分类号 TP393 学号 10069066

UDC 密级 公开

工学博士学位论文

开放计算环境下可满足问题的安全外包

博士生姓名 秦莹

学科专业 计算机科学与技术

研究方向 计算机软件与理论

指导教师 贾焰 教授

国防科学技术大学研究生院

二〇一五年九月

Research on secure outsourcing SAT Solving in Open Environment

Candidate: QIN Ying

Supervisor: Professor JIA Yan

A dissertation

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Engineering

in Computer Science and Technology

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

September 18, 2015

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科学技术大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：开放计算环境下可满足问题的安全外包

学位论文作者签名：_____ 日期： 年 月 日

学位论文版权使用授权书

本人完全了解国防科学技术大学有关保留、使用学位论文的规定。本人授权国防科学技术大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅;可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：开放计算环境下可满足问题的安全外包

学位论文作者签名：_____ 日期： 年 月 日

作者指导教师签名：_____ 日期： 年 月 日

目 录

表 目 录

图 目 录

摘　要

命题可满足 (SAT) 问题是指，对于一个命题逻辑公式 (通常表示为 CNF 公式)，是否存在对其所有变量的一个真值赋值使之成立。该问题是计算机科学和工程领域的重要问题，被广泛用于软硬件形式化设计与验证、密码学等多个领域。随着软硬件规模日益增大，设计和验证过程中生成的 CNF 公式规模也相应的急剧增长。

作为应对该挑战的一个有效手段，开放环境下的 SAT 求解服务，可利用云和网格等提供的弹性计算资源满足不同规模问题的求解需求。但是，开放环境在为用户提供使用便利的同时，未授权的第三方访问等问题对用户数据的隐私安全提出了严峻挑战。现有研究试图通过对计算数据进行全同态加密来达到隐私保护的目的，但是复杂的加密原语和随之带来的昂贵计算开销，限制了该方法的实用性。

为解决该问题，本文针对软硬件设计验证中 SAT 问题隐私保护的需求，从求解效率的角度出发，研究了 SAT 求解中输入数据结构信息保护和输出数据隐藏问题，提出了基于加噪混淆的数据隐私保护方法。该方法通过在原始输入数据中混入特定的噪声数据，实现输入和输出信息的隐藏；并通过混淆规则保持解空间一致性，以保证混淆后的 SAT 问题可直接使用原始求解器求解。本文以软硬件形式化验证和硬件对偶综合设计中的 SAT 问题为研究对象，系统研究了 SAT 问题求解中数据隐私保护的若干问题。本文的主要研究内容和创新点如下。

1. 开放环境下基于加噪混淆的 SAT 求解框架。现有保护 CNF 公式隐私的方法包括基于加密和基于数据分割的两类。这些算法都依赖于全空间遍历求解，求解效率低下。本文通过对 CNF 公式自身逻辑特点的分析，提出了基于加噪混淆算法的伪装方法。该方法通过在原始 CNF 公式中无缝的混入噪音公式，在隐藏原有的结构信息的同时，保持原有的 CNF 数据形式和解空间，从而复用目前已有的求解算法。该算法在效率和隐私保护上取得了良好的折中。本文从理论上证明了该算法的正确性并通过大量的仿真实验证明了该算法的高效性。

2. 结构感知的混淆策略。识别混淆后 CNF 公式结构的困难程度，是衡量混淆算法有效性的重要指标。本文研究了硬件设计中 CNF 公式携带的结构信息特征，提出结构感知的加噪策略。该策略将原始公式的结构映射至新的带噪声结构，使得该新结构与原始结构不存在明确对应，从而增大了第三方恢复出原始结构的难度。本文从理论上论证了结构感知混淆算法的有效性，并通过实验验证了方法的性能。

