基于深度优先搜索与增量式求解的极小一阶不可满足子式提取算法*

张建民,黎铁军,张 峻,徐炜遐,李思昆
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:随着寄存器传输级甚至行为级的硬件描述语言应用越来越广泛,基于一阶逻辑的可满足性模理论(Satisfiability Modulo Theories,SMT)逐渐替代布尔可满足性(Boolean Satisfiability,SAT),在VLSI形式化验证领域具有更加重要的应用价值。而极小不可满足子式能够帮助EDA工具迅速定位硬件中的逻辑错误。针对极小SMT不可满足子式的求解问题,采用深度优先搜索与增量式求解策略,提出了深度优先搜索的极小SMT不可满足子式求解算法。与目前最优的宽度优先搜索算法对比实验表明:该算法能够有效地求解极小不可满足子式,随着公式的规模逐渐增大时,深度优先搜索算法优于宽度优先搜索算法。

关键词:形式化验证;硬件错误定位;可满足性模理论;极小不可满足子式
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2012)05-0121-06

An algorithm for extracting minimal first-order unsatisfiable subformulae on depth-first-search and incremental solving

ZHANG Jianmin, LI Tiejun, ZHANG Jun, XU Weixia, LI Sikun
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: While registering transfer level or even behavioral level, the hardware description language is widely used, and Satisfiability Modulo Theories (SMT) gradually replaces Boolean Satisfiability (SAT), and plays an important role in VLSI formal verification. A minimal unsatisfiable subformula can help automatic tools to rapidly locate the errors. A depth-first-search algorithm is proposed to extract minimal unsatisfiable subformulae in SMT to adopt depth-first searching and incremental solving strategy. The experimental results show that the depth-first-search algorithm effectively derived minimal unsatisfiable subformulae, and is more efficient than the breadth-first-search algorithm, which is the best method for computing the minimal unsatisfiable subformulae in SMT, with the number of variables and clauses in the original formulae increasing.

Key words: formal verification; error localization of hardware; Satisfiability Modulo Theories (SMT); minimal unsatisfiable subformula

自从融合冲突学习机制等启发式方法的增强DPLL(Davis-Putnam-Logemann-Loveland)算法出现之后,布尔可满足求解器得到了飞速的发展,已经能够高效地求解布尔公式的可满足性。但是目前所面临的主要挑战是抽象层次更高、更接近于真实设计的字级应用越来越受到人们的重视。由于布尔公式是将原始问题转换到位级,这就导致两个问题:一是失去了原始设计中的部分逻辑信息;二是时间与空间的开销增加,限制了解决问题的规模。例如在集成电路设计中,目前工业界所采用的形式化验证工具大多基于门级电路,开销很大,难以适应飞速增长的芯片规模,如果直接在寄存器传输级甚至行为级进行验证,能够显著地降低开销,提高解决问题的效率。针对这些需求,基于一阶逻辑的可满足性模理论SMT求解技术近年来得到了突飞猛进的发展;由于SMT采用字级建模语言,描述能力更强,更接近于高层设计,具有更大的潜力以及更广阔的应用前景,所以SMT求解技术在集成电路的设计与验证领域得到了广泛的应用。

VLSI验证领域的许多实际问题都可以规约为公式,转换为约束可满足问题来解决。如果公式是可满足的,那么表示设计是正确的,符合设计规范。如果公式不可满足,通常表明待求解的问题中包含错误与不一致,往往需要查找不满足的原因,这就要求剔除无关信息,保留能够反映其原因的一部分短句,即提取不可满足子式。而极小不可满足子式,即其所有真子式都是可满足的,能够更加准确地查找公式不可满足的原因,帮助EDA工具更迅速地定位错误。

求解不可满足子式在VLSI设计与验证领域中具有非常重要的理论与应用价值,例如电路的

* **收稿日期**:2012-01-06
基金项目:国家自然科学基金资助项目(61103083, 61133007)
作者简介:张建民(1979—),男,山西平遥人,讲师,博士,E-mail: jmzhang@nudt.edu.cn

