·计算机工程与应用·

基于满足性判定的布尔网络环求解算法

郭文生[1]，杨国武[1]，李晓瑜[1]，高 敏[2]

(1. 电子科技大学计算机科学与工程学院 成都 610054；2. 加州大学洛杉矶分校电子工程学院 美国 加州 洛杉矶 CA 90034)

【摘要】环是布尔网络状态转换过程中的稳定态，在模式检测、基因调控网络和可达性分析等领域都有重要的意义。计算布尔网络状态转换中的所有环是一个NP完全问题。该文基于全解布尔满足性判定(SAT)算法，设计了一种求解所有小于等于指定步长环的算法。算法基于布尔网络的状态转换函数和状态环属性生成合取范式形式(CNF)的问题集，通过融合冲突子句学习(CDCL)、非时序回退、阻塞子句和变量分类等技术，降低算法的计算复杂度。实验结果表明，该算法能够高效地计算指定步长的环。对于无法计算所有环的复杂网络，指定步长计算环的方式将更有应用价值。

关 键 词 布尔网络；环；满足性判定；阻塞子句

中图分类号 TP301.6 **文献标志码** A doi:10.3969/j.issn.1001-0548.2015.06.015

SAT-Based Algorithm for Finding Cycles in a Boolean Network

GUO Wen-sheng[1], YANG Guo-wu[1], LI Xiao-yu[1], and GAO Min[2]

(1. School of Computer Science and Technology, University of Electronic Science and Technology of China Chengdu 610054;

2. School of Electronic Engineering, University of California-Los Angeles Los Angeles Califonia USA 90034)

Abstract A cycle which is a steady state in Boolean network state transition is important in modeling checking, genetic regulatory networks and reachability analysis. Obtaining all the cycles in Boolean network state transition is a NP complete problem. In this paper, we proposes an algorithm for solving all the cycles which have less than or equal to the assigned step length based on all-solution Boolean satisfiability (SAT) algorithm. By extending the state transition function of a Boolean network and the property of cycles, this algorithm creates a problem set in conjunctive normal form (CNF) and incorporates techniques such as conflict-driven clause learning (CDCL), non-chronological backtracking, blocking clause and variable classification to decrease computational complexity. Experiment result shows that this algorithm is efficient for solving the cycles. For large scale complex network that is hard to get all the cycles, this algorithm delivers more practical value.

Key words blocking clause; boolean networks; boolean satisfiability; cycles

布尔网络的稳定状态包括两种：不动点和环。不动点是环的一种特例，即环步长为1。布尔网络环的计算是一个NP完全问题[1-2]。近年来，随着布尔满足性判定SAT算法的不断优化，越来越多的布尔网络问题都可以用SAT算法求解，如无边界的符号模式检测、可达性分析、量词消除、基因调控网络求环等[3-8]，但对于复杂布尔网络的所有环的求解仍然具有非常高的计算复杂度。本文的目标是基于SAT全解算法计算所有小于指定步长的环。

1 相关技术

1.1 基本术语

使用SAT算法求解，绝大部分应用都需要转换成通用的SAT问题描述形式，即CNF范式。CNF范式由所有子句的合取形式构成，其中每个子句是若干文字(变量或非变量)的析取。每个文字是变量本身或者是变量的非。CNF范式的格式为：

$$F=\bigwedge_{i=1}^{N}C_i, C_i=l_1\vee l_2\vee\cdots\vee l_k \tag{1}$$

式中，F表示要求解的满足性问题；C_i表示子句；N表示范式中包含的子句的数量；l_k表示文字，$l_k\in\{v,\neg v\}$(v是一个变量)；k表示子句中包含文字的数量($k\leqslant n$，n为变量的数量)。

1.2 基于冲突子句学习和非时序回退的DPLL算法

Davis-Putnam-Longeman-Loveland(DPLL) 算法是完全SAT求解算法的基本架构。几乎它是所有现

收稿日期：2014－05－09；修回日期：2014－09－12

基金项目：国家自然科学基金(61272175)

作者简介：郭文生(1976－)，男，博士生，副教授，主要从事形式化方法和软件测试技术方面的研究.

