Software Engineering and Applications 软件工程与应用, 2016, 5(6), 319-326
Published Online December 2016 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2016.56037

Hans汉斯

Optimization of the Boolean Function Representing Transition Relation

Xiaozhen Zhang, Jianguo Jiang

School of Mathematics, Liaoning Normal University, Dalian Liaoning
Email: zhangxiaozhend@sina.cn, jjgbox@sina.com

Received: Dec. 1st, 2016; accepted: Dec. 18th, 2016; published: Dec. 21st, 2016

Abstract

In Symbolic Model checking, as a comparatively efficient method of representing transition relation with OBDD, we need firstly represent it as a boolean function with the aid of SMV language, and then synthesize the OBDD using relevant algorithms. However, during the course of representation from transition relation to the boolean function, we often encounter two problems: first, the assignment of next state is non-deterministic; second, the number of non-input variables with certain variation regularity is more. In this paper, through the concept of equivalent-input variables, we impose restriction on the assignment of the boolean function formula used to represent transition relation, and give definite solutions to these two problems, enabling the boolean function representation of transition relation is further accurate and efficient.

Keywords

Symbolic Model Checking, OBDD, Transition Relation, Boolean Function, Equivalent-Input Variables

迁移关系的布尔函数表示的优化

张小珍，江建国

辽宁师范大学数学学院，辽宁 大连
Email: zhangxiaozhend@sina.cn, jjgbox@sina.com

收稿日期：2016年12月1日；录用日期：2016年12月18日；发布日期：2016年12月21日

文章引用: 张小珍, 江建国. 迁移关系的布尔函数表示的优化[J]. 软件工程与应用, 2016, 5(6): 319-326.
http://dx.doi.org/10.12677/sea.2016.56037

摘　要

符号模型检测中，将迁移关系表示为OBDD的一种较为高效的方法是先借助SMV语言将迁移关系表示为布尔函数，再利用相关算法综合得其OBDD。但在将迁移关系表示为布尔函数时常会遇到两个问题：第一，下一状态取值不确定；第二，具有一定变化规律的非输入变量较多。本文提出等效输入变量的概念，对迁移关系的布尔函数公式的取值条件加以限制，给出了这两个问题的具体解决办法，进而使得迁移关系的布尔函数表示更加准确、高效。

关键词

符号模型检测，OBDD，迁移关系，布尔函数，等效输入变量

1. 引言

随着计算机科学技术的迅速发展，关于计算机硬件和软件系统的正确性验证也越来越重要，而关于其功能的验证主要采用模拟法和形式化验证法。相对于模拟法来说，形式化验证法更能保证验证的完备性，所以其在工业上的应用更为广泛。此外，形式化验证法大体上可分为等价性检测、模型检测和定理证明三类。

模型检测[1] [2]是一种自动的、基于模型的形式化验证技术，它通过遍历状态空间来对系统某方面的性质作出验证。然而，随着系统规模的不断扩大，状态空间的大小呈指数级增长，于是就引起了状态空间爆炸问题。为了克服显式模型检测在解决此问题上的局限性，符号模型检测应运而生。

符号模型检测是由J. R. Burch，E. M. Clarke和K. L. McMillan等人于20世纪90年代初提出的[3] [4] [5]，它通过布尔函数来表示状态集、迁移关系，而布尔函数则以有序二叉判定图(Ordered Binary Decision Diagrams，简称 OBDD) [6] [7] [8]来呈现。符号模型检测的提出使得状态数超过 10^{20} 的实际系统得到了验证，在很大程度上缓解了状态空间爆炸问题，进而促进了验证技术的重大突破。而近几年关于 OBDD 变量序的研究[9] [10]及符号模型检测在应用中的研究[11] [12] [13] [14]使得形式化验证技术得到了重大发展。

