \leftarrow 1$
4. **else**
5. **for** $i \leftarrow 0$ **to** C'的锁存个数 **do**
6. $ci \leftarrow ci + 2$
7. $con4 \leftarrow$ aiger_add_and()
8. $a1 \leftarrow ci + 2$
9. $ci \leftarrow ci + 2$
10. $con4 \leftarrow$ aiger_add_and()
11. $a2 \leftarrow ci + 2$
13. $ci \leftarrow ci + 2$
14. $con4 \leftarrow$ aiger_add_and()
15. **end for**
16. file $\leftarrow$ con4
17. **end if**

4.3. C 语言实现

本文使用 C 语言对认证器重新实现，由于认证器的输入和需要调用的所有组件都是由 C 语言编译的，所以本文利用 C 语言编译认证器的代码，使之整个认证器的代码更加完整，并且为相关研究人员之后的使用、维护、调试以及开发都提供了一个有力工具。

本文利用 C 语言对基于 k-归纳的模型检测结果的认证器进行实现，利用函数 popen()通过建立管道启动各个组件，然后利用函数 fread()从各个组件运行后所得到的文件中读取数据到内存缓冲区。并且使用函数 gettimeofday()得到各个组件运行花费的时间，从而得到整个认证器运行各个实例所花费的时间。本文还利用函数 fopen()打开并读取文件，从而得到 C、C'以及六个检测的电路大小、输入个数、与门个数等信息。

在认证器的运行过程中，首先为了读取输入文件，调用主函数将 AIGER 格式的具体实例传递给认证器。之后通过 mcaiger-n 调用开源的基于 k-归纳的模型检测器 Mcaiger，使 Mcaiger 对实例进行模型检测并且返回 k 值。再利用 aigcertify_kind-o 将 C 和 k 值传入到 k-证据生成器中，从而生成 k-证据电路 C'。接着通过 mapping_check-k 调用组合仿真检测器，该检测器生成两个 AIGER 格式和一个 QAIGER 格式的文件。随后使 aigcertify-o 调用归纳不变式检测器，该检测器生成三个 AIGER 格式的文件。然后调用工具 aigtocnf 将 AIGER 格式的文件转换为 CNF，利用 kissat-q 调用求解器 Kissat [15]对 CNF 求解。最后使用文件 qaiger2qcir 将 QAIGER 格式的 QBF 转换为 QCIR 格式，然后调用求解器 QuAbs [16]对 QBF 求解，当求解全部完成，就可得到认证是否成功的结论。认证器的工作流程如图 2 所示。

