使用SAT求解器产生所有极小冲突部件集

赵相福[1,2], 欧阳丹彤[1,2]
(1. 吉林大学计算机科学与技术学院, 吉林长春 130012; 2. 吉林大学符号计算与知识工程教育部重点实验室, 吉林长春 130012)

摘 要: 产生所有的极小冲突部件集为基于模型诊断中的一个重要步骤. 本文将待诊断系统的行为模型及观测分别使用合取范式(CNF)形式的文件描述, 从而提出将判定系统组件子集是否为冲突集的问题转化为: 首先提取相关组件的CNF模型及观测, 然后调用成熟的SAT求解器判定可满足性. 随后, 通过有效地结合CSISE-tree等方法来产生所有的极小冲突集. 为进一步提高效率, 给出了充分利用系统输入/输出结构信息的启发式策略. 实验结果表明, 使用结合SAT求解器及CSISE-tree等方法能够较快产生所有极小冲突集, 并且启发式策略使得求解效率进一步提高(平均提高约21%, 最高者甚至达到约48%).

关键词: 基于模型的诊断; 冲突集; 可满足性; SAT求解器; 启发式
中图分类号: TP306 文献标识码: A 文章编号: 0372-2112 (2009) 04-0804-07

Deriving All Minimal Conflict Sets Using Satisfiability Algorithms

ZHAO Xiang-fu[1,2], OUYANG Dan-tong[1,2]
(1. *School of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*;
2. *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, Jilin* 130012, *China*)

Abstract: An important step in model-based diagnosis is to derive all minimal conflict sets. In this paper, we propose an approach for judging whether a component set is a conflict set using SAT solvers with some satisfiability algorithms. Firstly the system model and the obtained observations are described in conjunctive normal form. Then all the related clauses of the component set to be considered are extracted, which act as the input of SAT solvers. Hence all the conflict sets can be derived using CSISE-tree or other algorithms combing with a SAT solver. Heuristic strategies are introduced to best suit the input/output structural information of the system. Results show that all the minimal conflict sets can be effectively computed. Besides, the efficiency is greatly improved using heuristic strategies: about 21% and 48% for the average and highest rate respectively.

Key words: model-based diagnosis; conflict set; satisfiability; SAT solvers; heuristic

1 引言

基于模型的诊断(Model-Based Diagnosis, MBD)为人工智能领域中一个十分活跃的研究分支, 对整个人工智能领域研究起着重要推动作用[1]. MBD专家Reiter指出用于产生所有极小冲突集(Minimal Conflict Sets, MCSs)的冲突识别和用于产生所有极小碰集(hitting-set)的候选产生, 为产生最终诊断结果的两个非常重要的步骤[2].

许多学者对冲突识别进行了研究. de Kleer首次提出了冲突集的概念[3], 为基于模型的诊断算法奠定了重要的基础, 随后他又提出了一些冲突识别的策略以及一个广为应用的推理引擎ATMS(Assumption-based Truth Maintenance System)[4]. ATMS具有正确性、完备性和一致性的特点. 但是在整个推理系统中使用它的时候, 由于模型本身的不完备, 规则设计的不合理等原因, 这些特性会受到很大影响[5].

一些使用定理证明器的方法, 比如DART[6]等也可以用作冲突识别, Haenni[7]在计算中使用归结的方法. 但由于传统的定理证明器及归结方法效率较低, 因而往往难以实际应用.

国内学者也做了一些冲突识别的相关工作, 如由栾尚敏等人提出的利用系统结构信息求解极小冲突集的方法[8], 代树武[9]等人提出利用元件参数矩阵来指导冲突的求解过程, 以及方敏提出的首先离线求冲突候选然后在线求解极小冲突的方法[10]等等. 然而它们或者依赖于系统的结构信息, 或者空间复杂度较高, 且通用性较差.

为了产生所有的极小冲突集, 在Hou的CS-tree[11]及Han和Lee提出的Inverse CS-tree、CS-tree with Mark Set[12]的基础上, 我们[13]提出了两种改进的算法CSSE-tree以及CSISE-tree, 并指出这些方法依然可用于候选产生阶段, 实现软件复用. 但这些更倾向于给出产生框

收稿日期: 2008-04-10; 修回日期: 2008-11-28
基金项目: 国家自然科学基金重大项目(No. 60496320, 60496321); 国家自然科学基金(No. 60773097, 60873148); 新世纪优秀人才支持计划; 吉林省科技发展计划(No. 20060532, 20080107); 欧盟项目(No. TH/Asia Link/010(111084)); 吉林大学"985工程"研究生创新基金(No. 20080115)

架, 并没有给出具体且高效的判定方法.