符号模型检测中，在将迁移关系表示为数据结构 OBDD 时常会遇到两个问题：第一，状态变量在下一状态的取值不确定；第二，具有一定变化规律的非输入变量较多时，它们之间关系的处理。对于这两个问题，文献[15]中给出了一些相应的的解决办法，但其较为抽象，而且形式化过程、结果都存在有一些赘余表示。

本文通过提出等效输入变量的概念，给出了借助 SMV (Symbolic Model Verifier)语言描述迁移系统并由此得出迁移关系的布尔函数时，对于这两个问题具体明确的解决办法，此外对文献[15]中所提关于 $f^{\rightarrow}$ 的公式进行了改进，使得形式化过程更加高效，形式化结果更加简洁，进而直接或者间接地提高了符号模型检测中某些具体问题的验证效率。

2. 基本概念

模型检测中，对系统某方面的性质作出验证时，首先要对实际系统建模得出一个迁移系统。

定义 1：迁移系统 TS 是一个多元组 $(S, I, \rightarrow, P, L)$，其中 S 表示系统的状态集合，I 表示系统的初始状态集合且满足 $I \subseteq S$，$\rightarrow$ 表示迁移关系，它是状态集 S 上的二元关系：对于 $\forall s \in S$，都 $\exists s' \in S$ 满足 $s \rightarrow s'$，P 是用于描述状态的原子命题集，L 是从状态集 S 到原子命题集 P 的幂集上的函数，称为标记函数。

一个迁移系统 $TS = (S, I, \rightarrow, P, L)$ 可以通过一个有向图来表示，如图 1 所示。

对于图 1 所示迁移系统 TS 有：

$$S=\{s_0,s_1,s_2\}$$

$$I=\{s_0\}$$

$$\rightarrow=\{(s_0,s_1),(s_0,s_2),(s_1,s_2),(s_2,s_1)\}$$

$$P=\{x,y,z\}$$

$$L(s_0)=\{x,y\},\ L(s_1)=\{y\},\ L(s_2)=\{z\}$$

符号模型检测中，需首先将迁移系统中的状态集和迁移关系表示为布尔函数的形式，而且在此表示过程中会涉及到一些变量。

定义 2：若一个变量 x 的定义域是 $\{0,1\}$，则称此变量 x 为布尔变量。令 x,y 为布尔变量，若集合 $\{0,1\}^n$ 上的函数 $f:\{0,1\}^n\rightarrow\{0,1\}^n$ 满足布尔运算 $\bar{}$、$\bullet$、$+$、$\oplus$，则称此函数 f 为从 $\{0,1\}^n$ 到 $\{0,1\}^n$ 上的 n 元布尔函数。

用来描述系统各状态的自变量，称为状态变量。系统模型的任意一个状态可以认为是对状态变量集中各元素的一个赋值。状态变量的取值可能会随着状态的迁移而变化。而由外界环境所决定的变量，称为输入变量。在有限状态的迁移系统中，不可以指定输入变量的初值，而且它的取值变化完全取决于模拟环境，而与状态迁移毫无关系。

布尔函数的表示方法有很多种，在诸多表示中，OBDD 是布尔函数较为紧凑且较为规范的表示形式。

定义 3：二叉判定图(Binary Decision Diagrams，简称 BDD)是一个有限有向无环图，满足条件：

1) 具有唯一初始结点；

2) 所有非终止结点用布尔变量标记，且都恰好有两条边指向其它结点：一个用虚线表示，一个用实线表示，其中虚线表示布尔变量取值为 0，实线表示取值为 1；

3) 所有终止结点标记为 0 或 1。

有序二叉判定图(Ordered Binary Decision Diagrams，简称 OBDD)是带有某些有序变量表的 BDD。

如图 2 是无变量序的 BDD，而图 3 是带有序 $[x,y]$ 的 OBDD。

3. 迁移关系的布尔函数表示

符号模型检测中，需要将迁移系统中的迁移关系表示为 OBDD，而当系统较大较为复杂时，则需借助 SMV 语言描述迁移系统，并依据此得出表示迁移关系的布尔函数 $f^{\rightarrow}$，进而通过相关算法综合求得其 OBDD。而用 SMV 语言描述迁移系统以及由此得出迁移关系的布尔函数时，通常会遇到两个问题的处理：第一个问题是，状态变量在下一状态的取值不确定；第二个问题是：具有一定变化规律的非输入变量较多时，它们之间关系的处理。

