doi:10.11887/j.cn.201501004 http://journal.nudt.edu.cn

求解布尔不可满足子式的消解悖论算法*

张建民[1],黎铁军[1],徐炜遐[2],庞征斌[2],李思昆[3]
(1.国防科技大学 计算机学院,湖南 长沙 410073;
2.国防科技大学 并行与分布处理重点实验室,湖南 长沙 410073;
3.国防科技大学 高性能计算国家重点实验室,湖南 长沙 410073)

摘 要:求解布尔不可满足子式在超大规模集成电路设计与验证领域都具有非常重要的理论与应用价值,帮助EDA工具迅速定位错误与不一致。针对求解不可满足子式的非完全方法,提出了消解悖论与悖论解析树的概念,在此基础上提出一种启发式局部搜索算法。该算法根据公式的消解规则,采用局部搜索过程直接构造证明不可满足性的悖论解析树,而后递归搜索得到不可满足子式;算法中融合了布尔推理技术、动态剪枝方法及蕴含消除方法以提高搜索效率。基于随机测试集进行了实验对比,结果表明提出的算法优于同类算法。

关键词:形式验证;布尔可满足问题;不可满足子式;消解悖论;局部搜索
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2015)01-021-07

A resolution-based Boolean unsatisfiable subformulas computing algorithm

ZHANG Jianmin[1], *LI Tiejun*[1], *XU Weixia*[2], *PANG Zhengbin*[2], *LI Sikun*[3]
(1. College of Computer, National University of Defense Technology, Changsha 410073, China;
2. National Key Laboratory of Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China;
3. State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China)

Abstract: Computing unsatisfiable subformulas of Boolean formulas has practical applications in VLSI design and verification. The unsatisfiable subformulas can help electronic design automation tools to rapidly locate the errors and inconsistency. The definitions of resolution refutation and refutation parsing tree, and a heuristic local search algorithm to extract unsatisfiable subformulas from the resolution refutation of a formula are presented. The approach directly constructs the refutation parsing tree for proving unsatisfiability with a local search procedure, and then recursively derives unsatisfiable subformulas. The algorithm combines with reasoning heuristics, dynamic pruning and subsumption elimination method to improve the efficiency. The experimental results show that our algorithm outperforms the similar algorithms on the random benchmarks.

Key words: formal verification; Boolean satisfaction; unsatisfiable subformula; resolution refutation; local search

在诸如电子设计自动化与超大规模集成电路(Very Large Scale Integration,VLSI)形式化验证等领域中,很多设计与验证问题都可以转换为布尔可满足问题来解决;如果公式不可满足,通常表示软硬件设计出现错误与不一致,那么需要缩小错误可能出现的范围,以便快速定位错误,即移除与公式不可满足无关的短句,查找导致公式中变元赋值冲突的最小集合,即提取原始公式的不可满足子式,因此求解不可满足子式在很多领域都具有非常重要的理论与应用价值。近几年来不可满足子式的求解方法逐渐成为研究热点。布尔不可满足子式的典型应用包括现场可编程逻辑门阵列(Field-Programmable Gate Array,FPGA)的布线方法[1]、错误诊断与定位[2]、电路综合[3]等。

求解不可满足子式的方法按照问题的解决方式能够划分为完全算法与非完全算法。通常来说,完全算法能够找到问题的精确解,但其面临的主要挑战是计算时间随问题规模的增大呈指数级增长。而非完全算法尽管有时无法给出精确解,但其运算速度快,求解效率高。随着基于Davis-Putnam-Logemann-Loveland(DPLL)回溯搜索过程的布尔可满足性问题(Boolean Satisfiability,

* **收稿日期**:2014-06-10
基金项目:国家自然科学基金资助项目(61103083,61133007);国家863计划资助项目(2012AA01A301);国家973计划资助项目(2011CB309705)
作者简介:张建民(1979—),男,山西平遥人,助理研究员,博士,E-mail:jmzhang@nudt.edu.cn

SAT)求解技术的不断发展,因此求解布尔不可满足子式的算法大多是基于 DPLL SAT 求解器的完全方法[4-10]。

著名的十个可满足问题的经典挑战[11]中第五个是:(在5~10年内)设计一种证明公式不可满足性的局部搜索算法。然而这个问题10年来无人解决。直到10年后 Prestwich 等[12]与 Audemard 等[13]才首次提出采用局部搜索过程来证明公式不可满足性的算法。但是他们的算法仅能证明公式是否不可满足,而不能求解公式的不可满足子式。直到目前为止,还没有公开发表的研究采用局部搜索方法从公式不可满足性的证明中提取布尔不可满足子式。