3. 解空间投影等价的混淆策略。寻找特定 CNF 公式的所有解的算法，称为 ALLSAT 算法。该算法构成了对上述混淆算法的一个潜在威胁：通过求解出全空间的解，并将在所有解中具有相同唯一赋值的变量作为候选噪音变量，进而从混淆后的公式中识别出原始公式。为了解决该问题，本文研究了解空间投影等价的混淆策略，以打破噪音变量必然是相同单一赋值的断言，以抵御上述基于 ALLSAT 的攻击。

4. 解空间上估计的 CNF 混淆算法。在研究了上述保护输入数据结构信息的算法后，本文进一步研究了隐藏 SAT 问题求解结果的方法。针对输出信息隐藏，本文提出了解空间上估计的 CNF 混淆算法。通过扩展噪音公式解空间，使得混淆后 SAT 问题的解空间为原始解空间的上估计。通过引入噪音解实现对原始解的隐藏。本文从理论上充分证明了算法的正确性，并通过大量的仿真实验验证了方法的有效性和性能。

5. 解空间可调制的 CNF 混淆算法。由于上估计的 CNF 混淆算法可能会多次调用 SAT 求解，其调用的概率取决于真实解与噪音解的比率。本文讨论了将投影等价和上估计两种混淆特性结合来实现解空间可调制的混淆算法。

6. CNF 混淆算法的有效性评价。混淆算法通过混淆规则保证混淆后的 CNF 公式可复用已有的求解器，并且可以用较小的开销恢复出原始的解。但是除此之外，在开放环境下为保证 SAT 计算外包的顺利实施，混淆算法仍然需要满足其他的特性。结合程序混淆的有效性评价标准，本文抽象出 CNF 公式混淆的有效性评价标准，通过对混淆策略的细分，针对两种混淆策略，对混淆后公式的隐形性和适应力进行了定量分析和定性评价，为设计出更好的混淆策略提供依据。

综上所述，本文对 SAT 问题外包中的若干关键问题进行了深入的研究，提出了具有高可用性、低开销的解决方案，并通过理论分析和大量的仿真实验验证了所提出算法的有效性和性能，对于促进 SAT 计算外包服务的实用化有一定的理论意义和应用价值。

关键词: **开放环境；SAT 求解；混淆；非等价解空间；解空间加噪；结构感知**

ABSTRACT

Propositional satisfiability[1] has been widely used in hardware and software verification[2, 3], cryptography[4] etc. With the rapid increase of the hardware and software system size, the size of SAT problem generated from verification also increases rapidly. On the other hand, Cloud and grid can provide elastic computing resource, which make outsourcing hard SAT problem to public Cloud or computation grid[5–7] very attractive.

However, security issues make many customers reluctant to move their critical computation tasks to grid or Cloud. Since the grid is constructed by loosely connecting a large number of readily available, high-end, heterogeneous computing facilities[5], the risks posed by hoarding participants in grid computing environments are real and immediate[8]. While the Cloud vendors can be trusted and the Cloud infrastructure (i.e., the virtualization layer) can be assumed to be secure, the virtual machines cannot be trusted to be always honest and loyal. Literature[9] points out security vulnerability that Amazon EC2 suffered from: Since Amazon Machine Image (AMI) are widely shared among the EC2 community, a malicious AMI could flood the community with hundreds of infected virtual instances. Literature [10] also points out the possibility of attacking virtual machine through another virtual machine in the same physical machine.

These facts show that input and output data of SAT problem may be exposed to untrusted third party, who may inspect valuable information from these data. For example, SAT program originated from verification may suffer from leakage of privacy, such as circuit structure information. Works carried out by Roy[11] and Fu [12] suggest the possibility of extracting circuit information from CNF formula. Furthermore, Du[8] called solution of hard SAT problem generating from cryptograph etc as high-value rare events, and point out that the solution of SAT problem should also be treated as privacy, because it may be leaked to third party by hoarding participants. Moreover, the SAT solver deployed in grid or Cloud may also be compromised by adversary, who may compel SAT solver to return incorrect result to mislead verification.

