ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.5, May 2006, pp.1034−1041
DOI: 10.1360/jos171034

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于悖论分析和增量求解的快速反例压缩算法*

沈胜宇+, 李思昆

(国防科学技术大学 计算机学院,湖南 长沙 410073)

A Fast Counterexample Minimization Algorithm with Refutation Analysis and Incremental SAT

SHEN Sheng-Yu+, LI Si-Kun

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573678 ext 801, Fax 86-731-4511529, E-mail: syshen@nudt.edu.cn, http://www.nudt.edu.cn

Shen SY, Li SK. A fast counterexample minimization algorithm with refutation analysis and incremental SAT. *Journal of Software*, 2006,17(5):1034−1041. http://www.jos.org.cn/1000-9825/17/1034.htm

Abstract: It is a hot research topic to eliminate irrelevant variables from counterexample, to make it easier to be understood. BFL (brute force lifting) algorithm is the most effective one among all the existing approaches, but its time overhead is very large due to one call to SAT (boolean satisfiability problem) solver per candidate variable to be eliminated. So a faster algorithm based on refutation analysis and incremental SAT is proposed. The counterexample minimization problem is first translated into a set of SAT instances for each free variable *v*, to determine if *v* can prevent the counterexample. Then refutation analysis on those UNSAT (unsatisfiable) instances is performed, to extract the set of variables that lead to refutation. All variables that don't belong to this set can be eliminated directly as irrelevant variables. At the same time, the incremental SAT approach is employed to share conflict clauses between similar instances, such that the overlapped state space can be prevented from being searched repeatedly. Theoretical analysis and experimental result show that this approach run much faster than BFL algorithm, and with minor lost in reduction rate.

Key words: model checking; counterexample minimization; refutation analysis; incremental SAT; safety assertion

摘 要: 使用反例压缩算法,从反例中剔除冗余信息,从而使反例易于理解,是目前的研究热点.然而,目前压缩率最高的 BFL(brute force lifting)算法,其时间开销过大.为此,提出一种基于悖论分析和增量式 SAT(boolean satisfiability problem)的快速反例压缩算法.首先,根据反证法和排中律原理,该算法对每一个自由变量 *v*,构造一个 SAT 问题,以测试 *v* 是否能够避免反例.而后对其中不可满足的 SAT 问题,进行悖论分析,抽取出导致悖论的变量集合.所有不属于该集合的变量,均可作为无关变量直接剔除.同时,该算法使用增量式 SAT 求解方法,以避免反复搜索冗余状态空间.理论分析和实验结果表明,与 BFL 算法相比,该算法能够在不损失压缩率的前提下获得 1~2 个数量级的加速.

关键词: 模型检验;反例压缩;悖论分析;增量式求解;安全断言
中图法分类号: TP301 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60573173 (国家自然科学基金)
Received 2004-07-02; Accepted 2005-10-27

近年来,模型检验方法成为复杂硬件系统验证的重要手段.该方法有两方面的优点:首先,模型检验方法不需要激励码,能自动完成验证工作;其次,当断言被违反时,模型检验方法能够给出反例,以解释其原因.

然而,在复杂硬件系统的验证中,模型检验方法给出的反例非常复杂,需要花费大量的时间才能找到真正的错误[1].因此,使用自动化方法从反例中剔除无关信息是目前的研究热点[1,2].

同时,剔除反例中的无关信息也将极大地加速许多验证算法.这些算法包括基于 SAT(boolean satisfiability problem)的抽象精化模型检验方法[3]和基于 SAT 的映象算法[4−6].在这些算法中,如果能从反例中抽取出一个较小的变量集合,使得该集合是导致反例的充分条件,则该集合就可以代表大量的反例.我们称这个变量集合为压缩集,并称抽取压缩集的过程为反例压缩.文献[4,5]指出,存储和处理压缩集比存储和处理反例集合,可在空间和时间复杂度上获得指数改进.

因此,无论是在便于反例理解方面,还是在提高验证算法的效率方面,反例压缩都具有重大的意义.