代的完全求解算法的基础，包括单文字子句消除、肯定-否定规则、原子公式消除等boolean constraint propagation(BCP)规则。而CDCL和非时序回退的技术能够有效的降低搜索空间，进而降低SAT算法的时间和空间复杂度[9]。

BCP算法实现子句单元广播，即根据已赋值的变量推导未赋值的变量值。如果BCP推导时出现冲突，则调用CDCL算法进行冲突学习并生成冲突子句。冲突子句可以避免重复的路径搜索。CDCL算法基于蕴含图(implication graph)技术实现冲突子句的学习和非时序的决策级回退。蕴含图描述在搜索过程中的部分变量赋值及蕴含关系，选择不同的决策变量选择和不同的BCP顺序将生成不同的蕴含图，因此搜索过程中蕴含图是动态变化的。

1.3 基于单解求解器的全解求解算法

计算满足性判定问题F的所有满足解，可以基于单解求解器通过如下步骤实现：

1) 判断F是否满足。若不满足，则输出不满足；否则，计算一个满足解$x_1,x_2,\cdots,x_n$，n是问题中的变量数。

2) 用v_i表示满足解中的一个变量赋值。若解$x_i=0$，则对应的变量表示为$\neg v_i$；否则表示为v_i。因此，一个满足解可以表示为由变量或变量的非形成的一个合取子句s。

3) 将满足解形成的合取子句取反，即所有变量取反并形成一个析取子句c。

4) 将析取子句c加入到求解的问题集F，形成新的问题集$F=F\wedge c$。

5) 判断新的问题集F'是否存在满足解。若存在，则转到步骤2)；否则输出所有已计算的满足解。

上述求全解的过程存在如下问题：

1) 每一次满足解的计算都是以整个问题为基础，对前一次求解过程中的搜索空间没有记录。

2) 没有根据求全解的特点进行搜索算法优化。

针对上述问题，文献[3]提出了阻塞子句(blocking clause)的方式。当单解SAT求解器获得一个满足解时，并不停止求解过程，而是将该满足解对应的变量合取表示方式取反形成一个析取子句作为阻塞子句加到问题集中，继续求解过程。通过这种方式可以保留前一次求解过程中所学习的冲突子句，并不再搜索已经搜索过的路径空间。然而，由于满足性问题的满足解可能非常多，从而导致阻塞子句需要较大的内存空间，并且大量的阻塞子句也会增加问题的求解复杂度。文献[4]为降低阻塞子句的数量，使用子句聚合方式对产生的阻塞子句集进行蕴含化简，缩短阻塞子句的长度，进而降低阻塞子句的数量。文献[10]提出了基于算法避免再次获得重复满足解的方式，降低内存开销并保持求解复杂度不变。它没有使用阻塞子句的方式避免计算重复解，而是基于决策变量与求得解的蕴含关系生成冲突子句，进而避免后续的计算出现重复的解。这种生成冲突子句的方式在本质上等同于生成阻塞子句的方式。同时，它提出了求解问题中变量划分的方法，它将要求解问题中的变量划分成两类：重要变量集和不重要变量集。在搜索过程中先选择重要变量进行决策赋值，当重要变量全部赋值完成后才对不重要变量进行决策赋值。然而该算法是基于CNF范式的求解，其重要变量和非重要变量没有结合具体问题进行定义。文献[11]为降低阻塞子句的数量并降低后续计算的搜索空间使用最短阻塞子句的方式计算全解，它基于覆盖算法最小化满足片段减少阻塞子句中的文字数，但该算法使用的覆盖算法本身复杂度也比较高，对于问题子句数和变量数多的求解问题计算复杂度较大。