These threats put customers who plan to outsource SAT solving in a dilemma: using public Cloud or grid in open environment is cost-efficiently, but may suffer from leakage of privacy or incorrect result. In order to meet this challenge, we develop novel techniques

for outsourcing SAT solving in Cloud or in computing grid securely, which can preserve privacy of input and output, without changing the solution.

Key Words: Open Environment; SAT Solving; Obfuscation; Solution Space Over-approximation; Phrase Transition sensitive

第一章　绪论

随着云计算和网格计算等基于互联网的按需计算模式由概念走向实践，基于互联网开放环境下的计算基础设施正逐渐成为计算资源和存储资源的主要提供方式之一。与此相适应，在开放计算环境下提供存储和计算服务，实现随时随地数据访问和按需计算，使用户摆脱地域束缚，也成为一种非常有吸引力的解决方案。

与此同时，命题可满足性问题 (SAT 问题) 是计算机科学和工程领域的一个重要问题。SAT 求解算法针对特定的命题逻辑公式，搜寻对其变量集合的一个赋值，以使该公式为真。SAT 求解算法在测试向量自动生成、符号模型检验及组合等价性检查等电子设计自动化领域[2]，以及密码破解中得到了广泛的应用。随着网格计算和云计算的发展，SAT 求解也成为开放环境下最普遍的计算服务之一。科研机构、云计算服务公司都试图或已经推出了开放环境下基于 SAT 求解的验证和密码破解应用 [5–7, 13, 14]。。

但是，和存储服务的迅速普及不同，对计算隐私泄露的担忧一直阻碍着计算服务的商业化普及。目前实用的加密技术，如基于属性的加密技术[15–17]，可以实现对加密数据的细粒度访问控制，从而解决数据传输或存储中的隐私保护，但却无法直接应用于计算数据。可搜索存储加密技术[18–20] 解决了开放环境下数据隐私性和可搜索性共存问题，但也仅仅适用于搜索计算。全同态加密[21] 从理论上提供了对通用计算透明的数据隐私性保护策略，但其计算复杂性过高，仍不能满足实用性的要求。类似的，数据隐私问题一直是阻碍开放环境 SAT 求解在电路设计、软硬件验证和密码破解等重要应用中发挥作用的决定性因素。本文将对软硬件设计验证中 SAT 问题进行研究，从实用性的角度出发，研究其中的数据隐私保护问题，以实现开放环境下可信且高效的 SAT 求解。

1.1　可满足问题

1.1.1　可满足问题以及相关概念

在分析 SAT 求解问题的隐私保护之前，有必要对软硬件设计验证领域 SAT 计算的相关概念进行介绍。

1.1.1.1　可满足问题

布尔集合表示为 $\mathbf{B} = \{0, 1\}$。对于一个布尔变量集合 V 上的公式 F，命题可满足问题（简称 SAT 问题）是指寻找一个可满足赋值 $A : V \to \mathbf{B}$，使得 F 为 1。如果这样的可满足赋值存在，F 就是可满足的，可满足赋值称为公式 F 的一个解；

否则，F 是不可满足的。公式的不可满足子集称为不可满足核。搜索可满足赋值 A 的计算程序称为 SAT 求解器[22]。

通常情况下，SAT 求解器接受的输入公式是合取范式（CNF），其中公式为子句的合取，子句是文字的析取，而文字则是变量或是反。

公式 (1.1) 的 Φ 是一个 CNF 公式，包含四个变量 x_1, x_2, x_3, x_4 和三个子句 $x_1 \vee \neg x_2, x_2 \vee x_3, x_2 \vee \neg x_4$，在子句 $x_1 \vee \neg x_2$ 中，文字 x_1 是变量 x_1 的正文字，而文字 $\neg x_2$ 是变量 x_2 的负文字。

$$\Phi = (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \neg x_4) \tag{1.1}$$

子句 C 中文字的数量记为 $|C|$。CNF 公式 F 中的子句数记为 $|F|$。例如 $|x_1 \vee \neg x_2| \equiv 2$，而 $|\Phi| \equiv 3$。