Ravi[1]提出了 BFL(brute force lifting)反例压缩算法.该方法对每一个变量 v,构造一个 SAT 问题 $SAT(v)$,以测试 v 是否能够避免反例.若 $SAT(v)$不可满足,则表明 v 是无关变量,可从压缩集中剔除.文献[1]的实验结果表明,BFL 通常能够剔除 70%以上的无关变量,是目前压缩率最高的方法,但其时间开销远大于其他方法.

BFL 算法时间开销过大的原因在于:

1. 每运行一次 SAT 求解器只能剔除一个变量,从而导致 SAT 求解器的运行次数过多;
2. 多个 SAT 问题之间有大量重叠的状态空间,独立求解这些问题将会反复搜索这些重叠空间.

因此,本文提出一种新的反例压缩算法,以降低 BFL 的时间开销:

1. 对 BFL 中不可满足的 $SAT(v)$进行悖论分析,提取导致悖论的压缩集 R'',并一次性剔除所有 $v\notin R''$;
2. 使用增量式 SAT 求解方法,在多个 SAT 问题之间共享冲突短句,避免反复搜索重叠的状态空间.

我们通过修改 zchaff 求解器[7]和 NuSMV 模型检验器[8]实现了该算法.本文使用 ISCAS89 测试集[9]作为实验例子.实验结果表明,本算法能够极大地压缩反例,且相对于文献[1]能够获得 1~2 个数量级的加速.

本文第 1 节介绍背景知识.第 2 节给出悖论分析算法的描述和证明.第 3 节给出 BFL 和本文算法的复杂性分析.第 4 节为增量式求解算法.第 5 节为实验结果.第 6 节为相关工作.

1 背景知识

1.1 限界模型检验

我们首先给出待验证的 Kripke 结构的定义.

定义 **1**. Kripke 结构 $M=(S,I,W,T,A,L)$中:S 为状态全集;$I\subseteq S$ 为初始状态集合;W 为输入变量集合;$T:S\times W\times S\to\{0,1\}$为状态转换关系;$A$ 为命题全集;$L:S\to 2^A$ 表示在状态 S 下成立的命题集合 $L(S)\subseteq A$.

与传统的符号模型检验[10]不同,限界模型检验(bounded model checking,简称 BMC)[11]仅考虑有限长度路径,称该路径的长度为其限界.记第 i 个和第 i+1 个周期的状态为 S_i 和 S_{i+1},状态转换关系为 $T_i(S_i,W_i,S_{i+1})$,其中 W_i 为第 i 个周期的输入变量.设有待验证的安全断言为 *ASSERT*,则限界模型检验的目的在于搜索对 *ASSERT* 的违反,即 $P=\neg ASSERT$.记第 i 个时钟周期的 P 为 P_i,则限界为 k 的限界模型检验问题记为

$$F=I\wedge\bigwedge_{0\le i<k}\neg P_i\wedge P_k\wedge\bigwedge_{0\le i<k}T_i(S_i,W_i,S_{i+1}) \tag{1}$$

在上式中,I 表示初始状态,$\bigwedge_{0\le i<k}\neg P_i$ 表示在第 0 个~第 k−1 个周期,都不违反断言 *ASSERT*,P_k 表示在第 k 个周期断言 *ASSERT* 被违反,$\bigwedge_{0\le i<k}T_i(S_i,W_i,S_{i+1})$表示从第 0 到第 k 个周期的状态转换关系.

在限界模型检验方法中,将式(1)转换为 CNF 格式的 SAT 问题,并用 SAT 求解器求解,即可验证 *ASSERT*.

1.2 SAT求解器中的冲突学习机制

在 SAT 求解过程中,每当发生冲突,SAT 求解器将分析蕴含图(implication graph)以构造学习短句(learned clause),并将其插入短句库.学习短句记录了已被搜索的状态空间,以防止它们被反复搜索[12].

因此,当 SAT 求解器完成求解的时候,在短句库中有两种类型的短句:原始短句和学习短句.前者是指在

SAT 求解器开始求解之前被插入短句库的短句.

1.3 BFL反例压缩算法及其问题

为了便于论述,我们在定义 2 中给出一些相关的记法和符号.