错误定位[1]、FPGA 的布线策略[2]、硬件形式验证中的谓词抽象[3]与空洞检测[4]等。近年来涌现了众多提取布尔不可满足子式的算法[5-12],成为形式验证领域最活跃的分支之一。然而,自从抽象层次更高、表达能力更强的可满足性模理论出现之后,随着 SMT 求解技术在近两年的飞速发展,SAT 面临被 SMT 取代的趋势,因此 SMT 不可满足子式的求解方法将成为今后研究的重点及主要突破的方向。

Cimatti 等[13]首次提出了求解 SMT 不可满足子式的算法。该算法是将分离的 SMT 求解器与布尔不可满足子式提取方法结合起来,从而求解 SMT 不可满足子式,但是该方法无法保证最终得到的 SMT 不可满足子式是极小的。Zhang 等[14]首次提出了极小 SMT 不可满足子式的求解算法 BFS-MUSE。该算法采用宽度优先搜索的策略,并通过实验表明,算法能够有效地求解极小 SMT 不可满足子式,但是随着原始公式不断增大时,BFS-MUSE 算法的求解时间急剧增加,因此,迫切需要一种能够高效求解较大规模极小 SMT 不可满足子式的算法。

针对较大规模极小 SMT 不可满足子式的求解问题,提出深度优先搜索的极小不可满足子式求解算法(Depth-First-Search Minimal Unsatisfiable Subformula Extractor,DFS-MUSE)。DFS-MUSE 算法基于高效 DPLL(T)的 SMT 公式可满足性证明架构,通过为公式构造一个搜索图,在深度优先搜索与增量式求解策略的指引下,启发式地删除短句;在搜索过程中,通过共享子公式之间的冲突短句实现增量式求解,并将搜索图中结点对应的子公式进行动态排序,结合 Cache 技术等搜索图剪枝方法,尽早发现并删除那些冗余的可满足性检测,以减小搜索空间,提高搜索效率,从而快速得到极小不可满足子式。基于大量测试集的实验表明,与宽度优先搜索算法[14]相比,本文算法能够有效地求解 SMT 公式的极小不可满足子式,并且随着原始公式的复杂度逐渐增大,即其短句数与变元数增加时,深度优先搜索算法更加高效。

1 极小 SMT 不可满足子式的定义

首先给出可满足模理论 SMT 问题的定义:

定义 1 给出一个无量词的一阶逻辑公式 φ,以及一个赋值模型 M,M 包含一个非空域 $|M|$,其中对于 n 元函数变元 f,$M(f):f\rightarrow_n|M|$,对于谓词变元 P,$M(P)\subseteq|M|^n$,对于个体变元 x,$M(x)\in|M|$;公式 φ 中一个项 τ 的赋值为 $M[x]=M(x)$,$M[f(\tau_1,\cdots,\tau_n)]=M(f)(M[\tau_1],\cdots,M[\tau_n])$。那么,$M\models\varphi$ 定义为:$M\models P(\tau_1,\cdots,\tau_n)\Leftrightarrow(M[\tau_1],\cdots,M[\tau_n])\in M(P)$;$M\models\tau_0=\tau_1\Leftrightarrow M[\tau_0]=M[\tau_1]$;$M\models\neg\varphi_0\Leftrightarrow M\not\models\varphi_0$;$M\models\varphi_0\vee\varphi_1\Leftrightarrow M\models\varphi_0$ 或 $M\models\varphi_1$;$M\models\varphi_0\wedge\varphi_1\Leftrightarrow M\models\varphi_0$ 且 $M\models\varphi_1$。如果存在这样的赋值模型 M,使得 $M\models\varphi$,那么称公式 φ 是可满足的;否则,称公式 φ 是不可满足的。

SMT 求解技术从最初的 eager 方法,到多数 SMT 求解器采用的 lazy 方法,再到最新的 DPLL(T)算法,求解效率越来越高,SMT 求解器也不断发展与完善,已经能够解决较大规模的问题,为求解 SMT 不可满足子式提供了理论基础与实践平台。

