改进的布尔公式学习算法①

哈晓琳[1,2], 李勇坚[1]

[1](中国科学院软件研究所 计算机科学国家重点实验室, 北京 100190)
[2](中国科学院大学, 北京 100190)

摘 要: 当前, 布尔公式学习算法的研究大多数是理论上的模型建立和推导, 很少有人考虑到布尔公式学习算法在实际应用中的效率改进. 现在较成熟的布尔学习算法主要利用的是询问模型, 而询问模型需要依赖外部的 SMT 工具进行询问问题的回答. 虽然, 布尔公式学习算法可以在多项式次数的询问之后得到正确结果, 但是, 减少询问的次数可以减少使用 SMT 工具进行问题计算的次数, 即减少问题计算的时间. 主要针对布尔公式学习算法在实际系统中的应用问题, 提出了利用单调理论中的最小赋值向量的方法, 来减少布尔公式学习算法的询问次数, 提高算法效率和适用性.

关键词: 布尔公式; 学习算法; 询问模型; 单调理论; 最小赋值向量

Improved Learning Algorithm of Boolean Formula

HA Xiao-Lin[1,2], LI Yong-Jian[1]

[1](State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190)
[2](University of Chinese Academy of Sciences, Beijing 100190)

Abstract: Currently, most of the study of Boolean formula learning is theoretically the modeling and derivation. Less people considers efficiency improvements in practical applications. Now more mature Boolean formula learning algorithms mainly use the query model, and the query model relies on external SMT tools to answer the query problems. Although Boolean formula learning algorithm can get the right result after a polynomial number of queries, reducing the number of queries can reduce the number of using SMT tools to answer the queries, namely to reduce the computing time of the queries. This paper took the smallest monotonous assignment vector of the monotone theory to reduce the number of queries in Boolean formula learning algorithm and improved the efficiency of the learning algorithm.

Key words: boolean formula; learning algorithm; query model; monotone theory; minterms

布尔公式学习算法是用某种计算模型推测布尔公式的表达方式的算法, 表达方式有 CNF, DNF, 决策树, 多元多项式等. 虽然目前学术界解决布尔公式学习算法的模型和表达方式各有不同, 但一般都是通过对输入的布尔变量进行某种方式的组合来得到目标公式.

利用成员询问和非限制性等价询问的模型来准确地学习任何布尔公式, 曾经是机器学习中的一个开放问题. 而这种询问模型[1]学习布尔公式的形式, 已经在理论上证明了可行性, 在实际中也有所应用. 当目标公式未知, 但是, 目标公式所满足的一些条件能够给出时, 这种询问模型学习布尔公式的算法可以学习出满足所有已知条件的布尔公式.

询问模型自动学习的概念提出以来, 在理论领域已应用到规范合成, 自动组合验证的理论推导. 1995 年 Bshouty N H 提出了具体的能自动学习任何布尔公式的 CDNF 算法[2]模型, 并在理论上证明了算法的可行性. 而布尔公式学习算法的研究大多数也是理论上的模型建立和推导.

① 基金项目:国家自然科学基金(61271365)
收稿时间:2014-01-09;收到修改稿时间:2014-02-21

直到2010年, Kong S 等人将 CDNF 算法实际应用到规律模型检测领域, 如自动推导循环不变式系统[3,4], 程序终止性证明系统[6]等, 显示出了 CDNF 布尔公式学习算法, 作为一种隐式算法, 可以解决传统显示算法无法解决的有一定困难的问题, 并能够极大的提高多种验证技巧的性能. Wang B Y 等人也将 CDNF 算法实现成工具[5]. 至此, 较少人考虑到布尔公式学习算法在实际应用系统中的效率改进. 本文中提出了一种改进 CDNF 算法的方法, 利用最小赋值向量的方法, 处理 SMT 工具返回的反例, 尽量减少算法中需要的询问次数, 从而加快算法学习的过程, 提高算法的效率.

1 研究基础

1.1 基本概念

假设$\mathbf{B}=\{0,1\}$是布尔域, $X=\{X_1,X_2,\dots,X_n\}$是组成布尔公式的变量的集合. 赋值 v 是对于各个变量的赋值, 形式是. $\{0,1\}^n$ 若$v[i]=1$, 则表示赋值 v 满足变量X_i. 布尔公式的定义为: $f:\ \{0,1\}^n\to\mathbf{B}$. 当$f(v)=1$时, 我们说赋值v满足f; 否则, 赋值v不满足f. 当存在赋值满足f时, 我们说f是可满足的; 否则, f是不可满足的.