定义 **2**. 令反例的限界为 k,则其自由变量集合记为 $Free=I\cup\cup_{0\le i\le k}\cdot W_i$.变量 v 在反例中的取值记为 $Assign(v)$,变量集合 V 在反例中的取值记为 $Assign(V)=\{Assign(v)|v\in V\}$.

v 是无关变量,当且仅当"v 取任何值均不能避免产生反例,均不能避免式(1)的 F 等于 1".严格定义如下:

定义 **3**(无关变量). 对自由变量 $v\in Free$,v 是无关变量的充要条件是

$$\begin{aligned}&\forall c\in\{0,1\}.F(v\Leftarrow c,Free-\{v\}\Leftarrow Assign(Free-\{v\}))==1\\&\Leftrightarrow\neg\exists c\in\{0,1\}.F(v\Leftarrow c,Free-\{v\}\Leftarrow Assign(Free-\{v\}))==0\end{aligned}\tag{2}$$

将式(2)中的 $F(v\Leftarrow c,Free-\{v\}\Leftarrow Assign(Free-\{v\}))==0$ 转变为 SAT 问题,则 v 是无关变量,当且仅当该 SAT 问题不可满足.因此,进行反例压缩以抽取压缩集的 BFL 算法如下:

算法 **1**. BFL 算法.

1. $F''=\neg P_k\wedge\bigwedge_{0\le i<k}\cdot T_i(S_i,W_i,S_{i+1})$ //等同于验证式(2)
2. for each $v\in Assign(Free)$
3. $F'=F''\wedge(Free-\{v\}\Leftarrow Assign(Free-\{v\}))$ //除了 v 以外,所有变量的值都等于反例中的取值
4. if ($SAT\_Solve(F')==UNSAT$) $Free=Free-\{v\}$ //v 取任何值都不能使$\neg P_k$成立,故 v 可以剔除
5. $Free$ 即为压缩集

对于算法 1 中步骤 3 的 F',其原始短句集合包含两个子集:模型短句集合和赋值短句集合.其定义如下:

定义 **4**. 模型短句集合:在算法 1 的步骤 3 中,由 F'' 生成的短句集合.

定义 **5**. 赋值短句集合:在算法 1 的步骤 3 中,由$(Free-\{v\}\Leftarrow Assign(Free-\{v\}))$生成的短句集合.

我们称前者为模型短句集合,因为 F'' 代表式(1)的模型检验问题;称后者为赋值短句集合,因为其作用在于使 $Free-\{v\}$的取值等于 $Assign(Free-\{v\})$.对变量 $v'\in Free-\{v\}$,其赋值短句仅包含一个文字,若 $Assign(v')==1$,则 v'的赋值短句为$\{v'\}$;否则为$\{\neg v'\}$.SAT 求解器将根据赋值短句完成赋值 $v'\Leftarrow Assign(v')$.

2 基于悖论分析的反例压缩算法

2.1 算法概述

为了在保持 BFL 算法[1]压缩效果的前提下降低时间开销,我们提出了基于悖论分析的反例压缩算法.该算法与 BFL 算法的差别在于,算法 1 的步骤 4 完成后,若结果是 UNSAT(unsatisfiable),则仅仅剔除一个变量 v;而本算法可以一次剔除多个变量.

具体的做法是,本算法通过悖论分析,寻找导致 UNSAT 的变量集合 R''.对任意 $v\notin R''$,无论取什么值,都将导致 $SAT_Solve(F')$的结果为 UNSAT.因此,本算法可以一次性剔除掉变量集合$\{v|v\notin R''\}$.显然,这样可以极大地减少运行 $SAT_Solve(F')$的次数,从而极大地减小时间开销.

详细算法描述如下,其中粗体部分为新加入的步骤.

算法 **2**. 基于悖论分析的 BFL 算法.

1. $F''=\neg P_k\wedge\bigwedge_{0\le i<k}\cdot T_i(S_i,W_i,S_{i+1})$
2. foreach $v\in Assign(Free)$
3. $F'=\wedge(Free-\{v\}\Leftarrow Assign(Free-\{v\}))$
4. **if ($SAT\_Solve(F')==UNSAT$)**
5. **$R''=Refutation\_Analysis()$** //悖论分析的结果 R'' 即为导致 UNSAT 的变量集合
6. **$Free=Free\cap R''$** //将所有 $v\notin R''$ 一次性剔除掉
7. $Free$ 即为压缩集

2.2 悖论分析

悖论分析算法的作用在于,抽取导致“*SAT_Solve*(*F*′)==*UNSAT*”的压缩集 *R*″.