上述求全解算法只考虑了如何避免已求得的满足解的问题，而计算不动点、基因调控网络吸引区求解等问题在计算过程中不能仅避免已求得的满足解，还需要避免已经计算的不动点或状态环。

2 布尔网络环求解算法

2.1 基于状态转换函数和环属性创建求解问题

布尔网络描述的是变量之间的相互依赖关系，每个变量的值只能是0或1。变量的下一个时刻的值由依赖节点的当前时刻的值确定，即：

$$v_i'=f_i(v_1,v_2,\cdots,v_m) \tag{2}$$

式中，v_i'表示节点i的下一个时刻的状态值；v_1，$v_2,\cdots,v_m$表示节点i依赖的m个节点的当前时刻的值。

布尔网络的环是布尔网络状态转换过程中的稳定状态。如果环的步长为N，则布尔网络至少需要经过N个时刻的状态迁移才能进入环状态。因此，根据布尔网络的状态转换函数，可以通过N步展开的方式生成包含$N+1$个布尔网络状态的转换路径，即：

$$\text{path}=\bigwedge_{k=1}^{N}T_k(s_{k-1},s_k) \tag{3}$$

式中，path表示布尔网络状态转换图中的一条路径；s_0表示网络的初始状态；$s_k=v_{k1},v_{k2},\cdots,v_{kn}$表示第$k$时刻网络的状态。

同时，步长小于等于N的所有环的属性p可以描

述成为：

$$p = \bigvee_{k=0}^{N}(s_N = s_{N-k}) \tag{4}$$

根据式(2)～式(4)可以得到计算步长小于等于N的环的求解问题集，并可以通过引入辅助变量的方式将其转换成CNF形式。

2.2 基于SAT全解算法的环计算

基于阻塞子句、冲突蕴含图生成阻塞子句和变量分类等SAT全解算法的实现技术，结合布尔网络转换关系和环属性描述方式，本文提出了求解步长小于等于N的环的求解算法。其算法的基本流程为：

Input: CNF范式F，变量集V，环步长N；

Output：全部满足解all-solution或不满足UNSAT；

初始化:

L=NULL; //L是确定赋值的文字集合；

DecisionLevel = 0; //DecisionLevel是搜索的深度；

Status = SAT;

生成CNF问题集 F ：

F = transtoCNF($(\bigwedge_{i=1}^{N} T(s_{i-1}, s_i)) \wedge (\bigvee_{k=0}^{N}(s_N = s_{N-k}))$) setDecisionVars(F);

IF (BCP(F ,L)==CONFLICT) THEN Return UNSAT；

WHILE (Status == SAT)

Begin

preClauseCount = sizeof(F); //获得原始问题的子句数量；

WHILE (sizeof(L) < sizeof(V)) //sizeof()获得变量的数量；

Begin

l =PickBranchLit(F ,V\Var(L)); //从未赋值的变量中选择一个变量赋值，Var(L)是已赋值的变量集合；

Add(l , L); DecisionLevel++; //增加赋值变量到解集L；

IF (BCP(F , L) == CONFLICT)

Begin

DecisionLevel = CDCL(F , L);

If (DecisionLevel < curDecisionLevel)

Begin

removeClausesFrom(preClauseCount);

c =GenBlockClause(D);

$F = F \wedge c$;

End

IF (DecisionLevel==0) Status = UNSAT, Break;

ELSE BackJump(DecisionLevel);

End

End

curDecisionLevel = DecisionLevel;

$s = L$;//获得一个满足解；

c =GenBlockClause(D);

$F = F \wedge c$;

BackJump(DecisionLevel − 1);

End

IF (Sizeof(All-Solution) > 0) Return All-Solution;

ELSE Return UNSAT;