最近几年的多数研究人员往往避开研究求解 MCSs 这一"领域相关"的问题, 而一般假设所有 MCSs 均已知的情况下后续的求解候选诊断问题, 即求解 MCSs 的所有极小 hitting-set 问题[14]. 我们也曾对如何产生所有的极小 hitting-set 问题进行过研究, 首先给出了判定一个组件集是否为 hitting-set 的方法 IsHS, 然后结合带终止节点的 SE-tree 给出了产生所有极小 hitting-set 的方法 HSSE-tree[15].

类似使用 IsHS 判定一个集合是否为 hitting-set 的方法, 并受这种方法启发, 本文探索了如何判定一个组件集是否为冲突集的方法. 从一方面来说, 所谓冲突集, 即为该集合内所有部件均正常工作是不可满足的, 于是可将判断一个部件集是否为冲突非常自然地转化为可满足问题: 即该集合内所有部件均正常的逻辑表示的合取公式是不可满足的. 于是可以使用 SAT 求解器进行验证, 此处的关键在于 SAT 求解器的效率. 而与之对应的是, SAT 求解器在最近几年得到了广泛的应用研究和改进, 如特别是在自动规划、模型检测、电路等价性验证[16] 等领域, 及最近在人工智能领域权威国际会议 IJCAI-07[17] 中的文章中也提出到 MBD 中的应用. 每年一度的国际 SAT 竞赛促使 SAT 求解器的效率不断提高(详见: http://www.satcompetition.org/). 因而, 为判定一组件集是否为冲突, SAT 求解器可为一个非常好的选择.

给出 SAT 求解器作为判定一个组件集是否为冲突集的算法 IsCS 之后, 可以结合 CSISE-tree、CSSE-tree 等框架方法来产生所有的极小冲突集.

2　预备知识

2.1　基于模型诊断的相关概念和定理

定义 1[2]　一个系统定义为一个三元组 $(SD, COMPS, OBS)$, 其中: SD 为系统描述, 是一阶谓词公式的集合; $COMPS$ 为系统的组成部件集合, 是一个有限的常量集; OBS 为一观测集, 是一阶谓词公式的有限集.

以下使用一元谓词 $AB(\cdot)$ 表示"abnormal", $AB(c)$ 为真当且仅当 c 反常, 其中 $c \in COMPS$.

定义 2[2]　冲突集(CS) 是一个部件集 $\{c_1, c_2, \cdots, c_n\} \subseteq COMPS$, 使得 $SD \cup OBS \cup \{\neg AB(c_1), \neg AB(c_2), \cdots, \neg AB(c_n)\}$ 不一致.

称某冲突集为极小冲突集(MCS), 当且仅当该冲突集的任意真子集都不是冲突集.

根据定义可知, 为判定一个组件集是否为冲突集则需要判定该组件集中所有组件的正常行为描述与相关的系统描述及观测描述是否逻辑一致.

此外, 由冲突集的定义还可以得到如下结论: 若集合 A 为一极小冲突集, 且集合 $B \subseteq COMPS$, 集合 $C \subseteq COMPS$, 则:

① 若集合 $B \subset A$, 则 B 不是冲突集;

② 若集合 $C \supset A$, 则 C 为非极小的冲突集.

此结论为以后产生所有极小冲突集扩展的过程(如 CSISE-tree 方法等)的修剪规则提供理论基础.

2.2　可满足性算法和 SAT 求解器

命题可满足性问题即 SAT 问题, 并不是一个新的话题, 但由于它的广泛应用, 目前对它的研究越来越活跃. 它的基本思想为: 将任何一个逻辑命题转化为一个子句集, 通过各种可满足性方法来判定该子句集是否可满足从而决定该命题是否成立.

现有许多学者研究 SAT, 每年一度的 SAT 竞赛使得产生了一些成熟的 SAT 求解器. 一般 SAT 求解器以 CNF 文件作为输入, 每一个命题公式(集) 均可以转化为一个 CNF 文件描述. 例如一个命题逻辑公式集 $S: \{A \rightarrow B, B \rightarrow C, C \rightarrow D, \neg D, A\}$, 可以表示为如下一个 CNF 文件(使用 1, 2, 3, 4 分别表示变量 A, B, C, D):

```
p cnf 4 5   //CNF 文件语法, p cnf 为关键字;
            //4 表示变量总数, 5 表示子句总数
-1 2 0      //负数表示相应的否定文字;
-2 3 0      //每一个子句以 0 结尾
-3 4 0
-4 0
 1 0
```

于是为了判定命题公式集 S 是否为真, 只需将 S 转化为 CNF 文件描述, 然后调用 SAT 求解器即可得到结果: 若可满足, 则 S 为真; 否则 S 为假.

基于这种思想, 为了判定一个部件集是否为冲突集, 即判定该组件集中所有组件的正常行为描述与相关的系统描述及观测描述是否逻辑一致, 只需将与该部件集相关的系统描述及观测描述的逻辑公式以及每一个部件正常的子句一起构建一个 CNF 文件, 然后调用 SAT 求解器求解即可.