对于 SAT 问题 *F*′,设其模型短句集合为 *F*″,其赋值短句集合为 *A*.当求解完成后,冲突短句集合记为 *C*.

若 SAT 问题 *F*′的求解结果是 UNSAT,则必然存在一个决策深度为 0 的冲突短句 *c*.由于此时的决策深度为 0,故不存在被决策(decided)的变量.任意变量的值只可能是被蕴涵的(implicated).

以短句 *c* 为源头,沿着蕴涵关系进行回溯,就可以找到导致“*SAT_Solve*(*F*′)==*UNSAT*”的短句集合 *S*.

S 中的赋值短句集合记为 *S*∩*A*,而其对应的变量集合 $R''=\{v|\{v\}\in S\cap A\}\cup\{v|\{\neg v\}\in S\cap A\}$即为压缩集.

下面,我们将描述如何使用悖论分析算法提取压缩集 *R*″.正确性证明在下一节给出.

算法 **3**. 悖论分析(Refutation_Analysis).

```
1. set     S=∅
2. queue Q=∅
3. foreach l∈c
4.       Q.push(l 的前件短句)
5.       set_visited(l 的前件短句)
6. while (Q.not_empty())
7.       cls=Q.pop()
8.       if (cls 是单位短句)
9.          S=S+{cls}
10.           if (cls 是学习短句) return   R″=Free-{v}
11.      else
12.          foreach l′∈cls
13.              令 ante 为 l′的前件短句
14.              if (¬has_visited(ante))
15.                  Q.push(ante)
16.                  set_visited(ante)
17.  R″={v|{v}∈S∩A}∪{v|{¬v}∈S∩A},即为导致悖论的压缩集.
```

2.3 正确性证明

由文献[13],我们有以下引理:

引理 **1**. 单位短句不参与 Resolution 以构造冲突学习短句.

我们首先证明以下定理:

定理 **1**. *F*″∧*S* 是 *F*′的不可满足短句子集.

证明:首先易知 *F*″∧*S* 是 *F*′的短句子集,因此只需证明 *F*″∧*S* 是不可满足的.

首先假设 *C*′是算法 3 遇到的所有学习短句集合,则易知 *F*″∧*S*∧*C*′是不可满足的.那么我们能否将 *C*′从 *F*″∧*S*∧*C*′中剔除,同时保持其不可满足性呢?

对于任意一个 *cl*∈*C*′,假设 *SRC*(*cl*)是参与冲突学习,以构造 *cl* 的短句集合.由引理 1 可知,*SRC*(*cl*)中不包含单位短句.故必有 *SRC*(*cl*)⊆*F*″.

因此, *F*″∧*S* 和 *F*″∧*S*∧*C*′是等价的.故 *F*″∧*S* 是 *F*′的不可满足短句子集.得证. □

定理 **2**. *F*″∧*R*″⇐*Assign*(*R*″)是 *F*′的不可满足短句子集.

证明:首先易知 *F*″∧*R*″⇐*Assign*(*R*″)是 *F*′的子集,只需证明 *F*″∧*R*″⇐*Assign*(*R*″)是不可满足的.

由算法 3 的步骤 17 易知, *F*″∧*R*″⇐*Assign*(*R*″)等价于 *F*″∧(S∩A).因此,只需证明 *F*″∧(S∩A)是不可满足的.

由算法 2 可知 *F*″∧*S* 不可满足.可将其重写为 *F*″∧(*S*∩*A*)∧(*S*-*A*-*F*″).那么,是否能将(*S*-*A*-*F*″)从中剔除掉,并

保持其不可满足性呢?我们分两方面加以讨论:

1) 如果 S-A-F'' 是空集,则显然 $F''\wedge(S\cap A)$不可满足.
2) 否则,算法 3 将会遇到一个单位学习短句,则算法 3 的步骤 10 将会被运行,此时将仅仅剔除变量 v.易知,此时 $F''\wedge R''\Leftarrow Assign(R'')$不可满足.