1　消解悖论算法的理论基础

合取范式公式的构造规则是:文字是变元及其否定形式,而若干个文字的析取构成短句,若干个短句的合取组成公式。

定义1(布尔可满足问题)　给出一个 CNF 公式 $\varphi:\varphi=\bigwedge_{i=1}^{n}C_i$,其中短句 $C_i=\bigvee_j x_j\vee\bigvee_k\neg x_k$,$V=\{x\mid x\in\varphi\}$表示变元集合。可满足问题是指给定一个赋值 M,M 表示从 φ 的变元集合 V 到真值{true, false}的映射:$V\to\{\text{true, false}\}^{|V|}$。如果存在一个赋值 M,使得 $\varphi=\text{true}$,即 $M\models\varphi$,那么称公式 φ 是可满足的;若对于任意的赋值 M,都有 $M\not\models\varphi$,则称公式 φ 是不可满足的。

定义2(不可满足子式)　给出一个公式 φ,ψ 是公式 φ 的一个不可满足子式当且仅当 ψ 是不可满足的,并且 $\psi\subseteq\varphi$。

定义3(消解)　假设 C_i 与 C_j 为两个短句,若 $l_i\in C_i$ 与 $l_j\in C_j$ 是一对互补的文字,则 $(C_i\backslash l_i)\vee(C_j\backslash l_j)$ 称为 C_i 与 C_j 的消解式,其中 l_i 和 l_j 称为消解基,C_i 和 C_j 称为消解母式。每次应用消解规则产生消解式叫做一个消解步骤。

引理1[14]　若短句 C 为 C_i 与 C_j 的消解式,则 $(C_i,C_j)\models C$。

引理2[14]　若 $C_i=l_i$ 与 $C_j=l_j$ 为两个单元短句,并且 l_i 和 l_j 是一对互补的文字,则 C_i 与 C_j 的消解式为空短句,即 $(C_i,C_j)\models\perp$。

定义4(消解序列)　假设 S 为短句集,且 C 为短句。若存在短句的有穷序列 $C_0, C_1,\cdots, C_n$ 满足:

1) $C_n=C$;

2)令 $0\leqslant i\leqslant n$,则短句 C_i 至少满足下列两个条件之一:

(ⅰ) $C_i\in S$;

(ⅱ) $\exists j, k, 0\leqslant j,k\leqslant i$,使得 $(C_j,C_k)\models C_i$;

那么短句 C 称为 S 的消解结果,表示为 $S|\!-\!|\!-C$,并将 $C_0, C_1,\cdots, C_n$ 称为由 S 导出 C 的消解序列。

引理3[14]　设 S 为短句集,且 C 为短句,若 $S|\!-\!|\!-C$,则 $S\models C$。

引理4[14](消解原理)　短句集 S 为不可满足的当且仅当 $S|\!-\!|\!-\perp$。

定义5(消解悖论)　给定一个不可满足公式 φ,$N=\{C_0,C_1,\cdots,C_n\}$ 为 φ 的一个有穷消解序列。若 N 的最终消解式为空短句,即 $C_n=\perp$,构造一个由消解步骤构成的集合 $R=\{P_i\mid P_i$ 为 $(C_j,C_k)\models C_l$,其中 $C_j, C_k, C_l\in N\}$,则 R 称为 φ 的一个消解悖论。

定义6(悖论长度)　R 所包含的消解步骤的数目称为悖论长度,表示为 $|R|$。

显然地,消解悖论是一类特殊的消解序列,由于其最终消解式为空短句,因此包含布尔公式不可满足的原因。那么当通过搜索过程得到一个不可满足公式的消解悖论时,如何从中提取出布尔不可满足子式呢?

定理1　给定一个 CNF 公式 φ,φ 是不可满足当且仅当 φ 至少包含一个消解悖论 R。若令集合 $D=Cla(\varphi)\cap Cla(R)$,其中 $Cla(\varphi)=\{C\mid C$ 为 φ 中短句$\}$,$Cla(R)=\{C\mid C$ 为 R 中短句$\}$,那么 $\psi=\wedge_{C\in D}C\models\perp$ 成立,且 ψ 是 φ 的不可满足子式。

证明:首先证明第一个结论。必要性是显然成立的,若 φ 包含一个消解悖论 R,根据定义5,R 的最终消解步骤的消解式为空短句,因此 φ 是不可满足的。

下面证明充分性。$Cla(\varphi)$ 表示公式 φ 中所有短句构成的集合,则 $Cla(\varphi)$ 为一个有限短句集。根据引理4的消解原理,若 φ 是不可满足的,表示为 $Cla(\varphi)|\!-\!|\!-\perp$。根据定义5,将短句集导出 $\perp$ 的所有消解步骤析出,就可以构成一个消解悖论 R,因此结论成立。

下面证明第二个结论。

令 $D=Cla(\varphi)\cap Cla(R)$,表示消解悖论 R 中属于原始公式 φ 的短句,构成一个短句集 D。

根据定义5,可以从消解悖论 R 中提取出一个短句序列 $N=\{C_0, C_1,\cdots, C_n\}$,使得 $C_n=\perp$,且 $\forall C_i\in N$,则要么 $C_i\in D$,要么 $(C_j,C_k)\models C_i$,其中 $C_j,C_k\in N$。