文字(Literal)是指一个布尔变量. 项(Term)是文字的合取. 短句(Clause)是文字的析取. 如果一个布尔公式是项的析取, 那么它是一个析取范式(DNF). 如果一个布尔公式是短句的合取, 那么它是一个合取范式(CNF).

1.2 单调理论

首先我们定义赋值上的偏序关系: 对于两个赋值x,y,如果对于所有的i.有$x[i]\preccurlyeq y[i]$则$x\preccurlyeq v$. 对于赋值$a\in\{0,1\}^n$,定义$x\preccurlyeq_a y$当且仅当$x\oplus a\preccurlyeq y\oplus a$. 这里的$\oplus$指的是按位异或操作.

定义 1.2.1 如果对于所有的赋值$v\preccurlyeq\mu$, 有$f(v)\preccurlyeq f(\mu)$, 则函数$f(x)$是单调的.

定义 1.2.2 如果对于所有的赋值$v\preccurlyeq_a\mu$,有$f[x+a](v)\preccurlyeq f[x+a](\mu)$,则函数$f(x)$是$a$-单调的. 其中, $f[x+a]$是由f这样计算得到的一个公式: 如果$a[i]=1$,则将f中的X_i替换为$\neg X_i$.

定义 1.2.3 若对于赋值v,$f(v)=1$,且对于任意的赋值$\mu\prec v$,有$f(\mu)=0$,则这样的赋值v为最小赋值向量(minterms).

定义 1.2.4 若v是满足上述定义的一个最小赋值向量.定义$T(v)$是由所有v满足的变量组成的项. 那么$T(v)$是单调的. 因为它不含任何否定的文字. 当$v=\{0\}^n$时, $T(v)=\mathbf{T}$.

引理 1.2.1 如果f是一个单调布尔公式.且A是f的所有最小赋值向量的集合, 那么, $f=\bigvee_{v\in A}T(v)$. 若$\mathrm{A}=\emptyset$, 则$f\equiv\mathrm{F}$.

证明: 假设存在赋值ξ, 满足某个项$T(v)$,即$T(v)(\xi)=1$. 那么, 由于$T(v)$是单调公式, ξ中对应的v中为 1 的位必定为 1. 所以. $\xi\succcurlyeq v$. 由于f是单调函数并且$f(v)=1$, 所以$f(\xi)=1$. 现在,再假设对于一个赋值ξ, $f(\xi)=1$. 那么, 必然存在某个最小赋值向量$v\in A$, $v\preccurlyeq\xi$,v对应项$T(v)$. 这样的最小赋值向量v是一定存在的, 因为假使对于所有的$\mu\preccurlyeq v$, $f(\mu)=1$,那么$v=\{0\}^n$依然是一个最小赋值向量. 由于$T(v)$是单调的,所以$T(v)(\xi)=1$. ■

定义 1.2.5 布尔公式f的 DNF 大小是指. f的一个最少项个数的 DNF 表示中项的个数,记为$size_{DNF}(f)$. 同样的, 布尔公式f的 CNF 大小是指, f的一个最少短句个数的 CNF 表示中短句的个数, 记为$size_{CNF}(f)$.

2 布尔公式学习算法

CDNF 算法是在给定的布尔变量集合上学习特定的布尔公式的一个算法. 假设f是定义在变量集合X上的一个布尔公式, CDNF 所使用的询问模型是通过每次与 Teacher(实际中使用的是 SMT 工具)的交流, 计算一个想要等价于目标公式f的 CDNF 公式, 直到计算的公式等价于目标公式.

Teacher 的机制是回答两种类型的询问:

①成员询问 Mem(v). v是对变量集合X的一个赋值. 如果$f(v)=1$,回答 Yes; 如果$f(v)=0$, 回答 No.

②等价询问 EQ($\bigwedge_{i=1}^{t}H_i$). 其中$\bigwedge_{i=1}^{t}H_i$算法学习过程中计算的对f的一个推测公式. 如果$\bigwedge_{i=1}^{t}H_i\equiv f$,回答 Yes, 并返回$\bigwedge_{i=1}^{t}H_i$; 否则, 返回一个反例赋值$v$, 使得$\bigwedge_{i=1}^{t}H_i(v)\neq f(v)$.