定理 2 得证. □

3 复杂性分析

3.1 时间复杂性分析

由于文献[1]并没有给出 BFL 的时间复杂性分析,因此,在分析本文算法的复杂性之前,我们首先需要分析算法 1 中 BFL 的时间复杂性.首先,我们有以下假设:

假设 1. *Free* 包含 M 个变量,其中 N 个是无关变量,而且均匀分布.

基于以上假设,我们有下面的定理:

定理 **3**. BFL 的时间复杂性为 $O(2^N)$.

证明:对于算法 1 中步骤 2 的第 i 次循环,由上述假设可知,共有 $N\times i/M$ 个无关变量已经被剔除.此时,搜索空间的大小为 $2^{N\times i/M}$;算法 1 中步骤 4 的时间复杂度为 $O(2^{N\times i/M})$.因此,整个步骤 2 的时间开销为 $O\left(\sum_{0\le i\le M}2^{N\times i/M}\right)\approx O(2^N)$.得证. □

由于算法 2 需要频繁调用算法 3,因此,我们必须首先分析算法 3 的时间复杂性.我们有下面的定理:

定理 **4**. 算法 3 的时间复杂性为 $O(M^2)$.

证明:由算法 3 和文献[12]中蕴含图分析算法的相似性易知,算法 3 的运算复杂性为 $O(V+E)$,其中 V 和 E 分别是蕴含图中的节点和边的数量.但为了能够和定理 3 作比较,我们必须在 V,E 和 M 之间建立起某种量化关系.

我们首先必须注意到,算法 2 中步骤 3 的 SAT 问题并非一个一般性的 SAT 问题,它有着某种特定的结构.只要我们赋予 *Free* 中所有变量确定的值,我们就能够通过 SAT 求解器中的 BCP 算法得到所有其他变量的赋值.因此,在最恶劣的情况下,该 SAT 问题的蕴含图可以看作一个以$\neg P_k$为根的树,其深度和叶节点的数量均小于 M.因此,$O(V+E)\approx O(M^2)$.得证. □

为了将算法 3 抽取 R'' 的能力考虑在内,我们有以下假设:

假设 2. 算法 3 剔除剩余无关变量的平均比率为 x.

基于定理 4,我们可以得到算法 2 的时间复杂性.

定理 **5**. 算法 2 的时间复杂性为 $O(2^{N*x^N})$.

证明:对于算法 2 中步骤 2 的第 i 次循环,易知运行算法 3 的次数约为 $N\times i/M$.此时,被剔除的无关变量个数约为 $N\times x^{N\times i/M}$.因此,此次运行的搜索空间约为 $O\left(\sum_{0\le i\le M}2^{N\times x^{N\times i/M}}\right)\approx O(2^{N\times x^N})$.与之相比,算法 3 的 $O(M^2)$时间复杂性可以忽略不计. □

比较定理 5 和定理 3,易知算法 2 的时间复杂性远低于算法 1 中的 BFL.

3.2 空间复杂性分析

当前的 SAT 求解器都包含冲突学习机制.而不同的冲突学习机制在空间开销方面存在着巨大差异[12].因此,我们并不直接分析本算法的绝对空间开销,而仅仅分析本算法相对于 BFL 的额外空间开销.

由于算法 2 和算法 1 相比唯一的差别在于算法 2 多运行了一个悖论分析.因此,在空间复杂度上的差别主要在于悖论分析算法.而悖论分析算法的主要空间开销在于集合 S 和队列 Q.

- 我们在每个短句上添加一个标志位,指出该短句是否属于集合 S,因此,集合 S 的开销是线性的;
- 由于算法 3 进行广度优先搜索,因此,Q 中有可能包含冲突短句.但由于 SAT 求解器的冲突学习算法[12]要进行类似的广度优先搜索,因此,Q 的空间开销不大于 SAT 求解器本身的空间开销.

我们将在实验结果中给出 S 和 Q 的峰值尺寸.事实证明,它们的尺寸远小于短句库.