PickBranchLit( )函数实现决策变量的选择并为其赋值。根据转换CNF过程中存在大量辅助变量的特性，选择决策变量时采用了变量分类的方式减少决策变量的选择空间。setDecisionVars( )函数用于设置变量的属性，即该变量是否可以作为决策变量。GenBlockClause( )函数根据当前的决策变量赋值获得阻塞子句。参数D表示决策级为k时的所有决策变量赋值对应的文字集合 $D = \{d_1, d_2, \cdots, d_k\}$，则GenBlockClause( )函数生成的阻塞子句为 $c = \neg d_1 \vee \neg d_2 \vee \cdots \vee \neg d_k$。

2.3 基于变量分类的决策文字选择优化

为提高搜索速度并降低搜索空间，本文将待选择变量分为两类：决策变量和蕴含变量。决策变量是布尔网络状态转换过程中对应的状态变量，蕴含变量是布尔网络转换成CNF范式过程中引入的辅助变量。本文中布尔网络的描述形式采用的是类似于berkeley logic interchange format (BLIF)的格式，转换关系示例如表1所示。

表1　布尔网络转换关系描述示例

类BLIF文件格式	说明
.v 3	.v指示当前布尔网络的节点数
.n 1 0	.n指示节点之间的依赖关系，.n后面的第一个数字表示节点号，第二个数字表示节点依赖的其他节点数，后面部分给出的是所有依赖节点号。.n下面的行描述所有可以使节点的下一个状态值为真的依赖节点赋值序列。其他所有依赖节点的赋值序列都会导致节点的下一个状态赋值为假。
.n 2 2 1 3	
0 1 1	
1 0 1	
.n 3 2 1 2	
0 0 1	
1 1 1	

表1能够准确的描述布尔网络节点之间的依赖关系，并根据该依赖关系得到对应的布尔网络状态转换函数。在同步布尔网络中，给定一个布尔网络的转换步数N后，可以根据转换关系展开得到布尔网络的状态转换图，并将其转换成CNF形式。然而在转换过程中将会引入大量的辅助变量。表1中的每一个转换定义项都需要引入一个辅助变量。下面以表1

中节点2的下一个状态转换函数为例描述转换过程：

1) 若节点的当前状态为1，则表示为 $v_i(i\in\{1,2,3\})$，否则表示为 $\neg v_i$。

2) 若节点的下一个状态为1，则表示为 $v_i'(i\in\{1,2,3\})$，否则表示为 $\neg v_i'$。

3) 根据表1中(.n 2 2 1 3)的描述，其转换函数 $v_2'=f(v_1,v_3)$，同时其后的两行分别表示 $v_2'=\neg v_1\wedge v_3=1$ 和 $v_2'=v_1\wedge\neg v_3=1$，即节点2的转换关系函数可表示为：

$$v_2'=(\neg v_1\wedge v_3)\vee(v_1\wedge\neg v_3) \tag{5}$$

4) 针对转换关系函数中的每一个合取项引入一个辅助变量 y_1,y_2，并令 $y_1=\neg v_1\wedge v_3, y_2=v_1\wedge\neg v_3$，则节点2的转换关系函数可表示为：

$$v_3'=$$
$$(y_1=\neg v_1\wedge v_3)\wedge(y_2=v_1\wedge\neg v_3)\wedge(v_2'=y_1\vee y_2) \tag{6}$$

5)步骤4)的转换关系转换成CNF范式的子句集为：

$$\begin{aligned}v_4'=&(\neg y_1\vee\neg v_1)\wedge(\neg y_1\vee v_3)\wedge(y_1\vee v_1\vee\neg v_3)\wedge\\&(\neg y_2\vee v_1)\wedge(\neg y_2\vee\neg v_3)\wedge(y_2\vee\neg v_1\vee v_3)\wedge\\&(\neg y_1\vee v_2')\wedge(\neg y_2\vee v_2')\wedge(y_1\vee y_2\vee\neg v_2')\end{aligned} \tag{7}$$

根据上述转换过程可以看出，辅助变量的赋值是由节点状态值所蕴含的，因此在SAT求解过程中只要确定了节点状态值，则辅助变量的赋值能够根据蕴含关系确定，因此在PickBranchLit( )函数选择决策变量的过程中不需要选择辅助变量作为决策变量，进而降低SAT求解的搜索空间。