定义 2 给出一个不可满足的 SMT 公式 φ,ψ 是公式 φ 的一个 SMT 不可满足子式当且仅当 ψ 是不可满足的,并且 $\psi\subseteq\varphi$。ψ 是极小 SMT 不可满足子式当且仅当 $\forall\phi\subset\psi$,使得 ϕ 是可满足的。

根据定义 2,对于 φ 的一个不可满足子式 ψ,需要测试 ψ 的所有真子式的可满足性,才能确定 ψ 是否为极小不可满足子式,算法复杂度很高。

引理 1[14] 给出不可满足的 SMT 公式 φ,及其不可满足子式 $\psi=\wedge_1^nC_i$,C_i 为短句,那么 ψ 是极小 SMT 不可满足子式,当且仅当从 ψ 中删除任意一个短句 $\forall i$,$C_i\in\psi$,$1\leqslant i\leqslant n$,都使得 $\psi\backslash C_i$ 是可满足的。

根据引理 1,算法只需将不可满足子式删除其中任意一个短句后,测试剩余短句构成公式的可满足性,即可判断其是否为极小不可满足子式,显著地降低了极小 SMT 不可满足子式的判别难度。

2 DFS-MUSE 算法

DFS-MUSE 算法将 SMT 公式的子公式组织为搜索图的结构,采用深度优先搜索与增量式求解策略提取极小不可满足子式。下面给出 SMT 公式搜索图的定义。

定义 3 给出一个不可满足的 SMT 公式 φ,若一个有向无环图 $G(V,E,s)$ 满足下面的条件:(1)包含唯一的汇结点 s,对应的 SMT 公式为 φ;(2)$\forall v\in V\backslash s$,是其父结点 $p\in V$ 的第 k 个子结点,假设结点 p 对应的公式为 $\psi=\wedge_1^nC_i$,那么结点 v 对应的子公式为 $\phi_k=\wedge_1^nC_i\backslash C_k$;而边 $e_{pv}\in E$ 表示从父结点 p 指向子结点 v。则称 $G(V,E,s)$ 为 SMT 公式 φ 的搜索图。

$G(V,E,s)$ 的结点集合 V 分为三类:live 结

点、dead 结点与 pending 结点。其定义分别为:live 结点表示其对应的子公式是不可满足的,并且搜索过程正运行于当前结点的分支上,该类结点可以继续向下扩展;dead 结点是指其对应的子公式是可满足的,这类结点不能继续向下扩展;pending 结点表示当前图中未被搜索过的结点。

搜索图中三类结点之间的转换关系为:从 pending 结点到 dead 结点的转换关系为,当一个 pending 结点所对应的子公式被证明为可满足时,该结点就转化为 dead 结点;从 pending 结点到 live 结点的转换关系为,当算法的搜索过程到达一个 pending 结点时,并且其对应的子公式是不可满足时,就将该结点转化为 live 结点。

假设一个不可满足的 SMT 公式 φ,DFS-MUSE 算法为公式 φ 构造一个搜索图 $G(V,E,s)$,汇结点 s 对应初始公式 φ,对于搜索图 G 中的每个结点 $\forall v \in V$,v 都对应公式 φ 的一个子公式。那么,定理 1 给出了 SMT 公式搜索图中 live 结点及 dead 结点与极小不可满足子式的关系:

定理 1[14]　给出一个 SMT 公式 φ,及其搜索图 $G(V,E,s)$,那么 G 中的 dead 结点对应的公式不包含不可满足子式。G 中的 live 结点对应的公式是不可满足子式;一个 live 结点对应的公式是极小 SMT 不可满足子式当且仅当其所有的直接子结点都是 dead 结点。