为有效解决这两个问题，我们引入了一种叫做等效输入变量的状态变量：在任意下一状态取值都不确定的状态变量，称为等效输入变量。很明显，等效输入变量不是输入变量，但其取值不会随着状态的迁移呈现一定的变化规律。

本节工作前提是状态集已被表示为布尔值和布尔函数。符号说明：x_i 表示状态变量，f_i 表示状态变量 x_i 在下一状态的取值所满足的关系表达式，而 x_i' 则表示状态变量 x_i 在下一状态的取值。

3.1. 非确定的下一状态

迁移关系表示为布尔函数过程中，为解决下一状态为非确定赋值的问题，我们对所研究的状态变量

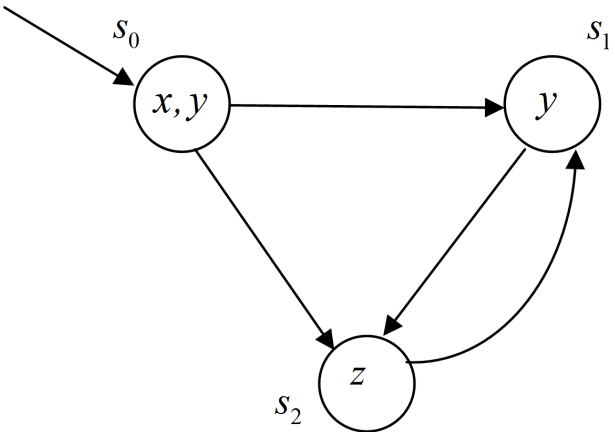

Figure 1. The directed graph of the transition system $TS=(S,I,\rightarrow,P,L)$

图 1. 迁移系统 $TS=(S,I,\rightarrow,P,L)$ 的有向图

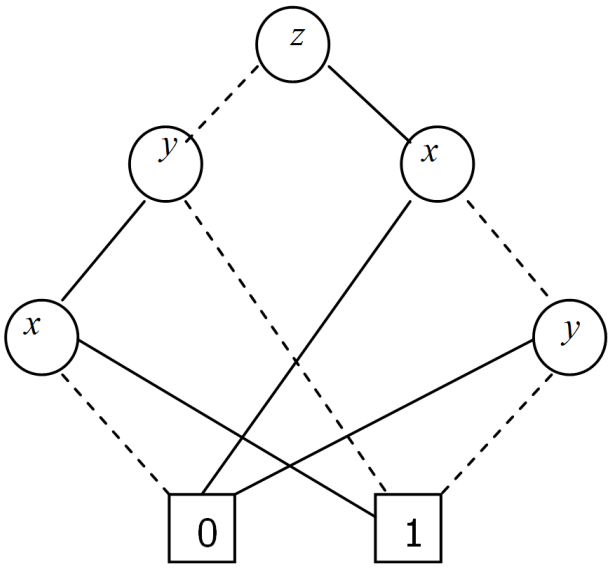

Figure 2. A BDD without an ordering of variables

图 2. 无变量序的 BDD

进行考究，看状态变量是否为等效输入变量，进而给出不同的解决办法，使得形式化结果更加简明。

若状态变量 x_k 为非等效输入变量，即只在某些条件下，其下一状态取值不确定：

1) 用 SMV 语言描述迁移关系时，需将该条件下的 $next(x_k)$ 赋值为所有可能的取值组成的集合；

2) 由 SMV 语言求得迁移关系的布尔函数时，只需将该条件的 f_i 取值为 x_i' 来表示，而无需再引入其它输入变量，因为输入变量的引入表示系统状态的增多。