3　算法描述及复杂度分析

根据以上的相关知识, 可给出算法的基本思想: 将判断一个部件集是否为冲突的问题转化为 SAT 求解问题, 仅将所含部件的行为描述以及相关的观测代入 SAT 求解器, 判断是否可满足; 若不可满足, 则该部件集构成一个冲突集. 为了产生所有可能的极小的冲突集, 可以使用文献[13] 提出的 CSISE-tree 等方法来不断地产生.

3.1　使用 CNF 对系统及观测建模

给定一个系统, 与 Reiter 的方法[2] 不同, 本文使用命题逻辑对其行为建模. 此外, 不仅对于系统中的每一个部件使用一个变量表示, 而且所有的联结点也分别

使用不同的变量表示该点的取值. 由于考虑基于一致性诊断, 对系统中的每一个部件, 仅考虑每一部件的所有可能的正常行为模型的子句表示. 每一个组件的行为使用命题逻辑公式表达, 当然每一个命题逻辑公式也可以等价转化为相应的子句形式. 于是, 整个系统的行为模型即为所有组件的行为模型, 即所有组件子句模型的集合, 当然也可看作一个 CNF 语句(后面使用 SD. cnf 文件存储表示). 另外, 由于对各联结点也分配了变量表示, 因而在系统的行为模型描述中, 其实也隐含了系统组件的部分结构行为描述.

以如图 1 所示的全加器为例, 说明如何对系统建模以及使用 CNF 文件表示. 设 1, 2, 3 分别表示相应的输入节点变量, 4, 5 分别表示相应的输出节点变量, 6, 7, 8 则表示相应的内部联结点变量. 此外, 所含的组件变量异或门 X1, X2、与门 A1, A2 以及或门 O1 则分别使用 9, 11, 10, 12, 13 表示. 另外, 为简化表示且在不引起混淆的情况下, 设 9-13 也同时表示各相应组件的正常谓词, 即它们取正值分别表示各部件正常工作; 如 9 表示 OK(X1) 等. 于是该全加器的系统描述可使用一个 CNF 文件如图 1 所示.

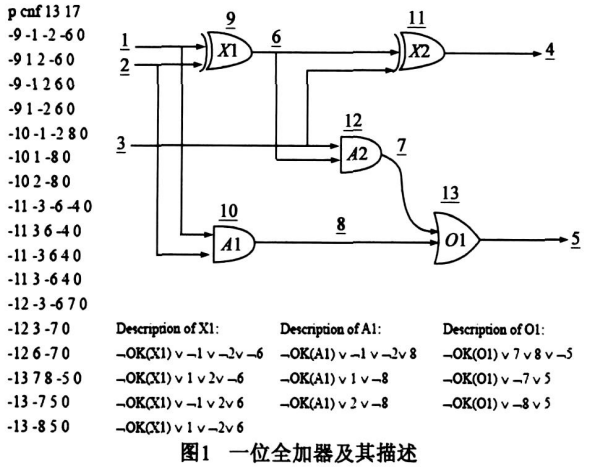

图1 一位全加器及其描述

类似地, 可以给出观测的 CNF 文件描述. 比如, 当前得到观测值: 观测点 1, 2, 3, 4, 5 的观测分别为低电平、高电平、低电平、高电平、低电平. 于是得到如下的 CNF 子句描述:

- 1 0
2 0
- 3 0
4 0
- 5 0

3.2 IsCS 算法描述

首先给出判定一个组件集为冲突集的算法.

Function IsCS(SubCOMP[ ])

输入: 待判定的系统组件子集 SubCOMP;

输出: BOOL 值, 若为冲突集则返回"真", 否则返回"假".

step1 根据当前的组件子集 SubCOMP, 从整个系统模型描述文件 SD. cnf 中仅取出与 SubCOMP 中每一个组件相关的系统描述子句放入一个新的 CNF 文件 SD_C_OBS. cnf 中(注: 此处可以通过一个组件-子句映射实现, 即该映射能够记录每一组件所关联的子句描述);

step2 将 SubCOMP 中的每一组件的正常行为描述子句追加到文件 SD_C_OBS. cnf;

step3 将所有观测描述文件(设为 OBS. cnf) 的内容追加到文件 SD_C_OBS. cnf;

step4 调用 SAT 求解器, 判定文件 SD_C_OBS. cnf 是否可满足. 若不可满足, 则 SubCOMP 为冲突集, 返回"真"; 反之不是冲突集, 返回"假".

由于要判定一个组件子集是否为冲突集, 也就是要检测它们都正常工作时它们的所有行为是否可满足, 因此, 仅考虑与之相关的组件的行为描述即可(step1). 其他组件正常与否均不可知, 而仅仅必须保证的是所考虑的组件必须正常工作(step2). 另外, 所有的观测值是已知的, 所以也要考虑(step3). 然后就可以通过调用 SAT 求解器来进行判定是否可满足, 从而进一步来决定是否为冲突.