根据定义4,短句集 $D|\!-\!|\!-C_n=\perp$。根据引理4,D 为不可满足的,所以 $\psi=\wedge_{C\in D}C\models\perp$ 成立。

并且 $\psi \subseteq \varphi$,那么 ψ 是 φ 的不可满足子式。

综上所述,命题结论成立。 □

定义 7(悖论解析树)　若一个有向树 $T(V, E, r)$ 满足下面的条件:

(1)包含唯一的根结点,对应的短句 $C_r = \perp$;

(2)$\forall v \in V \backslash L$,其中 L 表示 T 中叶结点的集合,假设 v 及其父结点 $p \in V$ 与 $q \in V$ 分别对应的短句为 C_v,C_p 与 C_q,那么 $C_p \wedge C_q | = C_v$;而边 $e_{pv} \in E$ 与 $e_{qv} \in E$ 分别表示从父结点 p 与 q 指向子结点 v。

则称 $T(V, E, r)$ 为悖论解析树。从根结点到所有叶结点的边数中的最大值称为解析树深度,记为 $|T|$。

在公式 φ 的不可满足性的证明过程中得到了一个消解悖论 R,是否一定能够构造一个符合定义 7 的悖论解析树 $T(V, E, r)$? 反之,公式 φ 的一个悖论解析树 $T(V, E, r)$ 能否导出 φ 的一个消解悖论 R?

定理 2　给定不可满足的公式 φ,存在一个 φ 的消解悖论 R,当且仅当存在一个对应于 R 的悖论解析树 $T(V, E, r)$。

证明:首先证明结论的充分性。假设存在一个公式 φ 的消解悖论 R,那么对 R 的悖论长度 $|R|$ 采用数学归纳法证明。

当 $|R| = 1$ 时,消解悖论 R 只包含 1 个消解步骤,可以表示为 $(x, \neg x) | = \perp$,其中 x 表示命题变元。以空短句 $\perp$ 为根结点 r,$V = \{(x), (\neg x)\}$ 为 r 的直接父结点,边 $E = \{e_1, e_2\}$ 分别由父结点指向根结点,根据定义 7,构成悖论解析树 T,因此结论成立。

假设 $|R| \leqslant m$,结论成立。

当 $|R| = m + 1$ 时,R 产生空短句的最后一个消解步骤表示为 $(C_i, C_j) | = \perp$,根据定义 7,以 $\perp$ 为根结点,构造悖论解析树,此时存在以下两种情况。

第一种情况:产生空短句 $\perp$ 的两个消解母式中只有一个是原始公式中的短句,这里假设 $C_j \in \varphi$,$C_i \notin \varphi$。由于 $|R| = m + 1$,因此 C_i 是由 m 个消解步骤产生的,符合假设条件,则存在一个以 C_i 为根结点的解析树 $T_i(V_i, E_i, r_i)$。那么,令 $V = V_i \cup \{C_j, \perp\}$,$E = E_i \cup \{e_i, e_j\}$,$r = \perp$,其中边 e_i 与 e_j 分别由 C_i 和 C_j 指向根结点 $\perp$,从而构造一个树 $T(V, E, r)$,满足定义 7,因此 $T(V, E, r)$ 为 φ 的一个悖论解析树。

第二种情况:$C_i \notin \varphi$,且 $C_j \notin \varphi$,即产生空短句 $\perp$ 的两个消解母式都是中间结果。假设产生 C_i 与 C_j 的消解序列分别为 S_i 和 S_j,其包含的消解步骤的数目记为 $|S_i|$ 与 $|S_j|$,因为 $|R| = m + 1$,所以 $|S_i| \leqslant m - 1$,$|S_j| \leqslant m - 1$;满足假设条件,则表示存在分别以 C_i 与 C_j 为根结点的解析树 $T_i(V_i, E_i, r_i)$ 和 $T_j(V_j, E_j, r_j)$。那么,令 $V = V_i \cup V_j \cup \{\perp\}$,$E = E_i \cup E_j \cup \{e_i, e_j\}$,$r = \perp$,其中边 e_i 与 e_j 分别由 C_i 和 C_j 指向根结点,从而构造一个树 $T(V, E, r)$,符合定义 7,因此 $T(V, E, r)$ 为 φ 的一个悖论解析树。

所以,若存在消解悖论 R,则必定存在一个对应于 R 的悖论解析树 $T(V, E, r)$。

下面证明必要性。

假设存在一个悖论解析树 $T(V, E, r)$,那么针对解析树深度 $|T|$ 采用数学归纳法进行证明。

当 $|T| = 1$ 时,即深度为 1 的悖论解析树,根据定义 7,T 的根结点对应的短句为 $\perp$,假设其父结点对应的短句分别为 C_1 与 C_2,对应 T 的消解悖论 R 由一个消解步骤构成:$(C_1, C_2) | = \perp$。