Bshouty N H在他提出CDNF算法时, 就证明了该算法的正确性[2].

在 CNDF 算法中, EQ 询问中的合取式是算法学习过程中得到的CDNF推测公式. 变量t记录了在当前推测公式(合取式)中 DNF 的个数. 初始时, $t=0$, 推测公式设为$\mathbf{T}$.

1) $t \leftarrow 0$;
2) EQ($\mathbf{T}$)→ v: 如果是 Yes. stop. 返回$\mathbf{T}$;
3) $t \leftarrow t+1$, $H_t \leftarrow \mathbf{F}$, $S_t \leftarrow \emptyset$, $a_t \leftarrow v$;
4) EQ($\bigwedge_{i=1}^{t} H_i$) → v: 如果是 Yes, stop, 返回$\bigwedge_{i=1}^{t} H_i$;
5) $I \leftarrow \{i | H_i(v) = 0\}$: 如果$I = \emptyset$, goto 3;
6) 对于I中的每一个i:
   a) $v_i \leftarrow v$;
   b) 保持$f(v_i) = 1$. 令v_i靠近a_i;
   c) $S_i \leftarrow S_i \cup \{v_i \oplus a_i\}$;
7) $H_i \leftarrow M_{DNF}(S_i)(x + a_i)$,i = 1,2, … , t;
8) Goto 4;

图 1 CDNF 算法伪代码

对于赋值a, 定义

$$M_{DNF}(a) = \begin{cases} \bigwedge_{a[i]=1} X_i, & a \neq \{0\}^n \\ T, & a = \{0\}^n \end{cases}$$

对于赋值集合S, 定义

$$M_{DNF}(S) = \begin{cases} \bigvee_{a \in S} M_{DNF}(a), & S \neq \emptyset \\ \mathbf{F}, & S = \emptyset \end{cases}$$

CDNF 算法的复杂度是用两种询问的次数的来衡量的. 如果忽略两种询问本身的复杂度, 则 CDNF 是多项式次数询问的复杂度, 最坏情况下进行的询问次数是 $size_{DNF}(f) size_{CNF}(f) n^2$ 次成员询问和 $size_{DNF}(f)\, size_{CNF}(f)$次等价询问[2]. 所以, CDNF 算法的提出证明了任意布尔公式是可以在复杂度为 $size_{DNF}(f)$, $size_{CNF}(f)$和n组成的多项式的询问次数中被学习出来的.

但是, 一个公式f往往有多种表达方式, 公式形式的差异体现在公式的长度, 组合程度, 冗余度等方面. CDNF 的学习过程中, 是根据反例赋值v来计算公式, 而反例赋值v的不同, 恰恰会使计算得到的公式表达方式不同. 虽然最终得到的公式都是与目标公式等价的, 但是, 假如忽略公式的表达的长短给视觉上带来的复杂度问题, 计算过程的不同还体现在询问次数和更新公式次数的差异上. 而最终的公式形式越简短, 表示算法学习过程中经历的公式更新次数和询问次数越少. 因为成员询问一般是多项式时间的, 而等价询问一般是指数时间的(它需要返回一个使得两个公式真值相反的赋值), 所以, 如果能将询问的次数减少, 则不仅可以提高算法的效率, 还能简化最终得到的公式.

3 A-CDNF算法

在 CDNF 计算的过程中, 每得到一个赋值v, 若它不满足f, 则把它存储在a_i中; 若它满足f, 则更新每个它不满足的H_i: 尽可能的将v靠近a_i后, 将$v \oplus a_i$存入S_i, 进而更新H_i的每个项. 根据这个过程, 我们知道计算出的 DNF 项是由尽可能少的变量合取而成的.

同时, CDNF 算法建立在单调理论的基础上, 每个H_i都是 a-单调的公式. 而根据引理 1.2.1, 单调公式可以通过所有最小赋值向量的集合计算得到. CDNF 算法在扩展学习算法到所有的布尔公式的过程中, 依赖的是 a-单调的公式的学习, 通过多个a-单调的公式的合取得到目标公式.