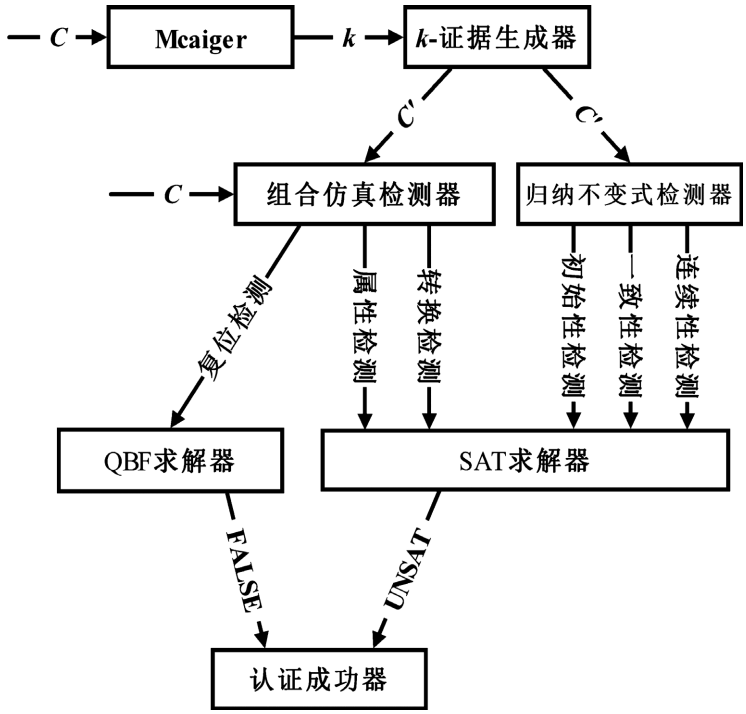

Figure 2. The Dataflow diagram of the Certifier
图 2. 认证器的数据流程图

4.3. 实验结果

本文实验选择 TIP suite [17]和 HWMCC10 [18]两种测试基准。本次实验使用了一台带有 Ubuntu18.04 虚拟机的电脑，配备 Intel i5-8250U 1.60 GHz CPU 和 4 GB 主内存。实验中所用的时间都是以秒为单位，电路的规模是就门的个数而言。

本文在使用 TIP suite 基准进行实验前，通过调用 AIGER 库中的 smvtoaig，将基准从 SMV 文件转换为 AIGER 格式的文件。在 TIP suite 基准中，为了确保一个正确的认证器的正常运行，事先对因为 k 值无意义或者 k 值过大导致运行时花费的时间过长从而认证失败的实例进行了预先过滤。实验中要求所选取的实例满足 k 值在 4 和 96 之间变化，并且 SAT 求解器能够处理而不出现超时情况。表 1 报告了 TIP suite 基准的实验结果。

Table 1. Experimental results for the TIP suite
表 1. TIP suite 的实验结果

文件名	k	规模	时间
c.periodic	96	1560	208.92
n.guidance_1	10	1910	6.64
n.guidance_7	27	2000	56.57
n.tcasp_2	6	3020	4.18
n.tcasp_3	5	2980	3.96
v.prodcell_{12}	29	2910	135.04
v.prodcell_{13}	8	2910	5.92

Continued

v.prodcell$_{14}$	16	2910	33.95
v.prodcell$_{15}$	23	2910	64.72
v.prodcell$_{16}$	5	2910	2.64
v.prodcell$_{17}$	27	2910	113.16
v.prodcell$_{18}$	13	2910	18.88
v.prodcell$_{19}$	22	2910	59.52
v.prodcell$_{24}$	37	2910	270.47

本文还使用了 2010 年硬件模型检测竞赛(HWMCC10)的基准。通过将各个实例分别在 Mcaiger 上运行 15 分钟，从而对基准进行预先过滤，并且从中排除了无意义 k 值的实例和需要简单路径约束的实例。由于同一组的基准 k 值基本相同，所以尽量挑选不在同一组的各个基准。表 2 报告了 HWMCC10 的部分基准的认证结果。

Table 2. Experimental results for the HWMCC10
表 2. HWMCC10 的实验结果

文件名	k	规模	时间
bj08amba2g1	3	950	0.51
eijks344	6	340	0.74
bobsmdct	88	1250	3231.84
bj08aut82	3	380	0.06
mentorbm1p02	2	36,290	407.52
nusmvtcasp3	5	2980	3.08
pdtswvibs8x8p1	39	3560	206.98
pdtvishuffman6	10	1030	2.18
pdtvisvsa16a04	2	6520	1.42
pdtvisvsar29	4	2530	2.65
pj2010	9	18,820	101.75
visprodcellp03	3	2910	0.88
pdtvisminmaxr1	2	950	0.13

如表 1 和表 2 所示，对于基于 k-归纳的模型检测结果的认证，展示了两种不同类型基准的实验结果。实验结果表明：本文利用 C 语言重新实现的认证器可以对模型检测结果进行认证，而且得到不同实例的 k 值、C 的电路大小和认证所花费的时间。

5. 结论

本文通过 C 语言重新实现了基于 k-归纳的模型检测结果的认证器，为后续的研究提供了一个有力的认证工具。在以后的工作中，我们希望可以将该方法扩展到常见的预处理技术，甚至扩展到更具挑战性的无限状态的系统中。

参考文献

[1] Sheeran, M., Singh, S. and Stlmarck, G. (2000) Checking Safety Properties Using Induction and a SAT-Solver. *Formal Methods in Computer-Aided Design*, *Third International Conference*, *FMCAD* 2000, Austin, 1-3 November 2000, 127-144. https://doi.org/10.1007/3-540-40922-X_8