给定系统描述 SD. cnf 后, 所有的组件——子句映射可以离线构建, 因而该算法 step1 在提取相应的子句描述时, 可以通过随机定位方法直接查找到相关的子句描述, 时间复杂度与子组件集长度成线性.

3.3 产生所有的极小冲突集算法

下面以 CSISE-tree 方法为例, 说明如何使用 IsCS 判定算法来产生所有的极小冲突集. 首先来看一下翻转的 SE-tree(inverse SE-tree(ISE-tree)).

与 SE-tree[18] 类似, 可以定义一个翻转的 SE-tree, 即按照子集长度由长到短的顺序生成一个给定集合的所有子集的树. 我们称该树为 ISE-tree. 比如集合 $\{A, B, C, D\}$ 的完全 ISE-tree 如图 2 所示.

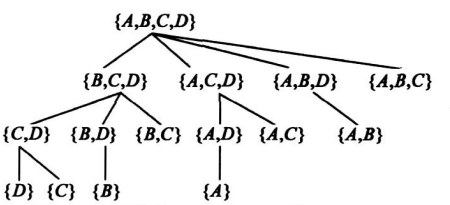

图2 有序集合{A, B, C, D}的ISE-tree

一个完全 ISE-tree 同样可以按照某个给定的顺序(如字母顺序, 数字顺序等) 系统地枚举某个给定集合的所有子集. 为产生所有的极小冲突集, 可根据某些特征, 通过加入一些修剪规则来减少许多不必要的扩展, 从而提高效率.

定义 **3**　一个有序的初始冲突集 S(初始可假设为 *COMPS*, 即所有的部件的集合) 的 CSISE-tree T_I 为如下一棵树:

①树 T_I 的每一个节点 n 均由三部分组成: 节点标签集 S_n, 父边标签 E_{p_n}, 子边标签集 E_{c_n}. 并且树根节点 T_I: $S_{T_I}=S$, $\{E_{P_{T_I}}\}=\{\}$, $E_{C_{T_I}}\}=S$.

②对于每一个节点 n, 如果 $|S_n|=1$, 那么终止; 否则假定 $E_{c_n}=\{e_1, e_2, \cdots, e_k\}$, 对于每一个 $e_i \in E_{c_n}(1 \leqslant i \leqslant k)$, 存在一个子节点 n_{e_i}, 相应的父子边标识为 e_i, 并且节点 n_{e_i} 组成为: $Sn_{e_i}=S_n-\{e_i\}$, $E_{p_{n_i}}=e_i$, $E_{c_{n_i}}=\{e_{i+1}, \cdots, e_k\}$.

为产生系统的所有极小冲突集, 通过如下过程生成一个带有剪枝的 CSISE-tree T_I.

Algorithm CSISE-tree

输入: 系统描述文件 SD. cnf, 观测描述文件 OBS. cnf

输出: 所有可能的极小冲突集

过程 **1**　生成初始冲突集 S(注: 初始情况下可以为系统所有的组件构成的集合) 的所有极小冲突集的一个带有修剪的 CSISE-tree T_I:

①宽度优先产生树 T_I;

②修剪规则:

同一层的节点从右至左使用算法 *IsCS* 判断:

(ⅰ)若节点 n 的标签集(SubCOMP) 可满足, 则标识为"×", 不再扩展该节点.

(ⅱ)若节点 n 的标签集(SubCOMP) 不可满足, 且 $E_{c_n} \neq \{\}$:

for(每一个在 n 的右面标识为"×"的节点 m)

　if($E_{p_m} \in E_{c_n} \wedge S_n-\{E_{p_m}\} \subseteq S_m$)

　　$E_{c_n}=E_{c_n}-\{E_{p_m}\}$;

③标识规则:

如果节点 n 的标签集(SubCOMP) 不可满足, 那么标识节点 n 为"√", 并删除其父节点标识"√";

for(每一在 n 的右上边标识为"√"的节点 m)

if($S_n \subset S_m$)

　删除节点 m 的标识"√";

修剪的 CSISE-tree T_I 生成以后, 初始冲突 S 的所有极小冲突集即为标识"√"的所有节点的标签集.

①修剪规则的正确性.

第一条修剪规则: 当一个集合是非冲突集时, 任何它的孩子节点的标签集都将是它的真子集, 因而必然不是冲突.

第二条修剪规则: 如果一个节点 n 的子边标签集 E_{c_n} 包含其右上边已产生的某非冲突集 m 的父边标签 E_{p_m}, 则根据 ISE-tree 的生成特点, 节点 n 将会有一个孩子节点, 且其标签集为 $S_n-\{E_{p_m}\}$, 但若 $S_n-\{E_{p_m}\} \subseteq S_m$, 则由于 S_m 为非冲突集, $S_n-\{E_{p_m}\}$ 必然为非冲突集. 因此, 为避免该子节点的生成, 需将节点 n 的子边标签集 E_{c_n} 改为 $E_{c_n}=E_{c_n}-\{E_{p_m}\}$.