深度优先搜索算法 DFS-MUSE 与宽度优先搜索算法 BFS-MUSE[14] 之间的主要区别在于搜索策略的不同。BFS-MUSE 算法是在搜索图的分支上水平地进行搜索,首先测试同样长度子式的可满足性,而后再考虑较小的子式,那么随着 SMT 公式所包含的短句数与变元数不断增加,会因为在水平方向存在较多的冗余可满足性检测,从而导致算法的性能迅速下降。而 DFS-MUSE 算法是在搜索图的分支内垂直地进行搜索,首先测试当前子图内长度依次递减的子式的可满足性,而后再测试相邻子图中公式的可满足性;根据定理 1,沿着 live 结点搜索,必然得到极小不可满足子式,因此能够有效地减少冗余可满足性检测,从而提高算法的效率。

极小 SMT 不可满足子式的深度优先搜索算法 DFS-MUSE 的伪代码如图 1 所示。DFS-MUSE 算法的输入是 SMT 公式 *formula*,采用先求解不可满足子式,后提取极小不可满足子式的增量式策略,便于共享不同分支之间的冲突短句与子公式。函数 ComputeUS 返回公式 *formula* 的不可满足子式。算法在得到不可满足子式之后,进行分支判断,若等于初始公式,则表明极小不可满足子式为 *formula* 本身;否则,根据定理 1 的结论,采用函数 VerifyMinimalUS 验证不可满足子式 *SmallUS* 的极小性。若 *SmallUS* 为极小不可满足子式,则算法终止;否则,以深度优先的搜索方式,将当前不可满足子式 *SmallUS* 作为输入公式,递归地求解极小不可满足子式。DFS-MUSE 算法的一个特点是:当改变搜索的分支顺序时,能够得到不同的极小 SMT 不可满足子式。

```
DFS_MUSE(formula)
1   SmallUS = ComputeUS(formula)
2   if (SmallUS == formula) then
3      return formula
4   else
5      IsMinUS = VerifyMinimalUS(SmallUS)
6      if (IsMinUS) then
7         return SmallUS
8      else
9         MinimalUS = DFS_MUSE(SmallUS)
10        return MinimalUS
```

图 1　深度优先搜索算法

Fig. 1　The depth-first-search algorithm

```
ComputeUS(formula)
1   ite = EliminateITE(formula)
2   abs = AbstratExpression(ite)
3   for (arity = 0; arity < formula.size; arity++)
4      interim = abs
5      for (count=formula.size; count>0; count--)
6         SmallUS = GraphPruning(interim)
7         cnf = BooleanConversion(SmallUS)
8         IsSAT = SATSolve(cnf)
9         if (!IsSAT) then
10              return SmallUS
11     abs = DynamicVarOrder(abs)
12  return formula
```

图 2　ComputeUS 求解过程

Fig. 2　The function called ComputeUS

图 2 为不可满足子式求解函数 ComputeUS 的伪代码。该函数基于 DPLL(T) 可满足性证明算法,通过为输入的公式构造一个搜索图 $G(V,E,s)$,利用深度优先的遍历方式,查找图中的 live 结点,返回其对应的不可满足子式。算法中采用了三种优化技术,优化的对象分别为变元、短句与子公式。

(1)动态变元排序:算法每一次循环时,将本次迭代中经过可满足测试的变元的优先级降低,使得下次循环对其他变元进行可满足性测试。

(2)共享冲突短句:记录子公式的冲突学习短句,当证明其他子公式的可满足性时,若包含相

同的导致冲突的短句,那么直接将学习短句加入到该子公式。

(3)子公式 Cache 技术:将搜索过的 dead 结点对应的子公式进行标记,当搜索到的 pending 结点与之前已标记的子公式相同时,直接将 pending 结点转换为 dead 结点。