CNF 公式 F 的变量集合，记为 V_F。CNF 公式 F 中所有变量被赋值为解 A，记为 $F(A/V_F)$。

1.1.1.2 Tseitin 编码

在硬件验证过程中，电路和属性通过 Tseitin 编码[23] 转换为 CNF 公式，而后交给 SAT 求解器求解。由于所有电路都可以被表示为二输入与门 AND2 和非门 INV 的组合形式，所以在这里我们仅仅给出 AND2 门和 INV 门的 Tseitin 编码：

1. 对于非门 $z = \neg x$，由 Tseitin 编码产生的 CNF 公式为 $(x \vee z) \wedge (\neg x \vee \neg z)$。

2. 对于二输入与门 $z = x_1 \wedge x_2$, 由 Tseitin 编码产生的 CNF 公式为 $(\neg x_1 \vee \neg x_2 \vee z) \wedge (x_1 \vee \neg z) \wedge (x_2 \vee \neg z)$。

3. 对于一个表示为二输入与门和非门组合的复杂电路 C，由 Tseitin 编码产生的 CNF 公式 $Tseitin(C)$ 是所有这些门的 Tseitin 编码的合取。

对于一个包含非门 $d = \neg a$ 和二输入与门 $e = d \wedge c$ 的简单电路 C，由 Tseitin 编码产生的 CNF 公式如公式 (1.2) 所示。

$$Tseitin(C) = \left\{ \begin{array}{ll} & (a \vee d) \\ \wedge & (\neg a \vee \neg d) \end{array} \right\} \wedge \left\{ \begin{array}{ll} & (\neg e \vee c) \\ \wedge & (\neg e \vee d) \\ \wedge & (e \vee \neg c \vee \neg d) \end{array} \right\} \tag{1.2}$$

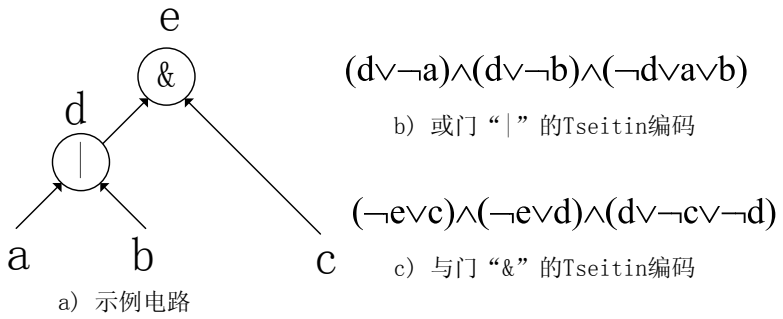


图 1.1　示例电路及其编码

1.1.2　可满足问题求解

1.1.2.1　可满足问题的简单解法

为了方便对 SAT 求解过程的描述，我们以图1.1中的电路为例子，首先给出一个简单低效但是直观的求解方法。并在下面逐步描述对该方法的改进措施，从而最终描述清楚现代 SAT 求解器中常用的高效算法。

最简单的 SAT 求解算法是简单地遍历所有可能的变量赋值，形成树形的二叉搜索空间。对于图1.1b) 的 SAT 公式，将导致图1.2所示的二叉搜索树。其中打钩的叶节点表示合法的求解结果。每次对特定变量进行二叉分解的步骤称为决策，每次决策产生一个新的决策层。图1.2的决策层 1、2 和 3 分别对应于分别对变量 a、b 和 c 进行二叉分解。

1.1.2.2　布尔约束传播 (BCP)

为了使一个特定的 SAT 公式成立，必须使其中每个子句都成立。而为了使某个特定子句成立，其中必须存在至少一个文字成立。因此当在某个子句中，只有一个特定的文字 w 尚未取值，而其他所有文字均取值为 0 时，则该文字必须取值为 1。如果该文字为某个特定变量 v，这将导致 v 取值为 1，否则取值为 0。这一推导过程称为布尔约束传播。