②对于标识规则, 若节点 n 为一个冲突集, 则它的父节点肯定不是一个极小的冲突. 另外, 由于同一层从右至左检测, 因此如果在 n 的右上边有节点 m 为极小冲突, 即标识为"√", 则其所有的孩子节点必然不是冲突, 但若 S_m 包含 S_n, 则 S_m 将不再是极小的, 因此删除节点 m 的极小标识.

③完备性: 由于一个完全 ISE-tree 能够按照某种预定的顺序枚举出一个集合的幂集, 因而除了已产生的极小冲突集的真超集, 初始冲突集 S 的所有其它可能的子集都将产生; 所有的极小冲突集也将最终全部产生.

④本方法的时间复杂度及空间复杂度主要都与产生节点的个数相关, 在最坏的情形为 $O(2^k)$, 其中 k 为初始冲突集 S 的长度; 但由于剪枝的存在, 一般情况下复杂度总是小于 $O(2^k)$ 的.

类似的, 还可以考虑 CSSE-tree、Inverse CS-tree 以及 CS-tree with Mark Set 等方法, 结合 IsCS 来判定冲突集从而产生所有的极小冲突集. 所有这四种扩展方法的最坏情况下的复杂度与组件数目成指数级, 即完成所有的扩展. 然而, 一般情况下, 由于加入了各种相应的终止规则使得扩展的数目小很多.

4　启发式策略

倘若对系统的结构进一步分析, 还可以给出几个启发式策略, 以便能够更少地调用 SAT 求解器从而能够更快地求解.

定义 **4**　若一个组件的所有输入端均为系统的输入端, 且输出端不为系统的输出端, 则称该组件为纯输入组件; 若一个组件的输入/输出端均为内部联结点, 则称该部件为内部组件; 若一个组件的输出端为系统的输出端, 且输入端为系统的内部联结点, 则称该部件为纯输出组件; 若一个组件的输入/输出端分别为系统的输入/输出端, 则称该部件为输入/输出组件.

注: 这几个定义都是相对的, 针对特定输入输出来说的. 比如说, 当一个内部组件的输入端进行测量得到观测时, 则使得该内部组件成为纯输入组件, 且以该测量点作为输出端的部件成为输出组件(或者输入/输出组件– 若其原为纯输入组件的话). 另外, 如下为了描述方便, 称一个组件为输入组件是指其为纯输入组件或者输入/输出组件; 类似地, 称一个组件为输出组件, 则该组件为纯输出组件或者输入/输出组件.

命题 **1**　仅纯输入组件集不为冲突集.

证明　若仅考虑纯输入组件, 则仅有输入观测而

无实际输出观测, 则无法判断实际输出值与组件正常时的理论输出值是否一致, 因而总是可满足的.

命题 **2** 仅内部组件集不为冲突集.

证明 若仅考虑内部组件, 则实际输入输出值均不能给出, 因而 SD_C_OBS.cnf 中仅有这些组件的正常行为描述及所有组件的正常谓词—这些必然都是可满足的; 以及所有的与之非直接连接的输入输出点的值—由于非直接相连, 它们并不能影响前面的可满足性结果. 因而仅内部组件不为冲突集.

命题 **3** 仅纯输入、内部组件集不为冲突集.

证明 即使纯输入及内部组件均是连接的, 由于未知实际的输出值因而无法判断实际输出值与组件正常时的理论输出值是否一致, 因而总是可满足的. 若某些部件不通过该部件集中的部件直接相连, 则不能判定是否冲突.

命题 **4** 若一个组件集为冲突集, 则其至少仅为一个输入组件与一个输出组件(它们两个可能为同一个输入/输出组件), 及依次连接它们的内部组件(相应地, 可能不存在)构成的集合 S_1; 或者至少仅含有两个相互连接的纯输出组件及其之间依次连接的组件构成的集合 S_2.

证明 先证前半部分, 假设将 S_1 中输出组件删除, 则如同前一命题所述, 一定不为冲突集; 若将 S_1 中输入组件删除, 则剩余组件由于无法传递由输入值产生的影响因而也不为冲突集; 若将 S_1 中任何一个内部组件删除, 则同样不能传递输入及输出对内部组件的影响, 因而也不为冲突集. 此外, S_1 中由输入组件的输入值通过连接的组件(当输入输出为同一个部件时, 内部组件不存在), 不断地传递直到输出组件可以得到预期的输出值, 而实际的输出已知(在 OBS.cnf 中), 因而 S_1 可能为冲突集.

类似地可证后半部分. 另外, 分别由两个纯输出组件及输出值沿着所连接的组件向前传播, 有可能导致某一内部节点的值出现两个不同的值, 于是 S_2 可能为冲突集.