假设 $|T| = m$,命题结论成立。

当 $|T| = m + 1$ 时,那么将 T 的所有叶结点除去,将根结点 $\perp$ 与其他分支结点重新组成一个树 T';符合定义 7,T' 为悖论解析树,并且 $|T'| = m$,根据假设条件,T' 对应一个消解悖论,记为 R'。而 T' 的叶结点是由 T 的叶结点通过消解得到的,那么将这些消解步骤加入到 R' 中,就构成了新的消解悖论 R。

所以,若存在一个悖论解析树 $T(V, E, r)$,则必定存在一个对应于 T 的消解悖论 R。

综合上面充分性与必要性的证明,得到结论:当且仅当存在一个对应于 R 的悖论解析树 $T(V, E, r)$ 时,存在一个 φ 的消解悖论 R。 □

上面定性地分析了消解悖论 R 与悖论解析树 T 的对应关系。定理 3 给出了消解悖论与悖论解析树之间的定量关系。

定理 3　给定一个不可满足的公式 φ,R 为 φ 的消解悖论,若悖论长度 $|R| = n$,则存在一个悖论解析树 $T(V, E, r)$,使得 $|T| \leqslant n$。

证明:对 R 的悖论长度 $|R|$,采用数学归纳法证明。当 $|R| = 1$ 时,消解悖论 R 只包含一个消解步骤,表示为 $(C_1, C_2) | = \perp$,那么以 $\perp$ 为根结点,以短句 C_1 与 C_2 为其父结点,构成深度为 1 的解析树,即 $|T| = 1$,结论成立。

假设 $|R| \leqslant m$,命题结论成立。

当 $|R| = m + 1$ 时,假设 R 中产生空短句的最后一个消解步骤为 $(C_i, C_j) | = \perp$,那么根据其父结点 C_i 与 C_j 的不同来源,可以分为下面两种

情况。

第一种情况：C_i与 C_j的其中一个是原始公式的短句，这里假设 $C_i \in \varphi$。因为 $|R| = m+1$，那么产生 C_j的消解步骤为 m，符合假设条件，因此产生 C_j的部分树的深度为 $|T_j| \leqslant m$，则悖论解析树 T 的深度为 $|T| \leqslant m+1$；

第二种情况：$C_i \notin \varphi$，且 $C_j \notin \varphi$，表明短句 C_i与 C_j都是在公式 φ 消解过程中产生的中间结果。这里假设产生 C_i与 C_j的消解序列分别为 S_i和 S_j，其包含的消解步骤的数目记为 $|S_i|$与 $|S_j|$。由于 $|R| = m+1$，因此 $|S_i| \leqslant m-1$，$|S_j| \leqslant m-1$，符合假设条件，将产生 C_i与 C_j的部分树的深度记为 $|T_i|$和 $|T_j|$，那么 $|T_i| \leqslant m-1$，$|T_j| \leqslant m-1$，所以 R 所对应的悖论解析树 T 的深度为 $|T| \leqslant m+1$。

综上所证，当消解悖论长度 $|R| = n$ 时，则存在一个悖论解析树 $T(V, E, r)$，使得 $|T| \leqslant n$。□

给出不可满足公式 φ 的一个消解悖论 R，那么对应于 R 的悖论解析树 $T(V, E, r)$ 的构造规则为：空短句⊥对应根结点 r；假设 R 中的任意一个消解步骤为 $(C_1, C_2) = C$，那么 C，C_1与 C_2分别对应 T 中的结点 $\{v, v_1, v_2\} \subseteq V$，而边 $\{e_1, e_2\} \subseteq E$ 分别由结点 v_1与 v_2指向 v。通过该规则，能够将消解悖论从最终的消解式⊥开始，一直回溯到原始公式中的短句，构造出悖论解析树。根据公式 φ 的悖论解析树 $T(V,E,r)$，如何求解 φ 的不可满足子式？

定理 4 给定不可满足公式 φ 的一个消解悖论 R，$T(V,E,r)$ 为 R 对应的悖论解析树，那么 $T(V,E,r)$ 中所有叶结点对应的短句，构成 φ 的一个不可满足子式。

证明：首先证明悖论解析树 T 中的叶结点对应的短句都属于原始公式 φ。

用反证法。假设存在一个叶结点 v，对应的短句 $C \notin \varphi$。那么根据定义 5，φ 中必定存在两个短句 C_i与 C_j，满足 $(C_i, C_j) = C$。根据定义 7，则 T 中存在两个结点 v_1与 v_2分别对应于 C_1与 C_2，因此短句 C 对应的结点 v 不是叶结点，产生矛盾，假设错误。