而由于 CDNF 算法计算过程中, 其依据的理论决定了它不依赖 Teacher 返回的反例, 即 CDNF 算法可以在随机的反例赋值下正常运算. 所以, 我们可以采用最小赋值向量的方法, 将反例赋值进行最小化后再作为计算公式的依据. 但是, 由于反例赋值是推动算法向下进行的最重要的依据, 所以在最小化的过程中, 需要保证反例本身的性质不变. 反例赋值具有以下两条性质:

① $H(v) = 1$, $f(v) = 0$;

② $H(v) = 0$, $f(v) = 1$.

将反例赋值最小化的方法很简单, 就是在保证其原有性质的情况下, 按位将其为 1 的位变为 0. 伪代码见图 2. 我们将采用最小赋值向量方法改进后的算法命名为 A-CDNF 算法, A-CDNF 算法的计算过程见图 3.

很显然, A-CDNF 算法对询问的过程没有干扰, 因此, 同原始 CDNF 算法一样, 可以使用多种 SMT 工具模拟 Teacher. 下面说明 A-CDNF 算法是正确的.

引理 3.1 $f \equiv \bigwedge_{a \in \{0,1\}^n} M_a(f)$, 其中, $M_a(f)$为公式f的最小a-单调布尔公式, 定义如下:

输入：赋值向量v;
输出：最小化的赋值向量v;
MinTerm(v):
if $f(v)=0$: flag=0;
else: flag=1;
for i =1,..., t:
  if $v[i]$=1 && flag=0:
    if $f(v_{v[i]\leftarrow 0})=0$ && $H(v_{v[i]\leftarrow 0})=1$:
      $v=v_{v[i]\leftarrow 0}$;
  if $v[i]$=1 && flag=1:
    if $f(v_{v[i]\leftarrow 0})=1$ && $H(v_{v[i]\leftarrow 0})=0$:
      $v=v_{v[i]\leftarrow 0}$;
return v;

图 2　最小赋值向量方法

1)　$t\leftarrow 0$;
2)　EQ($\mathbf{T}$) → v, $v\leftarrow$ MinTerm(v): 如果是 Yes, stop, 返回$\mathbf{T}$;
3)　$t\leftarrow t+1$, $H_t\leftarrow \mathbf{F}$, $S_*\leftarrow\emptyset$, $a_t\leftarrow v$;
4)　EQ($\wedge_{i=1}^{t}H_i$) → v, $v\leftarrow$ MinTerm(v): 如果是 Yes. stop, 返回$\wedge_{i=1}^{t}H_i$;
5)　$I\leftarrow\{i|H_i(v)=0\}$; 如果$I=\emptyset$, goto 3;
6)　对于I中的每一个i:
  a)　$v_i\leftarrow v$;
  b)　保持$f(v_i)=1$, 令v_i靠近a_i;
  c)　$S_i\leftarrow S_i\cup\{v_i\oplus a_i\}$;
7)　$H_i\leftarrow M_{DNF}(S_i)(x+a_i)$,i = 1,2, ..., t;
8)　Goto 4;

图 3　A-CDNF 算法

$$M_a(f)(x)=\begin{cases}1, & (\exists y\preccurlyeq_a x)f(y)=1\\ 0, & \text{其他}\end{cases}$$

证明：如果$\wedge_{a\in\{0,1\}^n}M_a(f)(x_0)=1$，则对于所有的$a$，存在$y\preccurlyeq_a x_0$，使$f(y)=1$. 如果我们选择$a=x_0$，则因为$y+a\preccurlyeq\{0\}^n$，所以$y=a=x_0$，$f(x_0)=1$. 这就意味着$\wedge_{a\in\{0,1\}^n}M_a(f)(x_0)=>f$. 而根据$M_a(f)$的定义，有$f=>M_a(f)$，所以有$f=>\wedge_{a\in\{0,1\}^n}M_a(f)(x_0)$. ■

定义 3.1 假设 C 是一个布尔公式类，定义 C 的单调密度$d=\mathrm{Mdim}(C)$为：使得 C 中的任意布尔公式f拥有下面性质的最少数目的赋值$a_1,a_2,\ldots,a_d$:

$$f\equiv\bigwedge_{i=1}^{d}M_{a_i}(f)$$

定义 C 的 M-基为：使得任意$f\in C$上述等式都能成立的赋值集合$\{a_1,a_2,\ldots,a_d\}$，此时d不必是满足条件的最小值.