以图1.1b) 的或门的 Tseitin 编码为例，为了使该公式成立，每个子句都必须成立。以第一个子句 $d \wedge a$ 为例，当 a 为 1 时，$d \wedge a$ 化简为 d，为了使其成立，d

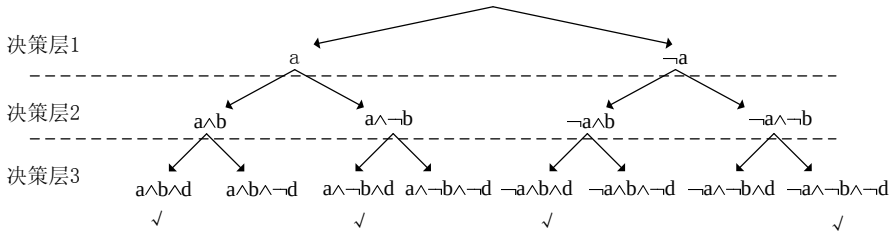


图 1.2　基于完全二叉树遍历的 SAT 求解

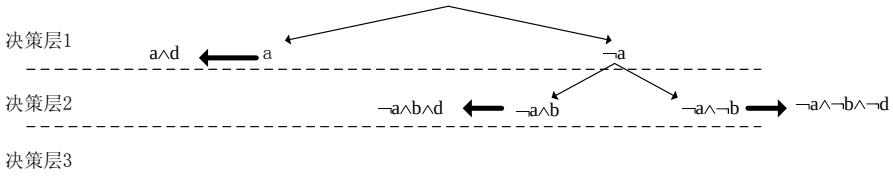


图 1.3　布尔约束传播

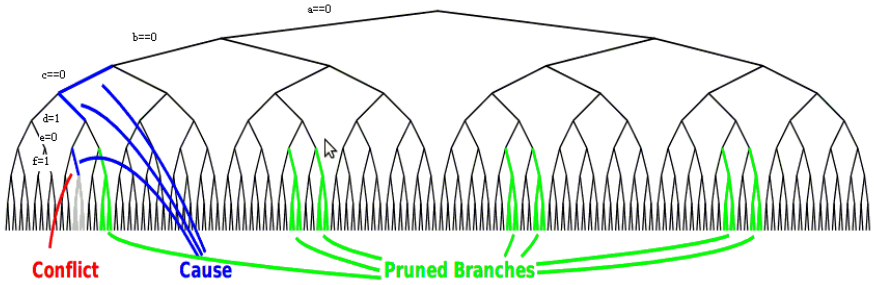


图 1.4　冲突指导的子句学习

必须取值为 1。此时搜索树如图1.3 所示。其中粗线代表在特定决策层内部的 BCP 操作。

1.1.2.3　冲突指导的子句学习

冲突指导的子句学习和非正交回溯[24] 是提升 SAT 求解器性能的另一个重要手段。其中非正交回溯与与本文重点关注的数据结构关系不大，因此将仅描述冲突指导的子句学习。

为了简明起见，仍然使用一个例子描述冲突指导的子句学习。如图1.4所示的一个二叉搜索树，当到达红色的标记为 conflict 的节点时，有 $\{a \equiv 0, b \equiv 0, c \equiv 0, d \equiv 1, e \equiv 0, f \equiv 1\}$。这将导致某个短句中的所有文字均成为 0，称这种情况为一个冲突 (conflict)。此时冲突分析算法将对该子句中的每一个文字，沿着如图1.4粗线所示的 BCP 关系逆向回溯，以便找到导致此次冲突的根本原因。假设找到的三个变量分别为 $\{c \equiv 0, d \equiv 1, f \equiv 1\}$，这意味着 a、b 和 e 与本次冲突无关。无论以后 a、b 和 e 取任何值，只要遇到 $\{c \equiv 0, d \equiv 1, f \equiv 1\}$ 的情况，都不必继续搜索。这意味图1.4中绿色所示的分支都可以被剪掉。