算法首先调用函数 EliminateITE 消除 SMT 公式中 If-Then-Else 结构项。接着函数 Abstrat-Expression 将公式中的所有文字,采用抽象变元进行置换。而后算法在公式的搜索图空间内求解不可满足子式,其中 GraphPruning 函数启发式地删除一个子图分支上的短句,通过共享子公式之间的冲突短句以及子公式 Cache 技术,移除那些冗余的可满足性检测。函数 BooleanConversion 是将 SMT 公式抽象为 CNF 形式的布尔公式。SATSolve 函数调用实现 DPLL 算法的 SAT 求解器,验证抽象后的布尔公式的可满足性。如果公式是不可满足的,那么就得到一个 live 结点对应的不可满足子式;否则,表示当前结点是 dead 结点,根据定理 1,它必定不包含不可满足子式,因此将其丢弃。而后采用动态变元排序技术调整公式的结构,将上一次循环已测试的短句的优先级降低,避免下一次循环不必要地测试。若循环结束,则返回公式 *formula*,表明不可满足子式为原始公式本身。

3 实验结果与分析

3.1 实验结果

基于业界公认的 SMT Competition 2010 测试集中的公式,将本文的 DFS-MUSE 算法与求解 SMT 不可满足子式算法 Lemma Lifting + AMUSE[13] 以及 BFS-MUSE 算法[14] 进行了对比与分析。所有算法的输入都是 SMT-LIB 格式的公式,运算时间都是以秒(s)为单位,运行时限都设置为 1800s。算法的运行环境是 2.5GHz 的 Athlon * 2 CPU,内存 2GB,操作系统为 Linux 的机器。

DFS-MUSE 算法采用 C ++ 实现,DFS-MUSE 与 BFS-MUSE 算法实现的下载地址为:http://www.ssypub.org/~zjm/,算法中集成了一个开源的 SMT 求解器 ArgoLib[15]。表 1 给出了 DFS-MUSE 算法、BFS-MUSE 算法与 Lemma Lifting + AMUSE 算法基于测试集中 15 个典型公式的实验结果。表中第 2 列数据是每个公式所包含的变元数;第 3 列表示每个公式所包含的短句数;第 4 ~ 5 列给出了 Lemma Lifting + AMUSE 算法[13] 的结果短句数与运行时间,该算法不能保证不可满足子式的极小性。第 6 列与第 7 列分别是深度优先搜索算法 DFS-MUSE 的运行时间与极小不可满足子式包含的短句数。最后两列分别是宽度优先搜索算法 BFS – MUSE 的执行时间与极小不可满足子式的大小。

表 1 三种算法在 SMT COMP 测试集上的实验结果

Tab. 1 The experimental results of three algorithms on SMT COMP benchmarks

Benchmarks	variables	clauses	求解 unsat core[13]		DFS – MUSE		BFS – MUSE	
			time(s)	size	time(s)	size	time(s)	size
bad_echos_ascend. base	58	259	1.82	115	5.18	11	5.03	11
sc_init_frame_gap. base	58	265	1.78	119	5.11	13	5.14	13
good_frame_update. induction	89	439	8.58	208	29.00	161	28.74	161
good_frame_update. base	89	467	20.56	402	72.81	311	67.75	311
windowreal – safe2 – 2	37	404	0.44	279	0.73	188	0.68	188
windowreal – safe – 2	37	411	0.45	286	0.75	195	0.68	195
lpsat – goal – 1	83	1345	0.80	192	1.67	17	1.69	17
lpsat – goal – 2	142	2650	5.52	1693	12.30	1283	17.16	1283
lpsat – goal – 3	201	3955	18.35	2968	43.47	2548	68.55	2548
windowreal – no_t_deadlock – 15	219	2933	45.20	2198	176.96	1351	179.53	1351
windowreal – no_t_deadlock – 16	233	3128	52.36	2342	208.13	1441	236.58	1441
windowreal – no_t_deadlock – 17	247	3323	73.11	2485	293.38	1531	303.32	1531
windowreal – no_t_deadlock – 18	261	3519	85.02	2627	347.24	1622	391.02	1622
windowreal – no_t_deadlock – 19	275	3714	112.57	2764	463.78	1712	497.97	1712
windowreal – no_t_deadlock – 20	289	3909	715.20	2897	547.44	1802	633.77	1802