4 增量式 SAT 求解方法

由算法 2 的步骤 3 易知, F'' 生成的模型短句集合独立于步骤 2 的循环.因此,可以在运行步骤 2 的循环之前,利用 zchaff[7]的分组机制,将 F'' 作为一个单独的短句组加入 zchaff 的短句库,从而避免每次运行 SAT_Solve 函数时,重新将模型短句集合加入短句库.

同时,对于步骤 3 中由($Free-\{v\}\Leftarrow Assign(Free-\{v\})$)生成的赋值短句集合,也没有必要在每次运行 SAT_Solve 函数时,重新将他们加入短句库.假设在两个连续的循环体中分别针对 v_1 和 v_2 进行处理,则仅需在完成第 1 次循环后进入第 2 次循环之前,将 v_1 的赋值短句添加入短句库,并将 v_2 的赋值短句删除.

因此,我们将算法 2 进行修改,得到以下的算法 4,其中粗体部分为新加入的步骤.

算法 **4**. 基于悖论分析和增量式求解的 BFL 算法.

1. $F''=\neg P_k\wedge\bigwedge_{0\le i<k}.T_i(S_i,W_i,S_{i+1})$
2. 将 $F''\wedge(Free\Leftarrow Assign(Free))$转换成 **SAT** 问题,添加入短句库
3. foreach $v\in Assign(Free)$
4. 从短句库中删除 v 的赋值短句
5. if (*SAT_Solve*()==*UNSAT*)
6. R'' =*Refutation_Analysis*() //悖论分析的结果 R'' 即为导致 UNSAT 的变量集合
7. *Free*=*Free*$\cap R''$ //将所有 $v'\notin R''$ 一次性剔除掉
8. 对所有 $v'\in Free-R''$, 从短句库中删除 v'的赋值短句
9. **else**
10. 将 v 对应的赋值短句重新添加进短句库
11. *Free* 即为压缩集

在算法 4 的步骤 4 和步骤 8 中,在删除 v 的赋值短句 cls 的同时,是否需要删除由 cls 导致的冲突短句?对于这一问题,由引理 1 易知,单位短句并不参与构造冲突学习短句.故无须删除由 cls 导致的冲突短句.

5 实验结果及分析

5.1 实验对象与方法

我们选取了 ISCAS89 测试集[9]中最复杂的 10 个电路进行反例压缩.我们在 zchaff[7]中同时实现了文献[1]中的 BFL 算法和本文的算法.我们使用 NuSMV[8]产生反例(描述见下一段)和式(2)的 CNF 文件,并分别用本文算法和 BFL 进行反例压缩.其结果见表 1.

Table 1 Experimental results

表 **1** 实验结果

Circuits	Number of variables	Number of clause	Length of counterexample	Number of free variables	BFL of Ref.[1]			Our algorithm			
					Number of irrelevant variables	Reduction rate (%)	Run time	Number of irrelevant variables	Reduction rate (%)	Run time	Speedup
s1512	14 858	39 735	21	667	606	90.85	39.85	606	90.85	3.45	11.55
s1423	16 565	44 248	24	483	400	82.81	292.65	397	82.19	7.12	41.10
s3271	21 769	59 656	15	507	432	85.21	70.16	432	85.21	6.11	11.48
s3384	19 452	50 353	13	743	615	82.77	126.1	615	82.77	9.61	13.12
s3330	6 935	17 322	6	373	295	79.08	19.69	295	79.08	1.77	11.12
s5378	16 180	42 415	10	530	416	78.49	133.27	411	77.55	8.21	16.23
s9234	18 291	49 555	7	362	226	62.43	75.37	226	62.43	10.36	7.28
s13207	107 079	284 839	22	1 352	1 109	82.03	>20 000*	1 093	80.84	153.92	>100
s38584	237 756	661 828	14	1 621	1 069	65.95	>20 000	1 069	65.95	682.19	>10
s38417	211 653	576 324	14	2 029	980	48.29	>20 000	981	48.35	947.84	>10

我们使用以下方法产生深度反例:

* s13207,s38584 和 s38417 的 BFL 运算在 20 000 秒内未能完成,其压缩集和压缩率是用无悖论分析的增量求解方法得到的.

1. 用可达性分析算法产生状态序列$\{S_0,\ldots,S_k\}$,并保证S_k在少于k次影像运算时是不可达的;

2. 将"S_k不可达"作为断言,输入 NuSMV 的限界模型检验算法,得到一个长度为k的反例.