[2] Bingham, J.D. (2008) Automatic Non-Interference Lemmas for Parameterized Model Checking. *Formal Methods in Computer-Aided Design*, *FMCAD* 2008, Portland, 17-20 November 2008, 1-8. https://doi.org/10.1109/FMCAD.2008.ECP.15

[3] Griggio, A., Roveri, M. and Tonetta, S. (2018) Certifying Proofs for LTL Model Checking. 2018 *Formal Methods in Computer-Aided Design* (*FMCAD*), Austin, 30 October 2018-2 November 2018, 1-9. https://doi.org/10.23919/FMCAD.2018.8603022

[4] Gurfinkel, A. and Ivrii, A. (2017) K-Induction without Unrolling. *Proceedings of the* 17*th Conference on Formal Methods in Computer-Aided Design*, Vienna, 2-6 October 2017, 148-155. https://doi.org/10.23919/FMCAD.2017.8102253

[5] Kuismin, T. and Heljanko, K. (2013) Increasing Confidence in Liveness Model Checking Results with Proofs. Springer, Cham. https://doi.org/10.1007/978-3-319-03077-7_3

[6] Namjoshi, K.S. (2001) Certifying Model Checkers. Springer-Verlag, Berlin. https://doi.org/10.1007/3-540-44585-4_2

[7] Wagner, L., Mebsout, A., Tinelli, C., *et al*. (2017) Qualification of a Model Checker for Avionics Software Verification. Springer, Cham. https://doi.org/10.1007/978-3-319-57288-8_29

[8] Yu, Z., Biere, A. and Heljanko, K. (2019) Certifying Hardware Model Checking Results. In: Ait-Ameur, Y. and Qin, S.C., Eds., *International Conference on Formal Engineering Methods*, Springer, Cham, 498-502. https://doi.org/10.1007/978-3-030-32409-4_32

[9] Yu, E., Biere, A. and Heljanko, K. (2021) Progress in Certifying Hardware Model Checking Results. In: Silva, A., Rustan, K. and Leino, M., Eds., *International Conference on Computer Aided Verification*, Springer, Cham, 363-386.

[10] Certifaiger (2021). http://fmv.jku.at/certifaiger

[11] Kernighan, B.W., Ritchie, D.M. 程序设计语言 C [M]. 第 2 版. 北京: 机械工业出版社, 2004.

[12] Degtyarev, A. and Voronkov, A. (2001) Equality Reasoning in Sequent-Based Calculi. In: Robinson, A. and Voronkov, A., Eds., *Handbook of Automated Reasoning*, Vol. 1, Elsevier, Amsterdam, 611-706. https://doi.org/10.1016/B978-044450813-3/50012-6

[13] Biere, A., Heljanko, K. and Wieringa, S. (2011) AIGER 1.9 and Beyond.

[14] Biere, A. and Brummayer, R. (2008) Consistency Checking of All Different Constraints over Bit-Vectors within a SAT Solver. 2008 *Formal Methods in Computer-Aided Design*, Portland, 17-20 November 2008, 1-4. https://doi.org/10.1109/FMCAD.2008.ECP.32

[15] Biere, A., Fazekas, K., Fleury, M. and Heisinger, M. (2020) CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020.

[16] Tentrup, L. (2016) Non-Prenex QBF Solving Using Abstraction. In: Creignou, N. and Le Berre, D., Eds., *International Conference on Theory and Applications of Satisfiability Testing*, Springer, Cham, 393-401. https://doi.org/10.1007/978-3-319-40970-2_24

[17] Eén, N. and Sörensson, N. (2003) Temporal Induction by Incremental SAT Solving. *Electronic Notes in Theoretical Computer Science*, **89**, 543-560. https://doi.org/10.1016/S1571-0661(05)82542-3

[18] Biere, A. and Claessen, K. (2010) Hardware Model Checking Competition 2010. http://fmv.jku.at/hwmcc10