上述引理和定义表明，最坏情况下，获取所有的赋值向量可以学习到目标公式. 但是，一般意义上来说，CDNF 算法在计算过程中，不需要全部的赋值向量. 定义 3.1 说明了，CDNF 算法在学习到最终结果时，得到了f的一个 M-基，而基中赋值的数目d是不确定的，因为询问机制得到的答案是随机的.

而 A-CDNF 算法，影响的正是赋值数目d的值. 引理 1.2.1 表明，当目标公式是单调公式时，A-CDNF 算法能够得到最小的d. 当目标公式是非单调的公式时，由于 CDNF 算法依赖单调理论，利用a-单调的公式合取出目标公式[2]，因此最小赋值向量的方法也能够发挥其优化作用，同时，由于 A-CDNF 算法主要依赖的是反例赋值的随机性和单调原理，没有对 CDNF 算法的计算过程进行修改，而是对 Teacher 的返回结果进行了最小化处理. 因此，总体来说，A-CDNF 算法的复杂性同原始 CDNF 算法相同，也是多项式的询问次数.

4　实验

实验在 RHEL 6.3 的系统，CPU 为 160 核 Intel Xeon CPU E7-8870 @ 2.40GHz 32M Cache，RAM 256G 的机器上运行. 采用 Python 语言实现 CDNF 算法和 A-CDNF 算法，使用 z3[7]作为 Teacher 回答算法询问的问题.

由于循环不变式中，公式的逻辑关系比较复杂，所以采用循环不变式抽象的布尔公式作为实验样本. 本文选用文献[3,4,6]中使用 CDNF 算法在自动推导循环不变式的系统中推导出的循环不变式以及他们组合成的较长公式作为实验的例子. 实验结果为运行 30 次的平均值. 实验数据见表 1.

表 1 实验数据对比

	CDNF (z3)			A-CDNF(z3)		
Vars	MEM	EQ	Time (s)	MEM	EQ	Time(s)
3	5	7	0.31	3	4	0.12
4	8	8	0.56	7	7	0.37
7	7	13	1.23	20	9	1.21
7	50	21	4.72	48	17	3.51
9	75	36	12.67	30	11	2.02
12	275	58	93.72	40	15	4.95
13	463	64	142.85	99	27	15.4
14	78	34	15.05	69	23	6.11
16	1500	239	1265.73	129	31	16.29
16	242	63	93.59	98	26	18.72
16	5859	532	14857.98	231	45	51.51
20	4021	484	6323.15	129	33	22.75
23	5551	620	15495.55	411	99	266.88

实验中, 最简单的变量数目为 3 的情况, 是用 CDNF 算法自动推导选择排序算法得到的循环不变式[4] 抽象而来的. 目标公式. 下面详细描述原始 CDNF 算法和 A-CDNF 算法在该实例下的计算过程(其他实例的详细过程及算法源码见 2):

表 2 原始 CDNF 算法计算过程

H	EQ($\wedge_{i=1}^{n-1} H_i \rightarrow v$)	I	A	S	MEM
T	(0,1,0)	-	-	-	-
[F]	(1,1,0)	[1]	[(0,1,0)]	[[(1,0,0)]]	1
$[x_2]$	(0,1,1)	[1]	[(0,1,0)]	[[(1,0,0),(0,0,1)]]	1
$[x_2 \vee x_3]$	(1,0,0)	[ ]	[(0,1,0), (1,0,0)]	[[(1,0,0),(0,0,1)]]	0
$[x_2 \wedge x_3$, F]	(1,1,0)	[2]	[(0,1,0), (1,0,0)]	[[(1,0,0),(0,0,1)], [(0,1,0)]]	1
$[x_1 \quad x_2$, $x_3]$	(0,0,1)	[2]	[(0,1,0), (1,0,0)]	[[(1,0,0),(0,0,1)], [(0,1,0),(0,0,1)]]	2
$[\quad \vee x_2 \vee x_3]$	-	-	-	-	-
总	EQ: 7	-	-	-	MEM: 5

[1]http://lcs.ios.ac.cn/~lyj238/public_html/student/haxl /testData.rar

表 2 和表 3 清楚的说明了 A-CDNF 算法与原始 CDNF 算法的运行结果和计算过程都不相同, A-CDNF 得到的目标公式更简洁, 且计算中需要的 MEM 和 EQ 询问次数更少. 而 A-CDNF 的高效之处, 就在于进行了反例的最小化处理.