由于 Lemma Lifting + AMUSE 算法不能求解极小不可满足子式,将三种算法放在同一个图中比较运行时间并不合理,因此仅在图中比较 DFS-MUSE 与 BFS-MUSE 算法。图 3 ~ 图 6 给出了

DFS-MUSE 算法与 BFS-MUSE 算法基于 SMT Competition 2010 测试集中的 4 组公式的实验结果,其中横坐标轴表示原始公式所包含的短句数,而纵坐标轴采用了对数(lg)坐标,表示算法的运行时间,单位为 s,两条曲线分别表示 DFS-MUSE 与 BFS-MUSE 随公式短句数递增时运行时间的变化趋势。图 3 中的测试集是 inf-bakery-mutex,其公式所包含的短句数范围是 65 ~ 1053。图 4 中的测试集为 windowreal-no-t-deadlock,其公式的短句数范围是 203 ~ 3908。图 5 基于 pursuit-safety 测试集,短句数的范围是 113 ~ 1763。而图 6 中的测试集是 gasburner-prop3,短句数范围是 28 ~ 522。

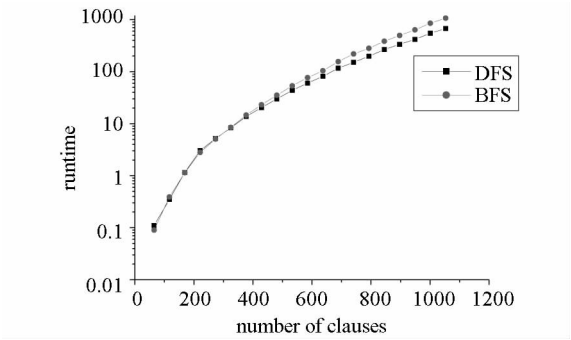

图 3　inf-bakery-mutex 测试集

Fig. 3　The inf-bakery-mutex benchmark

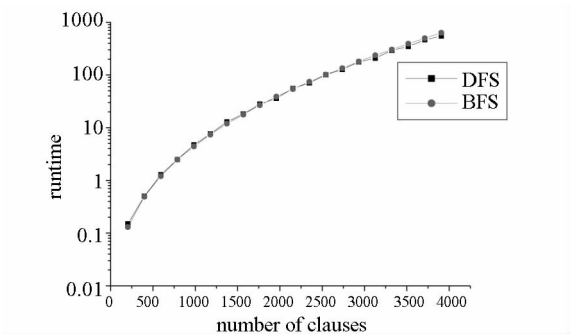

图 4　windowreal-no_t_deadlock 测试集

Fig. 4　The windowreal-no_t_deadlock benchmark

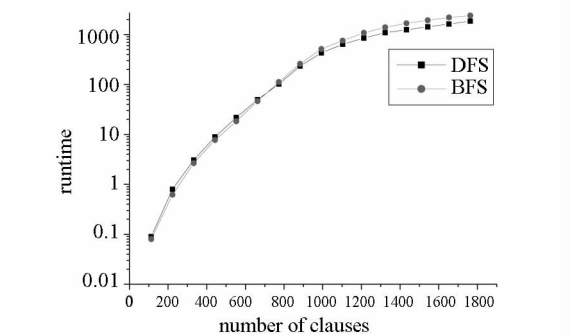

图 5　pursuit-safety 测试集

Fig. 5　The pursuit-safety benchmark

图 7 给出了 DFS-MUSE 算法与 BFS-MUSE 算法基于 SMT Competition 2010 测试集的实验结果,运行时限为 1800s,其中横坐标轴表示深度优先搜索算法 DFS-MUSE 的运行时间,纵坐标轴表示宽度优先搜索算法的 BFS-MUSE 的运行时间,二者都采用了对数(lg)坐标,时间单位为 s。