2.4 基于蕴含图的决策变量生成阻塞子句优化

根据计算全解和增加子句的过程，可将要求解的问题分成两部分：原始的问题 F 和阻塞子句集 B。SAT问题 F 的一个满足解 $s_i(x_1,x_2,\cdots,x_n)$ 可以用合取子句 $c_i=l_1\wedge l_2\wedge\cdots\wedge l_n$ 表示，如果 $x_i=1$，则子句中对应的文字 $l_i=v_i$；否则 $l_i=\neg v_i$。因此，所有已获得的 m 个满足解的集合将形成一个析取disjunctive normal form(DNF)范式，其格式为：

$$S=\bigvee_{i=1}^{m}c_i \tag{8}$$

基于满足解集，可以得到阻塞子句集 $B=\neg S$。在计算是否存在 $m+1$ 个满足解时，需要将 B 加入到问题集 F 中，即 $F=F\wedge B$，以避免获得重复的解。显然，S 中包含的子句越多且每个子句中的文字数越多，则计算 $m+1$ 个满足解的时间和空间复杂度越大。

优化的目标之一是降低阻塞子句中文字的数量。冲突子句学习的SAT求解算法会在求解过程中基于选择的决策变量进行蕴含推理，进而生成局部蕴含图。蕴含图示例如图1所示。

图1中待求解的CNF范式由4个子句构成 $F=C_1\wedge C_2\wedge C_3\wedge C_4$。$x_i=a@l$ 表示决策级为 l 时 x_i 的值为 a。子句数据库为 $C_1=x_1\vee x_2$，$C_2=\neg x_2\vee\neg x_3$，$C_3=x_3\vee x_4\vee x_5$，$C_4=\neg x_5\vee x_6$。决策级为1时，决策变量及分配的值为{x_1=0@1}，蕴含分配的值为{x_2=1@1, x_3=0@1}。决策级为2，决策变量及分配的值为{x_4=0@2}，蕴含分配的值为{x_5=1, x_6=1}。最后获得可满足解{x_1=0, x_2=1,x_3=0, x_4=0, x_5=1, x_6=1}。

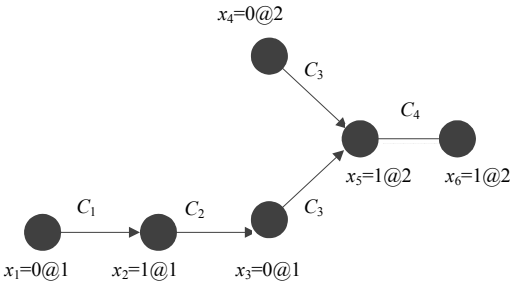

图1 SAT求解过程中的蕴含图示例

根据图1可以看到，当前的可满足解 $s=\{\neg v_1,v_2,\neg v_3,\neg v_4,v_5,v_6\}$ 是由两个决策变量的赋值 $d=\{x_1=0@1,x_4=0@2\}$ 确定的。因此，为避免下一次计算出现重复的解，则生成的阻塞子句是 $\neg v_1\wedge\neg v_4$ 的非，即 $v_1\vee v_4$。显然，使用决策变量生成阻塞子句的方式极大的降低了阻塞子句中文字的数量。

优化目标之二是减少阻塞子句的数量。在基于冲突子句的SAT求解过程中，当获得一个满足解时，采用时序回退的算法计算下一个满足解。即全解搜索方式采用深度优先搜索算法。在深度优先搜索时，如果冲突回退的决策级比当前的决策级小，则表明当前决策级之下已经不存在满足解，则可以删除当前产生的所有阻塞子句，并基于回退的决策级backlevel生成一个阻塞子句 $c=\neg d_1\vee\neg d_2\vee\cdots\vee\neg d_{\text{backlevel}}$，表示backlevel之后的所有满足解都已经被计算，并使用一个阻塞子句 c 避免所有搜索过的满足解空间。