所以，$T(V,E,r)$ 中的叶结点对应的短句都属于原始公式 φ，并且 T 中叶结点对应的短句都属于 R，那么根据定理 1，可得结论成立。□

2 消解悖论算法概述

通常来说，局部搜索算法具有搜索效率高，运算速度快的优点，尤其针对某些类型的问题，例如随机 2 - SAT、3 - SAT 问题，它是很有效的方法。因此，采用局部搜索方法作为解空间的搜索策略，基于定理 1 与定理 4 的结论，提出了一种从证明公式不可满足性的悖论序列中提取不可满足子式的局部搜索算法。算法 1 给出了消解悖论算法 R^2bLSA 的伪代码。

算法 1 基于消解悖论的算法

Alg. 1 Unsatisfiable subformulas extraction algorithm based on refutation resolution

输入： CNF 公式 *formula*

输出： 不可满足子式的短句集合 *SmallUS*

```
R2bLSAtoExtractUS (formula)
1   refuted = false
2   iteration = 0
3   sequence = ∅
4   while ((iteration<MAXITER) and !refuted) do
5     if (Unit_Clause_Propagation()返回UNSAT)
6       refuted = true
7       sequence= sequence∪{消解的短句}
8     else if (存在二元短句) then
9       Binary_Clause_Resolution()
10      Non_Tautology()
11      Equality_Reduction()
12      No_Same_Clause()
13      sequence= sequence∪{消解的短句}
14    else
15      随机地选择两个短句进行消解
16      sequence= sequence∪{消解的短句}
17    Subsumption_Elimination(formula,resolvent)
18    Trace_Updating(resolvent)
19    if (formula.size > MAXSIZE) then
20      从formula中随机地删除短句C
21      Trace_Pruning(C)
22    iteration++
23  if (refuted == true) then
24    print "unsatisfiable"
25    SmallUS = Compute_US(sequence)
26  else
27    print "unresolved"
28  return SmallUS
```

该算法的输入是 CNF 公式，目标函数是消解式为更短的短句或空短句。算法启发式或随机地选择两个短句进行消解，直到产生一个消解悖论或循环次数达到上限。在消解过程中，算法将每一次应用消解规则的消解步骤都保存到 sequence 中，如果公式是不可满足的，最终必会得到空短句。而后根据不可满足性的证明序列 sequence，可以很自然地构造一个树结构，同时利用高效的树搜索算法，也就是从 sequence 所包含一系列的消解母式中，根据定理 1 和定理 4，快速搜索悖论解析树的所有叶节点，从而得到原始公式的不可满足子式。

消解悖论算法首先通过单元短句传播函数 Unit_Clause_Propagation 判定公式是否不可满足，这是因为公式不可满足的充要条件是两个单元短句消解出空短句，该函数又称为纯文字处理过程。

如果当前的公式包含二元短句,算法采用一些布尔推理技术,例如二元短句消解函数 Binary_Clause_ Resolution 与等价约简函数 Equality_Reduction,启发式地选择短句进行消解。其中消除重言式函数 Non_Tautology 的作用是删除包含极性相反文字的短句;而 No_Same_Clause 函数剔除公式中重复的短句。若公式中不存在二元短句,则随机选择两个包含极性相反文字的短句进行消解,两个短句中相同的文字越多,那么被选择的概率就越高。

当产生的消解式加入到公式中时,蕴含消除过程 Subsumption_Elimination 用来删除公式中因存在蕴含关系而导致的冗余短句。但是随着公式与消解序列的不断增大,将会降低搜索效率,因此提出一种悖论剪枝方法来避免出现内存溢出的问题,通过 Trace_Pruning 函数与Trace_Updating函数实现。当公式大于某个预设的常数时,随机选择一个短句从公式中删除,其中较长的短句会以较大的概率被选中,与此同时,将该短句在证明序列中的一些冗余的源短句及其消解步骤删除。

在局部搜索过程中,算法记录产生空短句的消解步骤,存入集合 sequence 中。而后根据证明序列 sequence,构造一个悖论解析树,并通过递归函数 Compute_US 提取不可满足子式。从根节点的空短句开始回溯,找到其父节点,若其父节点是叶节点,根据定理 4,将其加入不可满足子式;若其父节点不为叶节点,则以该节点为根结点,与其祖先结点又构成一个子树,运用递归过程不断迭代,直到每个分支都停止于叶结点,这时就生成一个不可满足子式。

3　蕴含消除方法

当产生的消解式加入到公式中时,蕴含消除方法将删除公式中冗余的短句。下面首先给出 CNF 公式的蕴含关系与自蕴含关系的定义:

定义 8(蕴含关系)　给定 CNF 公式 φ 的两个短句 C_i 与 C_j,$L(C_i)$ 与 $L(C_j)$ 分别表示 C_i 和 C_j 中包含的文字集合,若 $L(C_i)\subseteq L(C_j)$,则称 C_i 蕴含 C_j,或称 C_j 蕴含于 C_i。