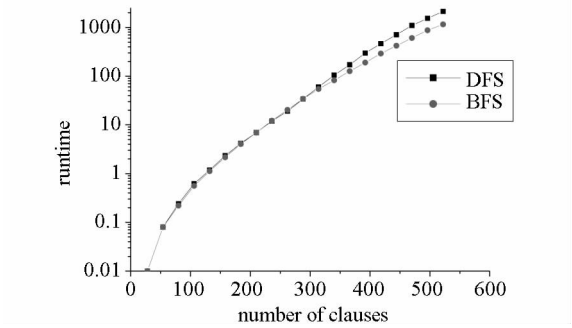

图 6　gasburner-prop3 测试集

Fig. 6　The gasburner-prop3 benchmark

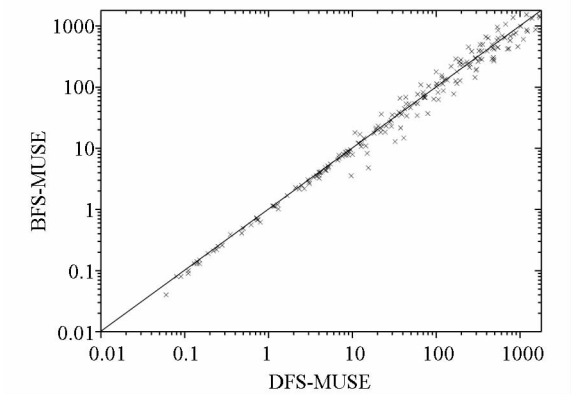

图 7　DFS-MUSE 与 BFS – MUSE 算法的对比

Fig. 7　Comparision between DFS-MUSE and BFS-MUSE

3.2　分析与结论

从表 1 中能够直观地看出,DFS-MUSE 和 BFS-MUSE 算法的运行时间大于 Lemma Lifting + AMUSE 算法,但前者结果所包含的短句数远小于后者;DFS-MUSE 和 BFS-MUSE 算法能够得到精确的极小不可满足子式,而极小不可满足子式的求解复杂度远大于非极小不可满足子式。另外,根据表 1 的实验结果,对于测试集中的 SMT 公式,极小不可满足子式所包含的短句数远远小于原始公式的短句数,通常占公式总短句数的 1% ~50% 左右,因此可以得到下面的结论:DFS-MUSE 算法能够有效地求解 SMT 公式的极小不可满足子式,而极小不可满足子式能够帮助 EDA 工具更加迅速准确地诊断与定位错误,具有重要的理论与应用价值。

在表 1 中,当公式所包含的短句数与变元数较少时,BFS-MUSE 算法较快,而后随着公式的短句数与变元数逐渐增加时,DFS-MUSE 算法更加

高效;根据图3~图5,可以得到相同的结论,并且随着公式复杂度的增加,两者的运行时间差距越来越大;在图6中,BFS-MUSE算法始终优于DFS-MUSE算法,其原因是这组测试集中的所有公式包含的短句数与变元数都比较少。在图7中,随着公式复杂度逐渐增加时,越来越多的实验结果位于对角线上方,表明对于较大规模的公式,DFS-MUSE算法效率更高。

综上所述,可以得出一个结论:当原始公式所包含的变元数与短句数较少时,DFS-MUSE算法和BFS-MUSE算法的运行时间基本相当,在多数情况下宽度优先搜索的算法更快。但是,随着公式的复杂度逐渐增加,即公式包含的短句数与变元数不断增加时,深度优先搜索策略的优势逐渐体现出来,变得更加高效,并且随着公式复杂度的增加,这种趋势更加明显。其主要原因是BFS-MUSE算法的实现较为简单,单位时间内运行的循环次数更多,因此对于短句数较少的公式,这个优势更加凸显,其运行时间较少,但当公式较大时,宽度优先搜索方式存在较多的冗余检测,导致算法效率迅速下降,而DFS-MUSE算法是在发现一个不可满足的子公式后,就在该子图上以深度优先的方法进行搜索,根据定理1的结论,必然能够到达极小不可满足子式,其搜索过程中启发式的策略更好,因此当公式复杂度较大时,DFS-MUSE算法更加高效。