在进行反例压缩的过程中,我们设定运行时间上限为 20 000 秒.

5.2 实验结果

在表 1 中,第 1 列是实验电路名称;第 2 列是 CNF 变量数;第 3 列是 CNF 短句数;第 4 列是反例长度;第 5 列是独立变量个数$|Free|$;第 6 列是被 BFL 算法剔除的无关变量个数;将无关变量个数除以第 5 列的独立变量个数,即为第 7 列的压缩率;第 8 列是 BFL 的运行时间;第 9 列是被本文算法剔除的无关变量个数;压缩率列于第 10 列;而运行时间列于第 11 列;相对于 BFL 的加速比列于最后一列.

在表 2 中,我们给出了反例压缩过程中本文算法的一些统计参数:第 1 列是实验电路名称,其独立变量个数见第 2 列;被放弃的变量个数列于第 3 列.这些变量是由算法 4 的步骤 8 剔除的.UNSAT 次数是指 SAT_Solve 函数不可满足的次数,列于表 2 的第 4 列,该值同时也是运行 Refutation_Analysis 函数的次数;SAT 次数是指 SAT_Solve 函数可满足的次数,列于第 5 列;集合S和队列Q的峰值尺寸分别列于第 6 和第 7 列.上述变量的关系是:

独立变量个数=被放弃的变量数+UNSAT 次数(同时也是分析次数)+SAT 次数.

Table 2 Run time statistics

表 2 运行参数统计

Circuits	Number of free variables	Number of variables being abandon	Number of UNSAT	Number of SAT	Peak size of S	Peak size of Q
s1512	667	601	5	61	3 219	397
s1423	483	369	29	85	8 250	736
s3271	507	416	16	75	5 042	448
s3384	743	596	19	128	7 127	458
s3330	373	278	17	78	2 308	321
s5378	530	387	24	119	4 253	484
s9234	362	173	53	136	6 454	581
s13207	1 352	1 025	68	259	34 108	1 999
s38584	1 621	1 005	64	552	73 971	5 119
s38417	2 029	910	71	1 048	81 855	7 703

由表 1 和表 2 有以下结论:

1. 比较表 1 的第 8 列和第 11 列,可见本算法在绝大多数情况下获得了 1~2 个数量级的加速.
2. 比较表 1 的第 6 列和第 9 列,可见两种算法剔除无关变量的能力都很强,被它们剔除的变量个数相近.
3. 比较表 1 的第 7 列和第 10 列,可见两个算法在大部分情况下剔除了 70%以上的无关变量.
4. 由表 2 的第 3 列可知,绝大部分变量被算法 3 剔除,不再需要运行开销巨大的 SAT_Solve 函数.
5. 由表 2 的第 4 列可知,运行算法 3 的次数很少.
6. 比较表 2 的最后两列与表 1 的第 3 列,可见S和Q远小于短句库.这证明了我们在第 3.2 节中的论断.

6 相关研究工作

由于反例压缩问题可以转化为 SAT 解集的最小化问题(minimal assignment),而后者是 SAT 映象算法的核心,因此,在有关 SAT 映象算法的文章中,很多都讨论了 SAT 解集的最小化问题.

在"全称量词消去法"[5]中,当计算前向映像时,将传输关系、现态和次态转换为 CNF,并用 SAT 求解器求解.而后分析蕴含图,以寻找满足传输关系的次态变量小子集.Kang[6]通过在 SAT 求解器中降低次态变量的决策优先级,使得当传输关系得到满足时,尽可能多的次态变量处于不定态,从而完成类似于文献[5]的功能.Chauhan[4]使用 ATPG 中的敏化和传播方法,判定特定变量是否会影响传输关系的可满足性.然而,这两种方法的压缩效果比文献[5]要差很多.

与上述方法相比,BFL[1]极大地提高了压缩效率,但也带来了巨大的时间开销.本文的目的正是在于降低 BFL 的时间开销.

另一方面,现有的反例压缩算法(BFL 和本文)都局限于处理线性路径反例(linear path-like counterexample),因此,我们在文献[14]中讨论了如何压缩环形反例.