为了达到这种剪枝效果，将对冲突分析的结果中每个变量取反，以构造一个冲突学习子句。即 $\{c \equiv 0, d \equiv 1, f \equiv 1\}$ 将会产生一个冲突学习子句 $\{c \vee \neg d \vee \neg f\}$，并加入子句数组。以后每次当 c、d 和 f 三个变量中的两个满足 $\{c \equiv 0, d \equiv 1, f \equiv 1\}$，则将立即通过冲突学习子句产生一次 BCP，使得第三个变量无法满足 $\{c \equiv 0, d \equiv 1, f \equiv 1\}$。这就构成了一次剪枝操作。

1.1.2.4 MiniSat 求解器的递增求解机制

本文中，我们使用 MiniSat 求解器[25] 求解所有 CNF 公式。和其他基于冲突学习机制[26] 的 SAT 求解器类似，MiniSat 从在搜索中遇到的冲突中产生学习短句，并记录他们以避免类似的冲突再次出现。该机制能够极大的提升 SAT 求解器的性能。

在许多应用中，经常存在一系列紧密关联的 CNF 公式。如果在一个 CNF 公式求解过程中得到的学习短句能够被其他 CNF 公式共享，则所有 CNF 公式的求解速度都能够得到极大的提升。

MiniSat 提供了一个增量求解机制以共享这些学习短句。该机制包括两个接口函数：

1. *addClause*(F) 用于将一个 CNF 公式 F 添加到 MiniSat 的短句数据库，以用于下一轮求解。

2. *solve*(A) 接收一个文字集合 A 作为假设，并求解 CNF 公式 $F \wedge \bigwedge_{a\in A} a$。其中 F 是在 *addClause* 中被加入短句数据库的 CNF 公式。

基于该机制，可以针对一个相同的 CNF 公式 F，使用不同的文字集合 A，来产生并递增地高效求解不同的 $F \wedge \bigwedge_{a\in A} a$。

1.1.3 可满足赋值遍历

多数时候，可以满足特定 SAT 问题的解并不是唯一的，求出所有可满足解的过程被称为可满足赋值遍历（简称 ALLSAT 求解）。

ALLSAT 求解由基于 SAT 求解的可满足赋值遍历算法来实现。直观上看，调用一次 SAT 求解器可以获得 SAT 问题的一个解，也就是一个完整的可满足赋值。将这个完整的可满足赋值中每一个文字取反，构造出阻断（block）子句并加入到待求解的 SAT 问题公式中，以引导 SAT 求解器避开已搜索过的解。通过多次重复，最终可以获得 SAT 问题所有的解。

绝大多数可满足赋值遍历算法致力于将由 SAT 求解得到的一个完整的赋值扩展为一个包含较多赋值的赋值集合，以便减少调用 SAT 求解器的次数并压缩存储赋值解的空间开销。文献[27] 提出了第一个此类算法。他在 SAT 求解器求解过程中构造一个蕴含图，用以记录每个赋值之间的依赖关系。每个不在该图中的赋值变量都可以从最终结果中剔除。在文献[28] 和[29] 中，每个变量如果在其不被约束的情况下不能使 $obj \equiv 0$ 被满足的话，则该变量可以从最终结果中剔除。在文献[30] 和[31, 32] 中，冲突分析方法被用于剔除与可满足性无关的变量。在文献[33] 中，变量集合被划分为重要变量和非重要变量集合。搜索过程中重要变量的优先

级高于非重要变量。因此重要变量子集构成了一个搜索树，而该树的每一个叶节点是非重要变量的一个搜索子树。Cofactoring [34] 则通过将非重要变量设置为 SAT 求解器返回的值以缩减搜索空间。

另一类算法通过 Craig 插值以扩大解集合。文献[35] 提出了第一个此类算法。该算法构造两个相互矛盾的公式，并从他们的不可满足证明中抽取 Craig 插值。在文献[36] 中，Craig 插值的产生过程类似于传统的可满足赋值遍历算法。不过其扩展算法包含两步，分别对应于两个参与计算的公式。该算法是第一个不需要产生不可满足证明的 Craig 插值算法。