若状态变量 x_k 为等效输入变量，即其在任意下一状态的赋值都不确定：

1) 用 SMV 语言进行描述时，不需要对其下一状态取值进行描述；

2) 由 SMV 语言求得迁移关系的布尔函数时，即依据 $f^{\rightarrow}=\prod_{1\le i\le n} x_i' \leftrightarrow f_i$ 求取 $f^{\rightarrow}$ 时，我们对 x_i 的条件进行修改，要求其不仅不可以是输入变量，而且不可以是等效输入变量。假设 x_j 是等效输入变量，则无论任何时候都满足 $f_j=x_j'$，而 $x_j' \leftrightarrow f_j = x_j' \leftrightarrow x_j' = 1$，于是 $f^{\rightarrow}=\prod_{1\le i\le n} x_i \leftrightarrow f_i = \prod_{1\le i\le n, i\ne j} x_i \leftrightarrow f_i$，通过这样的限制，可

以避免形式化过程中的冗余表示。

例 1：已知迁移系统 TS_1 的有向图如图 4 所示，试用 SMV 语言对该系统进行描述，并由此给出其迁移关系的布尔函数 $f^{\rightarrow}$。

首先，分析非输入变量 x_1, x_2 的变化规律可知：变量 x_1 在任意一个下一状态的取值是不确定的，既有 TRUE，也有 FALSE，因此为等效输入变量，所以在 SMV 语言描绘中不对 $next(x_1)$ 进行说明。而变量 x_2 则遵循一定的变化规率：当 x_1 为 TRUE 时，x_2 在下一状态的取值为 FALSE；而 x_1 为 FALSE 时，x_2 在下一状态的取值是不确定的，既有 TRUE 也有 FALSE。