7 结 论

为了有利于理解复杂反例和加快验证算法的速度,必须剔除反例中的无关变量.然而,目前压缩效果最好的 BFL 算法,其时间开销太大.为此,本文提出了一种基于悖论分析和增量式 SAT 求解的反例压缩方法,以高效而快速地剔除反例中的无关变量.理论分析和实验结果表明,该算法能够极大地压缩反例,且相对于现有的方法,能获得 1~2 个数量级的加速.

References:

[1] Ravi K, Somenzi F. Minimal assignments for bounded model checking. In: Jensen K, Podelski A, eds. Proc. of the 10th Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004). LNCS 2988, Heidelberg: Springer-Verlag, 2004. 31–45.

[2] Jin HS, Ravi K, Somenzi F. Fate and free will in error traces. In: Katoen JP, Stevens P, eds. Proc. of the 8th Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002). LNCS 2280, Heidelberg: Springer-Verlag, 2002. 445–459.

[3] Clarke EM, Gupta A, Kukula JH, Shrichman O. SAT based abstraction-refinement using ILP and machine learning techniques. In: Brinksma E, Larsen KG, eds. Proc. of the 14th Int'l Conf. Computer Aided Verification (CAV 2002). LNCS 2404, Heidelberg: Springer-Verlag, 2003. 265–279.

[4] Chauhan P, Clarke E, Kroening D. Using SAT based image computation for reachability analysis. Technical Report, CMU-CX-03-151, Pittsburgh: School of Computer Science, Carnegie Mellon University, 2003.

[5] McMillan KL. Applying SAT methods in unbounded symbolic model checking. In: Brinksma E, Larsen KG, eds. Proc. of the 14th Conf. on Computer Aided Verification (CAV 2002). LNCS 2404, Heidelberg: Springer-Verlag, 2002. 250–264.

[6] Kang HJ, Park IC. SAT-Based unbounded symbolic model checking. In: Proc. of the 40th Design Automation Conf. Anaheim: ACM, 2003. 840–843.

[7] Moskewicz MW, Madigan CF, Zhao Y, Zhang LT, Malik S. Chaff: Engineering an efficient SAT solver. In: Proc. of the 38th Design Automation Conf. Las Vegas: ACM, 2001. 530–535.

[8] Cimatti A, Clarke EM, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A. NuSMV 2: An open source tool for symbolic model checking. In: Brinksma E, Larsen KG, eds. Proc. of the 14th Int'l Conf. on Computer Aided Verification (CAV 2002). LNCS 2404, Heidelberg: Springer-Verlag, 2002. 359–364.

[9] http://www.ee.vt.edu/~ha/cadtools/download.cgi

[10] McMillan KL. Symbolic model checking: An approach to the state explosion problem [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1992.

[11] Biere A, Cimatti A, Clarke EM, Fujita M, Zhu YS. Symbolic model checking using SAT procedures instead of BDDs. In: Proc. of the 36th Conf. on Design Automation (DAC'99). New Orleans: ACM, 1999. 317–320.

[12] Zhang LT, Madigan CF, Moskewicz MH, Malik S. Efficient conflict driven learning in a boolean satisfiability solver. In: Proc. of the Int'l Conf. on Computer Aided Design (ICCAD 2001). San Jose: ACM, 2001. 279–285.

[13] Zhang LT, Malik S. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In: Proc. of the Design and Test in Europe Conf. (DATE 2003). Munich: IEEE Computer Society, 2003. 10880–10885.

[14] Shen SY, Qin Y, Li SK. Minimizing counterexample of ACTL property. In: Borrione D, Paul WJ, eds. Proc. of the 13th Advanced Research Working Conf. on Correct Hardware Design and Verification Methods (CHARME 2005). LNCS 3725, Heidelberg: Springer-Verlag, 2005. 393–397.

沈胜宇(1975－),男,广西南宁人,博士,助理研究员,主要研究领域为电子 CAD,嵌入式微处理器设计与验证方法.

李思昆(1942－),男,教授,博士生导师,CCF 高级会员,主要研究领域为电子设计自动化与 SoC 设计方法学,虚拟样机与分布式虚拟环境.