1.1.4 Craig 插值的原理和实现

在通常的 SAT 求解器，包括本文使用的 MiniSat[25] 中，要求待求解的公式被表示为 CNF 格式。其中一个公式是多个子句的合取 (conjunction)，而每一个子句是多个文字的析取 (disjunction)，而每个文字是一个布尔变量 v 或者其反 $\neg v$。如公式 $(v_0 \vee \neg v_1 \vee v_2) \wedge (v_1 \vee v_2) \wedge (\neg v_0 \vee v_2)$，包含子句 $v_0 \vee \neg v_1 \vee v_2$，$v_1 \vee v_2$ 和 $\neg v_0 \vee v_2$。而子句 $v_0 \vee \neg v_1 \vee v_2$ 包含文字 v_0，$\neg v_1$ 和 v_2。

当存在一个变量 v，使得一个子句 c 中同时包含两个文字 v 和 $\neg v$，则称 c 为 tautological 的。我们通常假设有待 SAT 求解器求解的公式中所有的子句都是非 tautological 的。

假设公式 F 的布尔变量全集为 V。若存在对 V 的赋值函数 $A : V \to \{0, 1\}$，使得 F 中的每个子句均能取值为 1，则称 F 是可满足的，此时 SAT 求解器能够找到赋值函数 A。否则称 F 为不可满足的，此时 SAT 求解器能够产生如下一小节所述的不可满足证明。

1.1.4.1 不可满足证明

对于两个子句 $c_1 = v \vee A$ 和 $c_2 = \neg v \vee B$，当 $A \vee B$ 是 tautological 时，$A \vee B$ 称为它们的 **resolvant**。而 v 称为它们的 **pivot**。易知以下事实:

$$\begin{aligned} resolvant(c_1, c_2) &= \exists v, c_1 \wedge c_2 \\ c_1 \wedge c_2 &\to resolvant(c_1, c_2) \end{aligned} \tag{1.3}$$

定义 1.1: 对于不可满足公式 F，假设其子句集合为 C，则其不可满足证明 Π 是一个有向无环图 (V_Π, E_Π)，其中 V_Π 是子句集合，而 E_Π 是连接 V_Π 中子句的有向边集合。Π 满足如下要求：

1. 对于节点 $c \in V_\Pi$：

(a) 要么 $c \in C$，此时称 c 为 Π 的根

(b) 或者 c 有且仅有两个扇入边 $c_1 \rightarrow c$ 和 $c_2 \rightarrow c$，使得 c 是 c_1 和 c_2 的 resolvant。

2. 空子句是 Π 的唯一一个叶节点。

直观的说，Π 就是一棵树，以子句集合 C 的子集为根，以空子句为唯一叶节点。而每个节点 c 的两个扇入边 $c_1 \rightarrow c$ 和 $c_2 \rightarrow c$ 代表了一个 resolving 关系 $c := resolvant(c_1, c_2)$。

包括本文使用的 MiniSat 求解器[25] 在内的许多 SAT 求解器，当公式不可满足时都将产生一个不可满足证明 Π。

1.1.4.2 Craig 插值算法

根据文献[37]，给定两个布尔逻辑公式 A 和 B，若 $A \wedge B$ 不可满足，则存在仅使用了 A 和 B 共同变量的公式 I，使得 $A \Rightarrow I$ 且 $I \wedge B$ 不可满足。I 被称为 A 针对 B 的 Craig 插值[37]。

目前最常见且最高效的产生 Craig 插值的算法是 McMillan 算法[38]。其基本原理描述如下。

对于上述公式 A 和 B，已知 $A \wedge B$ 不可满足，而 Π 是 SAT 求解器给出的不可满足证明。当一个变量 v 同时出现在 A 和 B 中时，我们称其为全局变量。若 v 只出现在 A 中，则称其为 A 本地变量。