定义 9(自蕴含关系)　给定 CNF 公式 φ 的两个短句 C_i 与 C_j,若 $(C_i, C_j)\mid = C$,且消解式 C 蕴含 C_i 或 C_j,则称 C_i 与 C_j 满足自蕴含关系。

如果公式中存在蕴含关系,那么被蕴含的短句对于公式不可满足性的证明是冗余的,可以被删除。对于自蕴含关系而言,将消解式加入到公式中后,能够将被蕴含的消解短句从公式中移除,例如,给出短句 $C_1 = (x_1 \vee x_2 \vee x_3)$ 与 $C_2 = (\neg x_1 \vee x_2)$,那么 C_1 与 C_2 基于消解元 x_1 的消解式为 $C = (x_2 \vee x_3)$,显然 C 蕴含 C_1,因此将消解式 C 替换公式中冗余的短句 C_1。从公式移除这些被蕴含的短句,既不影响公式的消解结果,也不改变公式的可满足性;并且这些短句会占用存储空间,降低求解效率。因此,消解悖论算法中引入了蕴含与自蕴含消除过程。蕴含与自蕴含消除过程的伪代码如算法 2 所示。

算法 2　蕴含消除过程

Alg. 2　Subsumption elimination procedure

输入: 公式 *formula* 与消解式 *resolvent*
输出: 新的公式 *formula*

```
Subsumption_Elimination(formula, resolvent)
1   if (d(C_i,C_j)=1) then
2       if (V(C_i) ⊆ V(C_j)) then
3           从 formula 中移除短句 C_j
4       else if (V(C_j) ⊆ V(C_i)) then
5           从 formula 中删除短句 C_i
6   for (each clause C ∈ formula) do
7       if (L(C) ⊆ L(resolvent)) then
8           return formula
9       else if (L(resolvent) ⊆ L(C)) then
10          从公式 formula 中移除短句 C
11  将 resolvent 加入到 formula 中
12  return formula
```

算法 2 中 $V(C)$ 表示短句 C 中命题变元的集合。

定义 10(短句距离)　给定两个短句 C_i 与 C_j,那么 C_i 与 C_j 中包含互补文字对的数目,称为 C_i 与 C_j 的短句距离,记为 $d(C_i,C_j)$。

当 $d(C_i,C_j)\geqslant 1$,即包含至少一对互补文字时,C_i 与 C_j 可以进行消解。根据自蕴含关系以及短句距离的定义,可以得到下面的结论:

定理 5　假设包含互补文字的两个消解短句 C_i 与 C_j,$(C_i,C_j)\mid = C$。若其中一个短句的命题变元集合是另外一个的子集,且 $d(C_i,C_j)=1$,这里假设 $V(C_i)\subseteq V(C_j)$,那么消解式 C 必定蕴含短句 C_j,即 C_i 与 C_j 满足自蕴含关系。

证明:由于 $(C_i,C_j)\mid = C$,假设 l_i 与 l_j 是它们的消解基,并且 $l_i\in C_i$, $l_j\in C_j$, 那么 $L(C) = L(C_i\backslash l_i) \cup L(C_j\backslash l_j)$。

由于 $d(C_i,C_j)=1$,即 C_i 与 C_j 除了 l_i 与 l_j 一对互补文字之外,不再包含其他的互补文字对,即不存在这样的文字:$l\in C_i$,并且 $l\neq l_i$,但 $\neg l\in C_j$。

并且根据假设,$V(C_i)\subseteq V(C_j)$, 则 $L(C_i\backslash l_i)\subseteq L(C_j\backslash l_j)$,得到 $L(C) = L(C_j\backslash l_j)$,即 $L(C)\subset L(C_j)$。根据定义 8,消解式 C 蕴含短句 C_j;而根据定义 9,C_i 与 C_j 满足自蕴含关系。结论成立。　□

根据这个结论,算法 2 中的第 1 ~ 5 行以更加简单的方式判定自蕴含关系,而后从公式中删除被 resolvent 蕴含的消解母式。第 6 ~ 11 行消除公式中存在的蕴含关系,又分为前向蕴含消除与后向蕴含消除。如果公式中存在某个短句 C,C 蕴含消解式 resolvent,那么就直接将消解式丢弃,该过程称为前向蕴含消除,由第 7 ~ 8 行代码实现。所谓后向蕴含消除,是指若公式中存在某个短句 C,消解式 resolvent 蕴含 C,则将短句 C 删除,该过程由代码第 9 ~ 10 行实现。通过实验表明,自蕴含以及前向与后向蕴含消除能够有效地减少存储空间消耗,从而加快消解过程。