由此得出该迁移系统的 SMV 语言描绘如下所示：

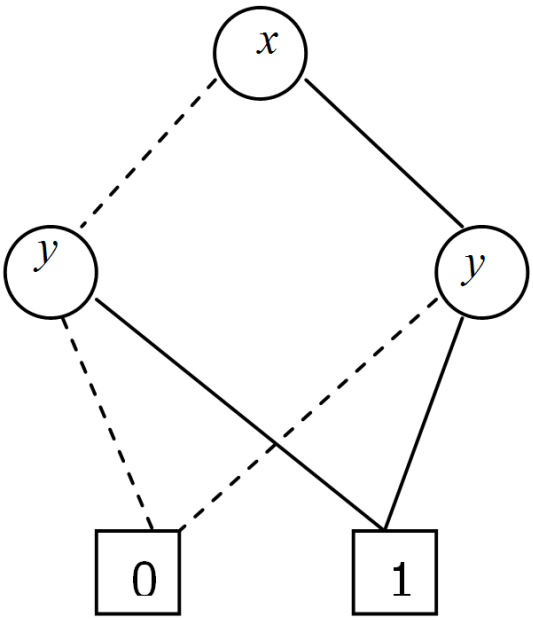

Figure 3. A OBDD with variable ordering [*x*,*y*]
图 3. 带有序[*x*,*y*]的 OBDD

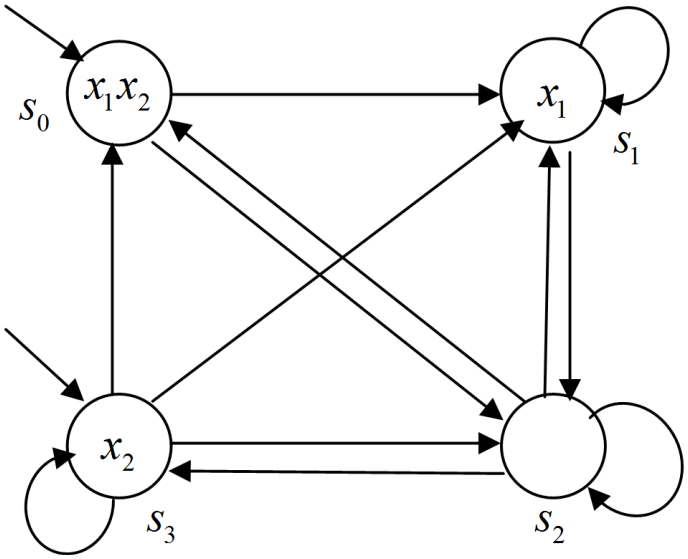

Figure 4. The directed graph of transition system TS_1
图 4. 迁移系统 TS_1 的有向图

```
MODULE main
VAR
    x1 : boolean;
    x2 : boolean;
ASSIGN
    init(x2) := 1;
    next(x2) := case
                    x1 : 0;
                     1 : {0，1};
                esac;
```

其次，由 SMV 语言描绘求得迁移关系的布尔函数 $f^{\rightarrow}$ 时，关于下一状态是不确定值的处理：首先分析 x_1 是等效输入变量，所以不需要求 f_1。而 x_2 是非等效输入变量的状态变量，当 $x_1=0$ 时， f_2 的取值是不确定的，此时需使 f_2 取值为 x_2' 即可；而且写 f_2 表达式时，没有必要列真值表，只需将 $f_2=1$ 时的各种情况用布尔表达式表示然后再相加即可。

由 SMV 语言中的 next 表达式分析： $f_2=1$ 时的条件是 $x_1=0$ 且 $x_2'=1$，故 $f_2=\overline{x}_1\cdot x_2'$。

最后，得出图 4 所示系统 TS_1 的迁移关系的布尔函数是：

$$f^{\rightarrow}=x_2' \leftrightarrow f_2=\overline{x}_2' \oplus (\overline{x}_1\cdot x_2').$$

3.2. 多个非等效输入

当迁移系统中有具有一定变化规律的非输入变量较多，即有多个非等效输入变量的状态变量 x_i 时，处理好各 $next(x_i)$ 之间的关系，方可得到原迁移系统的正确描述。因为用 SMV 语言描述迁移系统时，需分析各 x_i 的变化规律，而当非等效输入变量的状态变量较多时，分析某个变量的变化规律可能会引入一些原迁移系统中不存在的迁移，此时在分析其他变量的变化规律时需避免这些赘余迁移，也就是说，依据每个 $next(x_i)$ 都会得到一个迁移图，而它们的公共部分才为已知迁移系统的有向图。