4 结束语

针对较大规模极小SMT不可满足子式的求解问题,提出了深度优先搜索算法DFS-MUSE。算法为SMT公式构造搜索图,采用深度优先搜索与增量式求解策略,并融合一些启发式方法,从而得到极小不可满足子式。实验结果表明,该算法都能有效地求解极小SMT不可满足子式,并且随着原始公式所包含的变元数与短句数逐渐增加时,深度优先的搜索策略会更加高效。在下一步工作中将采用更多的启发式方法,优化搜索图的剪枝过程,进一步提高算法的效率。

参考文献(References)

[1] SuelflowA, Fey G, Bloem R, et al. Using unsatisfiable cores to debug multiple design errors[C]//Proceedings of ACM GLVLSI'08. Orlando: ACM, 2008: 77-82.

[2] Nam G J, Aloul F, Sakallah K, et al. A comparative study of two Boolean formulations of FPGA detailed routing constraints [C]//Proceedings of ISPD'01. Sonoma County: ACM, 2001: 222-227.

[3] Jain H, Kroening D. Word level predicate abstraction and refinement for verifying RTL Verilog[C]//Proceedings of DAC'05. Anaheim: ACM, 2005: 445-450.

[4] Simmonds J, Davies J, Furfinkel A. Exploiting resolution proofs to speed up LTL vacuity detection for BMC[J]. International Journal on Software Tools for Technology Transfer, 2010, 12(5): 319-335.

[5] Oh Y, Mneimneh M N, Andraus Z S, et al. AMUSE: a minimally-unsatisfiable subformula extractor[C]//Proceedings of DAC'04. San Diego: ACM, 2004: 518-523.

[6] 邵明, 李光辉, 李晓维. 极小布尔不可满足子式的提取算法[J]. 计算机辅助设计与图形学学报, 2004, 16(11): 1542-1546.
SHAO Min, LI Guanghui, LI Xiaowei. Algorithms for extracting minimal unsatisfiable Boolean sub-formula[J]. Journal of Computer Aided Design & Computer Graphics, 2004, 16(11): 1542-1546. (in Chinese)

[7] Gershman R, Koifman M, Strichman O. Deriving small unsatisfiable cores with dominator[C]//Proceedings of CAV'06. Seattle: LNCS, 2006: 109-122.

[8] Gregoire E, Mazuer B, Piette C. Local-search extraction of MUSes[J]. Constraints, 2007, 12(3): 325-344.

[9] 陈振宇, 徐宝文, 周从华. 一种基于消解的变量极小不可满足子公式的提取方法[J]. 计算机研究与发展, 2008, 45(s1): 43-47.
CHEN Zhenyu, XU Baowen, ZHOU Conghua. A resolution-based approach for extracting variable minimal unsatisfiable subformulas[J]. Journal of Computer Research and Development, 2008, 45(s1): 43-47. (in Chinese)

[10] Liffiton M H, Sakallah KA. Algorithms for computing minimal unsatisfiable subsets of constraints[J]. Journal of Automated Reasoning, 2008, 40(1): 1-33.

[11] 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)

[12] Nadel A. Boosting minimal unsatisfiable core extraction[C]//Proceedings of FMCAD'10. Lugani: IEEE, 2010: 221-229.

[13] Cimatti A, Griggio A, Sebastiani R. A simple and flexible way of computing small unsatisfiable cores in SAT modulo theories[C]//Proceedings of SAT'07. Lisbon: LNCS, 2007: 334-339.

[14] 张建民, 沈胜宇, 李思昆. 求解极小SMT不可满足子式的宽度优先搜索算法[J]. 计算机辅助设计与图形学学报, 2009, 21(7): 984-990.
ZHANG Jianmin, SHEN Shengyu, LI Sikun. A breadth-first-search algorithm for deriving minimal unsatisfiable subformulae in satisfiability modulo theories[J]. Journal of Computer Aided Design & Computer Graphics, 2009, 21(7): 984-990. (in Chinese)

[15] Maric F, Janicic P. Argo-lib: a generic platform for decision procedures[C]//Proceedings of IJCAR'04. Cork: IEEE, 2004: 231-217.