推论 **1** 若一组件集不含任何输出组件(包括纯输出组件及输入/输出组件), 则肯定不为冲突集.

通过使用启发式策略, 可以减少提取相关系统描述以及调用 SAT 求解器的次数, 从而在一定程度上提高求解效率, 特别是当输出组件个数较少而其它种类组件个数较多时, 更适用这种启发式策略.

5 实验结果及分析

前面提到的4种算法均已实现, 并且为了测试各种方法的性能, 我们进行了大量测试. 实现及测试环境: Dell Dimension C521, Windows XP, VC++ 6.0, MD Athlon (tm) 64 X2 Dual Core Processor 3600+, 1.9GHz, 1022MB RAM. 其中为了判定一个组件集是否为冲突集, 我们使用的是 2007 年国际 SAT 竞赛的冠军求解器 RSAT[19].

我们对基准电路 ISCAS-85 中的 c17, 全加器电路(1 位全加器 Fulladder_1 和 2 位全加器 Fulladder_2), 以及 Polybox 型的电路[2] 进行了测试(5 个组件的 Polybox_5 和 9 个组件的 Polybox_9, 该类型电路来源于文献[2]中的加法器——乘法器电路, 这里我们分别使用与非门(NAND)和或非门(NOR)将原来的加法器和乘法器替换, 目的是能够更容易枚举出所有可能的情形进行测试, 由于组合逻辑电路的输入和输出均为 0 或 1, 如图 3 所示, 其中 NAi 表示与非门, NOi 表示或非门). 首先给出各电路的系统描述文件 SD.cnf, 其次根据输入/输出端所有可能的组合给出所有可能的观测描述文件 OBS.cnf, 然后分别使用前面提到的 4 种方法进行求解可能的极小冲突集, 并记录相关的求解时间和产生的节点的个数, 最后对每种方法每个电路求出平均的求解时间和产生的平均节点数进行汇总如表 1 所示.

注: 表 1 中 A, B, C, D 分别表示 CSISE-tree, CS-tree with Mark Set, CSSE-tree, Inverse CS-tree).

实际采用的启发式策略: 将所有未与任何输出端相连的组件作为一个最大势的不必判断可满足性的集合, 即所有纯输入及内部组件的集合 InandInter. 若待扩展的组件子集 SubCOMP 为集合 InandInter 的子集, 则不必调用 SAT 求解器, 直接判定 SubCOMP 为非冲突集.

为了比较引入启发式策略的方法与基本方法, 我们还另外分别测试了相应的启发式策略方法的平均求解时间和产生的平均节点的个数情况如表 2 所示(注: 表 2 中 A, B, C, D 和表 1 有相同含义).

从结果可以看出, 一般来说, CSISE-tree 方法结合

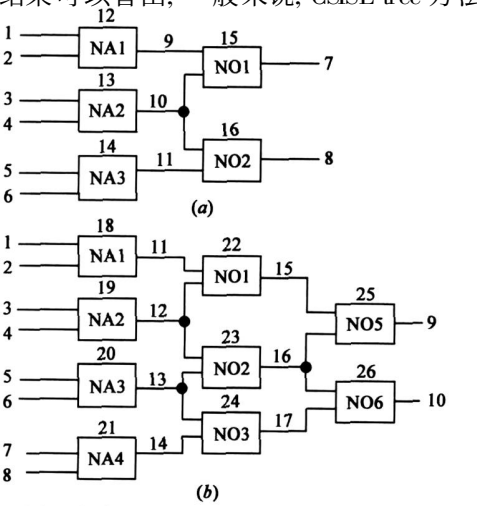

图3 电路Polybox_5 (*a*) 和 Polybox_9 (*b*)

SAT 求解器产生最少的节点数, 具有最好的效率(在 Fulladder_2 时与 CS-tree with Mark Set 方法基本相同, 但其它所有实例均比后者快). 其次为 CS-tree with Mark Set、CSSE-tree、Inverse CS-tree.

表 1　4 种方法的平均的运行时间和节点数

	ISCAS-85_c17		Polybox_5		Polybox_9		Fulladder_1		Fulladder_2	
	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes
A	1.353226	13.320313	1.142517	11.511719	18.020933	121.926758	0.932663	9.062500	21.082430	208.289063
B	2.037873	21.703125	1.492849	16.609375	26.688645	207.944336	1.043067	11.625000	19.611920	210.296875
C	5.795260	30.046875	2.426439	14.046875	60.735135	204.016602	2.748664	14.906250	87.605697	427.578125
D	6.467154	61.500000	3.013111	29.500000	72.807055	480.375000	3.016638	29.000000	93.340621	883.125000