例 2：已知迁移系统 TS_2 的有向图如图 5 所示，试用 SMV 语言对该系统进行描述，并由此给出其迁移关系的布尔函数 $f^{\rightarrow}$。

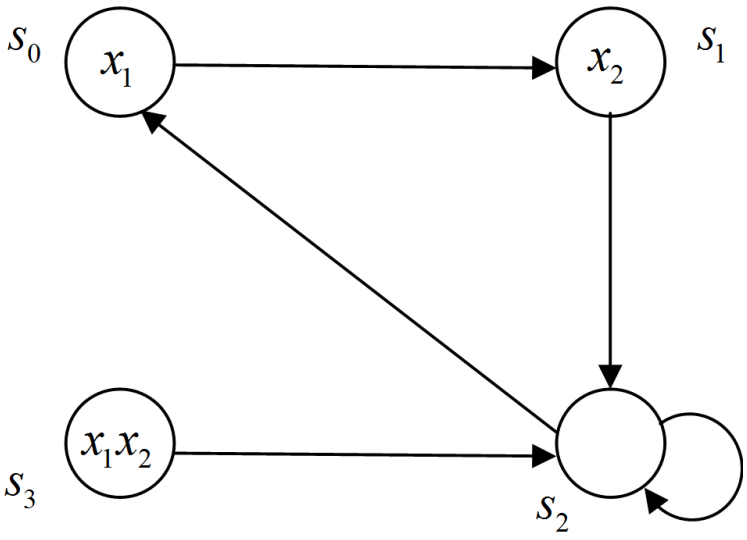

Figure 5. The directed graph of transition system TS_2
图 5. 迁移系统 TS_2 的有向图

首先，分析可知，状态变量 x_1 和 x_2 都不是等效输入变量，而变量 x_1 的变化规律是：只有当 x_1 和 x_2 均为 FALSE 时，下一状态 x_1 的真值不确定，既有 TRUE，也有 FALSE；而其它情况下，下一状态 x_1 的取值一定为 FALSE；但殊不知，这个变化规律增加了迁移 $s_0 \to s_2, s_3 \to s_1, s_1 \to s_1, s_2 \to s_1, s_2 \to s_3$，如图 6 中的左图虚线箭头所示，因此在分析 x_2 的变化规律时，必须避免这些迁移的出现。对于 x_2，要求当 $x_1 = 1$ 且 $x_2 = 0$ 时，x_2 在下一状态的取值为 TRUE，这样避免了迁移 $s_0 \to s_2$；而其他情况下，x_2 在下一状态的取值一定为 FALSE，这样的赋值避免了其他的赘余迁移。虽然又引入了额外的迁移，但也不影响最后结果，如图 6 中的右图说明了变量 x_2 的变化规律，其中虚线部分为所引入的无关迁移。图 6 中左图和右图的公共部分即为图 5 中的迁移。

该迁移系统的 SMV 语言描绘如下所示：

```
MODULE main
VAR
    x1 : boolean;
    x2 : boolean;
ASSIGN
    next(x1) := case
                    !x1 & !x2 : {0，1};
                              1 : 0;
                esac;
    next(x2) := case
                    x1 & ! x2 : 1;
                              1 : 0;
                esac;
```

由迁移系统的 SMV 语言描述求 $f^{\rightarrow}$ 时，利用同例 1 的方法，只需将各 $f_i = 1$ 时的条件转化为对应的布尔表达式再相加即可。所以可求得 $f_1 = \overline{x}_1 \cdot \overline{x}_2 \cdot x_1'$，$f_2 = x_1 \cdot \overline{x}_2$，于是可得图 5 中所示迁移系统 TS_2 的迁移关系的布尔函数是：

$$f^{\rightarrow} = (x_1' \leftrightarrow f_1) \cdot (x_2' \leftrightarrow f_2) = (\overline{x_1'} \oplus \overline{x}_1 \cdot \overline{x}_2 \cdot x_1') \cdot (\overline{x_2'} \oplus x_1 \cdot \overline{x}_2)$$

4. 结语

本文给出了符号模型检测中将迁移关系表示为布尔函数时，关于下一状态取值不确定以及非等效输

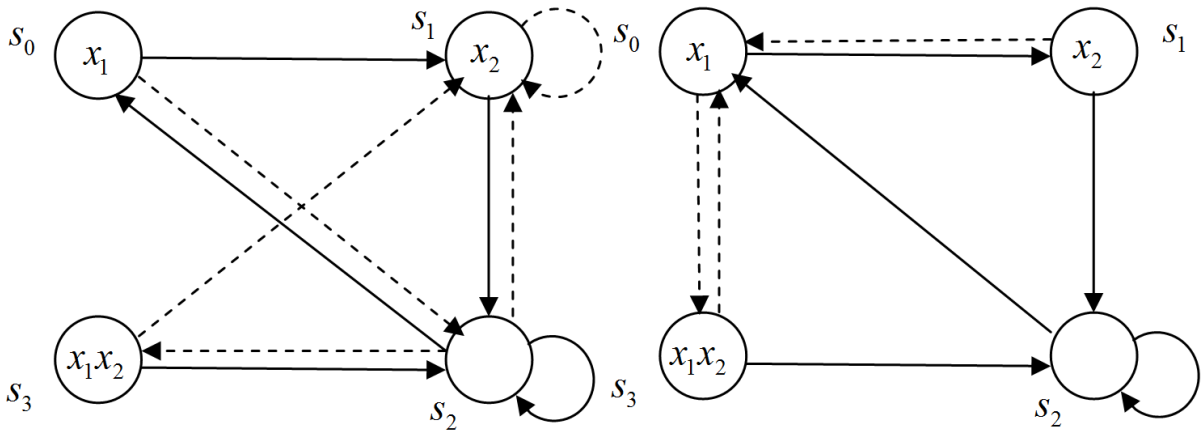

Figure 6. The respectively corresponding transition graph of the varying rules of x_1 and x_2
图 6. x_1 和 x_2 的变化规律各自所对应的迁移图