3 实验结果分析

基于Minisat求解器[12]实现了不动点和步长小于等于 K 的周期性环的SAT全解算法period cycles algorithm based on decision node and important variable(PC-DI)。并选择了一个基于SAT计算所有环的求解器BNS[5]作为比较工具。实验使用与文献[5]

的测试案例和*N-K*随机布尔网络作为测试基准。*N-K*网络使用R环境中的BoolNet包随机生成8个最大环长不超过100的布尔网络。测试平台的配置为Intel Xeon CPU@3.3 GHz 8-Core(测试时只用了一个核)，内存为128 GB，操作系统是Ubuntu 12.04。

实验分析了两方面的性能：所有周期性环求解性能和指定步长周期性环求解性能。

为了使用本文的算法计算所有的周期环，首先使用BNS计算出所有环并获得最大的环长，然后以最大环长为指定步长运行本文的求解器，测试结果如表2所示。

表2　所有环求解计算结果

布尔网络	节点数量	环的最大步长	BNS/s	PC-DI/s
Arabidopsis	15	1	0.005	0.002
Bud yeast	12	1	0.012	0.002
Drosophila	52	1	0.057	0.002
Fission	10	1	0.005	0.002
Mammalian	10	7	0.004	0.005
T-helper	23	1	0.005	0.002
T-cell-r	40	6	0.011	0.007
Random1	200	25	13.204	8.52
Random2	200	60	268.231	216.687
Random3	400	48	12.565	5.176
Random4	400	57	675.413	36.719
Random5	600	16	8.434	1.302
Random6	600	66	587.873	484.624
Random7	800	40	5.738	3.202
Random8	800	36	287.427	22.721

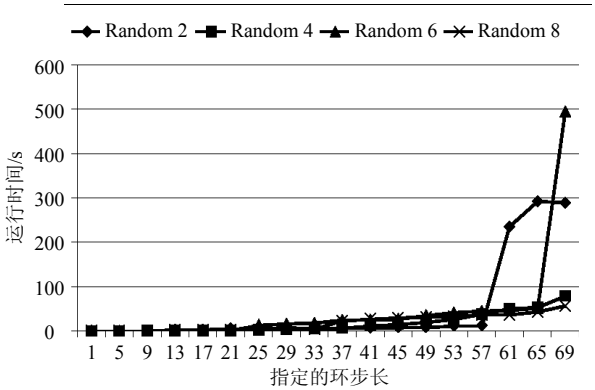

图2　指定步长环的运行时结果

实验结果显示本文算法在计算环的最大步长小于100的真实案例和随机案例中都具有较大的性能优势。

基于表2的测试结果，选择其中4个BNS计算时间在100 s以上的案例(Random2, Random4, Random6, Random8)进行指定步长环计算的分析。对每一个测试案例指定不同的步长进行测试。指定的步长初始值为1，每次增加4个步长，最大指定步长设置为69(大于所有测试案例的环的最大步长)。测试结果如图2和图3所示。

图2比较的是指定最大环步长时计算所有周期环的时间，图3比较的是指定最大环步长时计算出的周期环的数量。Random2、Random4、Random6和Random8的所有周期环分别是5 632个、24个、2 280个和478个。图2显示Random4和Random8的运行时间增加比较平缓，Random2和Random6分别在指定步长为61和69时运行时间增加较大。结合图3可以看出，Random2和Random6分别在指定步长为61和69时周期性环数显著增加。