表 2　引入启发式策略后的 4 种方法的平均的运行时间和节点数

	ISCAS-85_c17(h)		Polybox_5(h)		Polybox_9(h)		Fulladder_1(h)		Fulladder_2(h)	
	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes	ave time(s)	ave nodes
A	1.293886	13.320313	1.130418	11.511719	13.600884	121.926758	0.870035	9.0625000	20.836117	208.289063
B	1.789390	21.703125	1.266185	16.609375	16.555384	207.944336	0.869695	11.625000	18.211042	210.296875
C	4.146011	30.046875	1.736941	14.046875	31.324691	204.016602	1.950766	14.906250	73.346123	427.578125
D	4.710413	61.500000	2.228533	29.500000	40.296815	480.375000	2.147849	29.000000	78.916140	883.125000

此外, 对比表 2 和表 1 还可以看出, 引入启发式策略后所用时间比未使用启发式策略时, 所花费时间减少平均约为 20.98%, 最大为 48.42% (Polybox_9 中的 CS-tree with Mark Set 方法), 这主要是由于启发式策略能够避免较多地提取相关子组件集的系统描述及观测的 CNF 文件以及因而调用 SAT 求解器的原因. 图 4 更形象地说明了 4 种方法未引入启发式及使用启发式信息之后的效率比较.

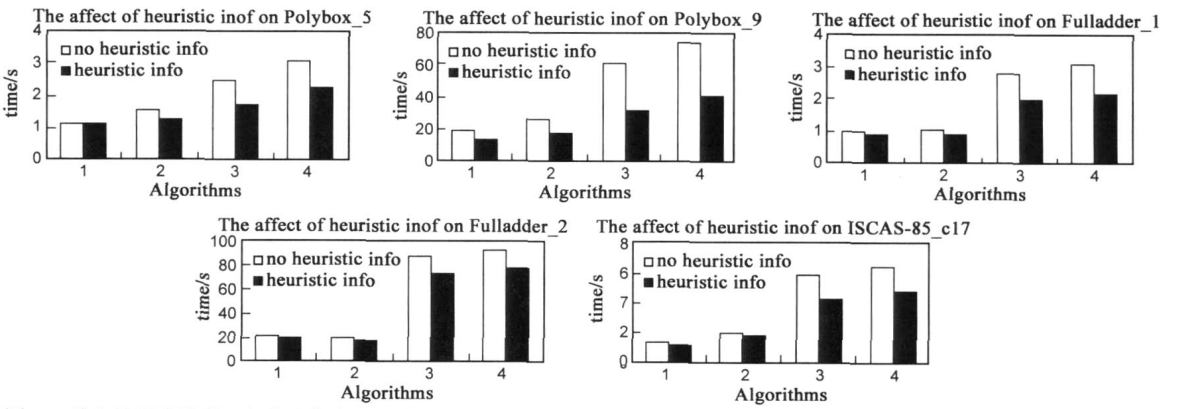

图4　4种方法效率比较 (启发式信息对效率的影响): 1, 2, 3, 4 分别表示CSISE-tree, CS-tree with Mark Set, CSSE-tree, 和Inverse CS-tree 方法

另外, 从实验结果中还发现各算法的效率与冲突集长度的关系: 所产生的冲突集长度越大, 使用 CSISE-tree 和 CS-tree with Mark Set 方法较好, 这是由于它们这两种方法首先判定较长的部件集是否为冲突, 若为冲突则可以避免其真子集的扩展; 相应地, 若产生的冲突集长度越小, 则使用 CSSE-tree 和 inverse CS-tree 较好, 因为它们较早判定长度较短的组件集, 若为冲突则可避免其真超集的扩展.

6　讨论

在 MBD 领域中, 较早时候主要使用基于 ATMS 的方法求解 MCSs, 但是由于基于一致性诊断中仅含有正常行为模型, 使得 ATMS 求解所有 MCSs 可能不完备, 从而使得诊断结果可能不完备甚至不正确; 并且使用 ATMS 需要计算复杂的支撑环境等信息使得效率不够高. 本文所提出的方法不必计算支撑环境, 并能产生所有的 MCSs, 具有完备性.

早期的定理证明器(如 DART)等, 效率较低, 难以应用. 而 SAT 求解器最近几年却得到如火如荼的发展, 每年一次的国际 SAT 求解器竞赛大大促进了 SAT 求解器的研究, 使得 SAT 求解的效率不断提高. 正因如此, 只需采用最快的 SAT 求解器, 我们的求解方法效率也会不断提高. 并且, 由于 SAT 方法的广泛通用性使得我们的方法同样具有较好通用性.

7　结论

本文提出使用 SAT 求解器来判定极小冲突集的方法, 并使用 CSISE-tree 等方法产生所有极小冲突集. 为进一步提高效率, 讨论了启发式策略. 实验结果表明,

CSISE-tree 方法产生最少的节点数,一般能取得最好的效率;并且启发式策略也确实进一步地提高了求解效率.