入变量较多时的处理方法，从而使得迁移关系的布尔函数表示更加高效、准确。但对于该表示过程中所遇到的其它问题，比如用当前值无法表示下一状态取值而需要引入其它输入变量等问题的处理，还有待于进一步研究。

参考文献 (References)

[1] Clarke, E.M. and Emerson, E.A. (1981) Design and Synthesis of Synchronization Skeletons Using Branching-time Temporal Logic. *Logic of Programs, Lecture Notes in Computer Science*, **133**, 52-71.

[2] Queille J.P. and Sifakis. J. (1981) Specification and Verification of Concurrent Systems in CESAR. 5*th International Symposium on Programming*, **137**, 337-351.

[3] Burch J.R., Clarke J.M., McMillan K.L., Dill D.L. and Hwang, J. (1992) System Model Checking: 10^{20} States and Beyond. *Information and Computation*, **98**, 142-170. https://doi.org/10.1016/0890-5401(92)90017-A

[4] Clarke E., Grumberg O. and Long, D. (1993) Verification Tools for Finite-State Concurrent Systems. In: de Bakker, J.W., de Roever, W.P. and Rozenberg, G. Eds., *A Decade of Concurrency*, Lecture Notes in Computer Science, Springer, Verlag, 124-175.

[5] McMillan, K.L. (1993) Symbol Model Checking. Kluwer Academic Publishers, Norwell. https://doi.org/10.1007/978-1-4615-3190-6

[6] Bryant, R.E. (1986) Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Compilers*, **35**, 677-691. https://doi.org/10.1109/TC.1986.1676819

[7] Bryant, R.E. (1991) On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication. *IEEE Transactions on Computers*, **40**, 205-213. https://doi.org/10.1109/12.73590

[8] Bryant, R.E. (1992) Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Computing Surveys*, **24**, 293-318. https://doi.org/10.1145/136035.136043

[9] Lai Y., Liu D.Y. and Wang, S.S. (2013) Reduced Ordered Binary Decision Diagram with Implied Literals: A New Knowledge Compilation Approach. *Knowledge and Information Systems*, **35**, 665-712. https://doi.org/10.1007/s10115-012-0525-6

[10] Bollig, B. (2016) On the Minimization of (Complete) Ordered Binary Decision Diagrams. *Theory of Computing Systems*, **59**, 532-559. https://doi.org/10.1007/s00224-015-9657-x

[11] Beyer, D. and Stahlbauer, A. (2013) BDD-Based Software Model Checking with $\text{CPA}_{\text{CHECKER}}$. *Doctoral Workshop on Mathematical & Engineering Methods in Computer Science*, **7721**, 1-11. https://doi.org/10.1007/978-3-642-36046-6_1

[12] Beyer, D. and Stahlbauer, A. (2014) BDD-based Software Verification Applications to Event-Condition-Action Systems. *International Journal on Software Tools for Technology Transfer*, **16**, 507-518. https://doi.org/10.1007/s10009-014-0334-1

[13] 孔庆爱. 基于符号模型检测若干问题的研究及应用[D]: [硕士学位论文]. 长春: 吉林大学, 2008: 1-70.

[14] 逄涛. 命题投影时序逻辑符号模型检测及其应用研究[D]: [博士学位论文]. 西安: 西安电子科技大学, 2014: 1-89.

[15] Huth, M. and Ryan, M. (2004) Logic in Computer Science: Modeling and Reasoning about Systems. Cambridge University, Cambridge, 382-390. https://doi.org/10.1017/cbo9780511810275