表 3 A-CDNF 算法计算过程

H	EQ($\wedge_{i=1}^{n-1} H_i \rightarrow v$)	v'	I	A	S	MEM
T	(1,0,0)	(0,0,0)	-	-	-	-
[F]	(1,1,0)	(1,1,0)	[1]	[(0,0,0)]	[[(1,1,0)]]	2
$[x_2 \wedge x_3]$	(1,0,1)	(0,0,1)	[1]	[(0,0,0)]	[[(1,1,0),(0,0,1)]]	1
$[x_1 \wedge x_2 \vee x_3]$	-	-	-	-	-	-
总	EQ: 4					MEM: 3

从表 1 我们可以看到, 当公式中变量数目的增多时, 算法进行询问 MEM 和 EQ 的次数, 以及运行时间, 整体呈较大的增长趋势, 可见变量数目对于运行时间有极大的影响, 这也说明了 CDNF 算法对于输入的变量的依赖性强. 当变量数目较大的, 如何减少 CDNF 算法随之产生的波动有重大意义. 例如将 CDNF 应用到循环不变式的自动推导中时, 由于自动推导算法依赖最初输入的原子公式(抽象为布尔变量), 输入较多的原子公式是算法准确性的需要, 也是算法能自动推导出正确结果的保证. 而 A-CDNF 算法随着变量数目的增多, 运行时间增加不大. 这说明了对于需要变量数目巨大的情况下, A-CDNF 算法更高效且具有更好的适用性.

同时, 我们可以看到, 当变量数目不变和变量数目少量增加时, 算法进行询问 MEM 和 EQ 的次数, 以及运行时间并无一定的规律, 却有较大波动. 这说明, 公式的逻辑复杂性也是影响 CDNF 算法的运行效率的重要因素, 逻辑更复杂的公式, 运行时需要的询问次数和时间会就更多. 但是, A-CDNF 算法使询问次数保持在较低的增长范围, 表现出了比原始 CDNF 算法更高的效率.

5 结论

布尔公式学习算法 CDNF 算法, 本质上是一种机器学习的方法. 它在理论领域到实际的形式验证领域的发展, 证明了这种学习算法作为一种隐式算法, 可以解决传统显示算法无法解决的有一定困难的问题, 并能够极大的提高多种验证技巧的性能.

我们利用最小赋值向量的理论, 改进 CDNF 算法计算的反例, 使其不依赖外部 Teacher 的返回结果, 利用更加有意义的反例减少询问 EQ 和 MEM 的次数, 促进算法的学习过程. 由于 CDNF 算法本身的复杂度与变量的个数和公式的逻辑有关, 所以随着变量数目的增加和公式逻辑复杂性的变化, CDNF 算法的运行时

间有较大增加和波动. 改进的 CDNF 算法却能够极大的减少两种询问的次数, 并提高运行的效率. 当实际应用中需要大量的变量和复杂的逻辑时, 改进的 CDNF 算法尤为适用. 在当前 CDNF 算法适用的形式验证系统如循环不变式推导、程序终止性证明中, CDNF 算法是进行计算和推导的核心算法. 由于改进的 CDNF 算法并未改变算法学习的过程, 只是对 Teacher 返回的反例进行了处理, 所以, 改进的 CDNF 算法能够提高 CDNF 算法在实际应用中的效率.

参考文献

1 Angluin D. Queries and concept learning. Machine Learning, 1988, 2(4): 319–342.

2 Bshouty NH. Exact learning boolean functions via the monotone theory. Information and Computation, 1995, 123(1): 146–153.

3 Jung Y, Kong S, Wang B Y, et al. Deriving invariants by algorithmic learning, decision procedures, and predicate abstraction. Verification, Model Checking, and Abstract Interpretation. Springer Berlin Heidelberg. 2010. 180–196.

4 Kong S, Jung Y, David C, et al. Automatically inferring quantified loop invariants by algorithmic learning from simple templates. Programming Languages and Systems. Springer Berlin Heidelberg. 2010. 328–343.

5 Chen YF, Wang BY. BULL: a library for learning algorithms of boolean functions. Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg. 2013. 537–542.

6 Lee W, Wang BY, Yi K. Termination analysis with algorithmic learning. Computer Aided Verification. Springer Berlin Heidelberg. 2012. 88–104.

7 De Moura L, Bjrner N. Z3: An efficient SMT solver. Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg. 2008. 337–340.