充分利用系统的相关的结构信息(主要是连接关系)以及系统组件行为之间的制约关系,提供更多的启发信息,可以作进一步研究.

分层方法也在 MBD 领域中得到了广泛的应用,通过对系统分层,使得各不相关部分独立(或并行)求解,然后合并,从而提高效率,各子部分独立求解还可以缩小求解空间,增大可扩展性.这也可作为下一步研究求解 MCSs 的一个方案.

参考文献:

[1] L Console, O Dressler. Model-based diagnosis in the real world: lessons learned and challenges remaining[A]. In Proceedings of 16th International Joint Conference on Artificial Intelligence(IJCAI-99)[C]. Stockholm, Sweden, 1999. 1393-1400.

[2] R Reiter. A theory of diagnosis from first principles[J]. Artificial Intelligence, 1987, 32(1): 57-96.

[3] J de Kleer. Local methods for localizing faults in electronic circuits[M]. Cambridge, MA, MIT AI Memo, 1976. 394.

[4] J de Kleer. An assumption-based tms[J]. Artificial Intelligence, 1986, 28(2): 127-162.

[5] J de Kleer. Problem solving with the ATMS[J]. Artificial Intelligence, 1986, 28(2): 197-224.

[6] M R Genesereth. The use of design descriptions in automated diagnosis[J]. Artificial Intelligence, 1984, 24(1-3): 411-436.

[7] R Haenni. A query-driven anytime algorithm for argumentative and abduction[A]. In Proceedings of 17th National Conference on Artificial Intelligence(AAAI-00)[C]. Texas, 2000. 337-342.

[8] 栾尚敏,戴国忠. 利用结构信息的故障诊断方法[J]. 计算机学报, 2005, 28(5): 801-808.
Luan Shang-min, Dai Guo-zhong. An approach of diagnosing a system with structure information[J]. Chinese Journal of Computers, 2005, 28(5): 801-808. (in Chinese)

[9] 代树武,孙辉先. 基于模型故障诊断中的冲突求解[J]. 控制理论与应用, 2003, 20(4): 630-632.
Dai Shu-wu, Sun Hui-xian. Computing conflict sets for model-based diagnosis[J]. Control Theory & Applications, 2003, 20(4): 630-632. (in Chinese)

[10] 方敏. 一种识别最小冲突集的实用方法[J]. 合肥工业大学学报(自然科学版), 1999, 22(1): 39-43.
Fang Min. A practical method to identify the minimal conflict sets[J]. Journal of Hefei University of Technology(Natural Science Edition), 1999, 22(1): 39-43. (in Chinese)

[11] A Hou. A theory of measurement in diagnosis from first principles[J]. Artificial Intelligence, 1994, 65(2): 281-328.

[12] B Han, S J Lee. Deriving minimal conflict sets by CS-trees with mark set in diagnosis from first principles[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1999, 29(2): 281-286.

[13] X Zhao, D Ouyang. Improved algorithms for deriving all minimal conflict sets in model-based diagnosis[A]. Lecture Notes in Computer Science[C]. Springer Berlin/Heidelberg, 2007, 4681, 157-166.

[14] L Lin, Y Jiang. The computation of hitting sets: Review and new algorithms[J]. Information Processing Letters, 2003, 86(4): 177-184.

[15] X Zhao, D Ouyang. A method of combining SE-tree to compute all minimal hitting sets[J]. Progress in Natural Science, 2006, 16(2): 169-174.

[16] 严晓浪,郑飞君,葛海通,杨 军. 结合二叉判决图和布尔可满足性的等价性验证算法[J]. 电子学报, 2004, 32(8): 1233-1235.
Yan Xiao-lang, Zheng Fei-jun, Ge Hai-tong, Yang Jun. Combining Binary Decision Diagrams and Boolean Satisfiability for Equivalence Checking[J]. Acta Electronica Sinica, 2004, 32(8): 1233-1235. (in Chinese)

[17] J Rintanen, A Grastien. Diagnosability Testing with Satisfiability Algorithms[A]. In Proceedings of 20th International Joint Conference on Artificial Intelligence(IJCAI-07)[C]. Hyderabad, India, 2007. 532-537.

[18] R Rymon. Search through systematic set enumeration[A]. In Proceedings of 3rd International Conference on Principles of Knowledge Representation and Reasoning(KR-92)[C]. Cambridge, MA, 1992. 539-550.

[19] Rsat, UCLA Automated Reasoning Group[OL]. http://reasoning.cs.ucla.edu/rsat/

作者简介:

赵相福 男,1981 年生于山东汶上,吉林大学博士研究生,主要研究方向为基于模型的诊断.
E-mail: xiangfuzhao@gmail.com

欧阳丹彤 女,1968 年生于吉林长春,博士、吉林大学教授、博士生导师,主要研究方向为基于模型的诊断、自动推理和模型检测.
E-mail: ouyangdantong@163.com