4 悖论剪枝方法

在算法搜索的过程中,由于产生的消解式不断地加入公式中,并且要保存消解悖论序列,因此占用的存储空间会越来越大,同时会降低搜索的效率。为了避免存储空间的不断增长乃至溢出,提出了一种悖论剪枝方法。在短句进行消解时,为每个消解式 resolvent 构造 2 个域值:一个是产生该消解式的源短句序列(resolvent. trace),另一个是当前消解式的后代短句的数目(resolvent. offspring_count)。悖论剪枝方法包含两个过程:Trace_Updating 函数为每个加入公式的消解式初始化并维护其 2 个域值;当一个短句从公式中删除时,Trace_Pruning 函数删除与该短句相关但与证明不可满足性无关的消解步骤。悖论剪枝方法伪代码如算法 3 所示。

算法 3 悖论剪枝过程

Alg. 3 Refutation pruning procedure

输入:消解式 *resolvent*;随机选择的短句 C
输出:消解悖论的短句队列 *sequence*

```
Trace_Updating(resolvent)
1    resolvent.trace = parent_clauses.trace
2    resolvent.offspring_count = 0
3    for (resolvent.trace 中的每个短句 C1) do
4        C1.offspring_count++
Trace_Pruning(C)
1    if ((C.offspring_count == 0) and (C.trace != ∅)) then
2        for (C.trace 中的每个短句 C1) do
3            C1.offspring_count--
4            Trace_Pruning(C1)
5        从 sequence 中删除关于 C 的消解步骤
6        释放 C.trace 空间
7    return sequence
```

函数 Trace_Updating 的过程为:当产生一个消解式 resolvent 时,其后代短句的数目初始化为 0;对于 resolvent 的源短句序列中的每个短句 C_1,将 C_1 的 offspring_count 域值都加 1。Trace_Pruning 函数的具体流程为:当从公式中删除一个短句 C 时,如果 C 还有后代短句,那么其后代短句可能还参与空短句的消解过程;否则,表明 C 已经不可能属于最终的消解悖论,因此删除 C 的消解步骤及其 trace 域,并且对于 C 的源序列中的每个短句 C_1,将 C_1 的 offspring_count 域都减 1;但是短句 C_1 的后代数目也可能会变为 0,所以该步骤是一个递归过程,不断迭代,直到把所有冗余的短句及其消解步骤都删除。通过实验表明,悖论剪枝方法能够有效地削减证明序列,从而减少存储空间与运行时间的消耗。

5 实验结果与分析

3-SAT 问题是 SAT 测试集中一类典型的问题,并且其他类型的 SAT 问题都可以转换为 3-SAT问题求解,因此基于随机 3-SAT 和 2-SAT 测试集,将 R^{2}bLSA 算法与 AMUSE 算法[4]进行了实验对比。AMUSE 算法是公认的能够高效地求解不可满足子式的一种算法。R^{2}bLSA 算法采用 C ++ 与 STL 实现。算法输入都是 DIMACS CNF 格式的公式,运行的时限设置为 3600s。实验环境是 1. 6GHz 的 Athlon * 2 CPU,内存 1GB,操作系统为 RHEL Linux 的机器。

采用经典的 k-SAT 公式生成方法产生测试集,其输入参数包括变元数 N,短句数 C,以及短句所包含的文字数 k。构造 *k-SAT* 公式的过程为:从 N 个变元中随机的选取 m 个变元,其中以概率 p,$m = k$;以概率 $1 - p$,$1 \leqslant m \leqslant k$;而后以概率 q 构成正文字或负文字,从而组成 m 元短句。图 1 给出了 R^{2}bLSA 算法与 AMUSE 算法在随机 2-SAT 测试集上的实验结果。图 2 是 R^{2}bLSA 算法与 AMUSE 算法基于随机 3 - SAT 测试集上的实验结果。实验时的参数设置为:$N = 200$,概率 $p = q = 0.5$,公式所包含的短句数从 100 到 2000,每组测试集递增 100 个短句。