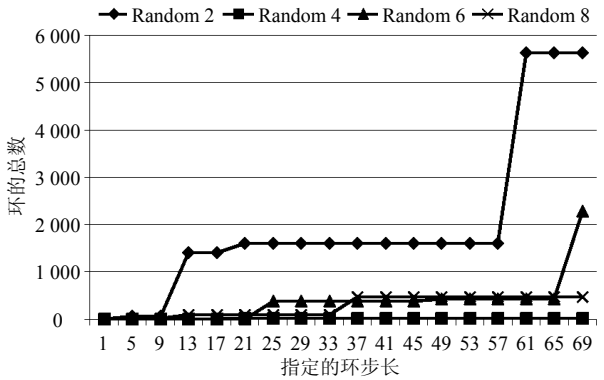

图3　小于等于指定步长环的总环数

实验结果显示，本文算法对于环长分布比较均匀的布尔网络具有较低的计算复杂度，并且指定的步长与运行时间成线性增加的关系。同时，对于运行时间较长且环长分布不均匀的案例(步长大的环占所有环的比例较高)，其主要的时间开销集中在步长大的周期环计算过程中。

基于实验结果可以看到，当指定的步长小于最大状态环步长时，算法计算复杂度较低。因此本文算法可以通过指定步长的方式求解更复杂的布尔网络的状态环。对于无法在有限的时间内计算所有周期环的案例，本文将具有更大的优势。

4　结 束 语

本文提出了一种高效的周期性状态环求解算法。为降低决策变量的选择空间提升搜索速度，该算法将布尔网络转换成CNF过程中引入的辅助变量标记为非决策变量。通过引入基于蕴含图技术建立阻塞子句的方式，该算法能够降低阻塞子句的数量并实现非时序题集的方式，可以充分利用SAT全解求解算法计算状态环。实验结果表明，该算法具有较低的计算复杂度，并且能够对复杂的网络进行指定步长的状态环求解。

参 考 文 献

[1] ZHAO Q. A remark on "scalar equations for synchronous boolean networks with biological applications" by C. Farrow, J. Heidel, J. Maloney, and J. Rogers[J]. IEEE Transactions on Neural Networks, 2005, 16(6): 1715-1716.

[2] T AKUTSU, S KOSUB, A A MELKMAN, et al. Finding a periodic attractor of a Boolean network[J]. Transactions on Computational Biology and Bioinformatics, 2012, 9(5): 1410-1421.

[3] MCMILLAN K L. Applying SAT methods in unbounded symbolic model checking[C]//14th International Conference on Computer Aided Verification. Denmark: Springer, 2002.

[4] CHAUHAN P, CLARKE E M, KROENING D. Using SAT based image computation for reachability analysis[M]. The Netherlands: Springer, 2003.

[5] DUBROVA E, TESLENKO M. A SAT-based algorithm for finding attractors in synchronous Boolean networks[J]. Transactions on Computational Biology and Bioinformatics (TCBB), 2011, 8(5): 1393-1399.

[6] ZHENG De-sheng, YANG Guo-wu, LI Xiao-yu, et al. An efficient algorithm for computing attractors of synchronous and asynchronous Boolean networks[EB/OL]. [2014-01-01] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0060593.

[7] GUO W, YANG G, WU W, et al. A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks[EB/OL]. [2014-01-03] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0094258.

[8] BRAUER J, KING A, KRIENER J. Existential quantification as incremental SAT[C]//23rd International Conference on Computer Aided Verification. Snowbird UT , USA: Springer, 2011.

[9] LIMA J F. Boolean satisfiability solvers: Techniques, implementations and analysis[J]. Electrónica e Telecomunicações S. A., 2013, 5(2): 218-225.

[10] GRUMBERG O, SCHUSTER A, YADGAR A. Memory efficient all-solutions SAT solver and its application for reachability analysis[C]//5th international confrence on Formal Methods in Computer-Aided Design. Texas, USA: Springer, 2004.

[11] YU Y, SUBRAMANYAN P, TSISKARIDZE N, MALIK S. All-SAT using minimal blocking clauses[C]//27th International Conference on VLSI Design and 13th International Conference on Embedded Systems. Mumbai, India: IEEE, 2014.

[12] EEN N, SORENSSON N. Minisat 2.2.0[CP/DK]. [2014-02-02]. http://minisat.se/downloads/minisat-2.2.0.tar.gz.

编 辑 叶 芳