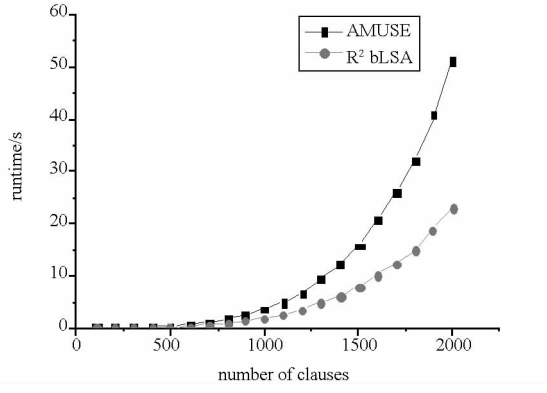


图 1 算法基于 2-SAT 测试集的实验结果

Fig. 1 Experimental results of two algorithms on 2-SAT benchmarks

从图 1 和图 2 的实验结果可以看出,R^{2}bLSA 算法优于 AMUSE 算法,尤其是随机 2-SAT 测试

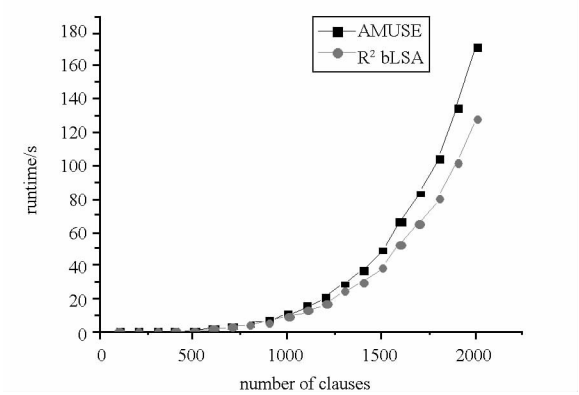


图 2　算法基于 3-SAT 测试集的实验结果

Fig. 2　Experimental results of two algorithms on 3-SAT benchmarks

集;并且随着公式所包含短句数增加,R^2bLSA 算法的性能优势更加明显。其主要原因包括以下三点:第一,不可满足子式的求解过程与公式可满足性判定过程结合得非常紧密,在进行公式的可满足性判定时,就以优化的数据结构保存了求解不可满足子式所需要的信息,因此提取不可满足子式的过程就变得非常高效;第二,R^2bLSA 算法的可满足性检测过程实现简单,单位时间内执行的循环次数更多;第三,R^2bLSA 算法中包括很多针对长度较小的短句,尤其是二元短句的优化技术,而随机 2-SAT 与 3-SAT 测试集中包含很多单元短句与二元短句。

6　结论

本文针对布尔不可满足子式的求解问题,提出了一种基于消解悖论的局部搜索算法,并在算法中融入了众多启发式方法,包括蕴含消除与悖论剪枝技术。基于随机测试集进行了实验,结果表明消解悖论算法优于 AMUSE 算法。通过实验说明,算法对于短句长度较小但数量众多的公式,效果较好。下一步的工作是针对如何提高包含更多文字的短句的消解效率,在算法中融入更加高效的推理技术。

参考文献(References)

[1] Nam G J, Aloul F, Sakallah K, et al. A comparative study of two Boolean formulations of FPGA detailed routing constraints[C]// Proceedings of the 2001 International Symposium on Physical Design, New York, 2001: 222 – 227.

[2] Stuckey P J, Sulzmann M, Wazny J. Improving type error diagnosis [C]//Proceedings of the 2004 ACM SIGPLAN Workshop on Haskell, New York, 2004: 80 – 91.

[3] Shen S Y, Qin Y, Wang K F, et al. Synthesizing complementary circuits automatically [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(8): 1191 – 1202.

[4] Oh Y, Mneimneh M N, Andraus S, et al. AMUSE: a minimally-unsatisfiable subformula extractor[C]//Proceedings of the 41st Design Automation Conference, New York, 2004: 518 – 523.

[5] 赵相福, 欧阳丹彤. 使用 SAT 求解器产生所有极小冲突部件集[J]. 电子学报, 2009, 37(4): 804 – 810.
ZHAO Xiangfu, OUYANG Dantong. Deriving all minimal conflict sets using satisfiability algorithms[J]. Acta Electronica Sinica, 2009,37(4): 804 – 810. (in Chinese)

[6] 张建民, 沈胜宇, 李思昆. 最小布尔不可满足子式的求解算法[J]. 电子学报, 2009, 37(5): 993 – 999.
ZHANG Jianmin, SHEN Shengyu, LI Sikun. Algorithms for deriving minimum unsatisfiable Boolean subformulae[J]. Acta Electronica Sinica, 2009, 37(5): 993 – 999. (in Chinese)

[7] Pitte C, Hamadi Y, Sais L. Efficient combination of decision procedures for MUS computation [C]//Proceedings of 7th International Symposium on Combining Systems, 2009: 335 – 349.

[8] Ryvchin V, Strichman O. Faster extraction of high-level minimal unsatisfiable cores [C]//Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing, 2011: 174 – 187.

[9] Belov A, Lynce I, Marques-Silva J. Towards efficient MUS extraction[J]. Journal AI Communications, 2012, 25(2): 97 – 116.

[10] Liffiton M H, Malik A. Enumerating infeasibility: finding multiple MUSes quickly [C]//Proceedings of the 10th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Yorktown, USA, 2013: 160 – 175.

[11] Selman B, Kautz H A, McAllester D A. Ten challenges in propositional reasoning and search[C]//Proceedings of 15th International Joint Conference on Artificial Intelligence, New York, 1997: 50 – 54.

[12] Prestwich S, Lynce I. Local search for unsatisfiability[C]// Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing, 2007: 283 – 296.

[13] Audemard G, Simon L. GUNSAT: A greedy local search algorithm for unsatisfiability [C]//Proceedings of the 20th International Joint Conference of Artificial Intelligence, 2007: 2256 – 2261.

[14] Gallier J H. Logic for computer science: foundations of automatic theorem proving [M]. 2003 Edition. Harper & Row Publishers Inc, 2003.