对于文字 v 或者 $\neg v$，当变量 v 是全局变量或者 A 本地变量时，称该文字为全局文字或者 A 本地文字。

对于子句 c，令 $g(c)$ 为 c 中所有全局文字的析取，而 $l(c)$ 为 c 中所有 A 本地文字的析取。

例如，假设有两个子句 $c_1 = (a \vee b \vee \neg c)$ 和 $c_2 = (b \vee c \vee \neg d)$。并假设 $A = \{c_1\}$ 和 $B = \{c_2\}$。则 $g(c_1) = (b \vee \neg c)$，$l(c_1) = (a)$，$g(c_2) = (b \vee c)$，$l(c_2) = FALSE$。

定义 1.2: 令 (A, B) 为一对公式，而 Π 是 $A \wedge B$ 的不可满足证明，且其唯一叶节点是空子句 $FALSE$。对于每一个节点 $c \in V_\Pi$，令 $p(c)$ 为如下定义的一个公式：

1. 如果 c 是根节点则

(a) 如果 $c \in A$ 则 $p(c) = g(c)$

(b) 否则 $p(c) = TRUE$

2. 否则令 c_1 和 c_2 分别是 c 的两个扇入节点，而 v 是他们的 pivot 变量

(a) 如果 v 是 A 本地变量，则 $p(c) = p(c_1) \vee p(c_2)$。

(b) 否则 $p(c) = p(c_1) \wedge p(c_2)$。

上述定义3.2是构造性的，已经给出了从不可满足证明 Π 得到最终的 Craig 插值的算法，即以 Π 的根节点为起点，为每一个 c 计算相应的 $p(c)$，直至到达最终的唯一叶节点 $FALSE$。我们有以下定理：

定理 1.1: 定义3.2为唯一叶节点 $FALSE$ 产生的 $p(FALSE)$ 即为 A 相对于 B 的 Craig 插值。

该定理的详细证明可见文献[39]。

计算 A 相对于 B 的 Craig 插值的时间复杂性为 $O(N + L)$，其中 N 是 Π 中包含的节点个数 $|V_\Pi|$，而 L 是 Π 中的文字个数 $\Sigma_{c \in V_\Pi}|c|$。而所产生的插值可以视为一个电路，其空间复杂性为 $|O(N + L)|$。当然，Π 的尺寸在最坏情况下也是 $A \wedge B$ 的尺寸的指数。

1.2 面向软硬件设计验证的可满足问题求解

可满足问题 (SAT)[1] 是硬件电路设计和软件可信验证领域[2, 3] 共同关注的重要问题。许多重要的电路设计和软件验证问题均可转换为可满足性问题，并由 SAT 求解器求解。随着集成电路制造工艺的发展，在单个芯片内集成的晶体管个数将在 2020 年接近一千亿；而社会信息化程度的提高促使软件系统越来越复杂，以 Linux 操作系统为例，在 2008 年仅其内核代码就已经突破 1 千万行。软硬件系统的规模日益增大，服务于硬件设计和软件验证的 SAT 求解器的运算量也急剧攀升。在过去的 10 年，作为形式化工具基本引擎的 SAT 求解器性能已经显著提升，几分钟内即可处理数百万变量和数亿子句，但是依然无法满足日益增长的计算要求。

传统的硬件辅助与设计（EDA）综合与验证工具的核心框架通常包含以下主要功能模块：

1. 抽象问题表示：该模块用于管理与特定推理过程和引擎无关，但是与问题本身密切相关的数据结构，如简化布尔电路（Reduced Boolean circuits）[40] 和与非图（And-Inverter Graph）[41] 等。

2. 问题编码：该模块用于将特定的抽象问题表示，转换为满足特定推理引擎，如二叉决策图 (简称 BDD)[42] 或 SAT 要求的数据结构，以便进行高效的推理工作。针对 SAT 推理引擎，该模块通常使用在空间和时间方面均具有多项式复杂性的 Tseitin[23] 编码。

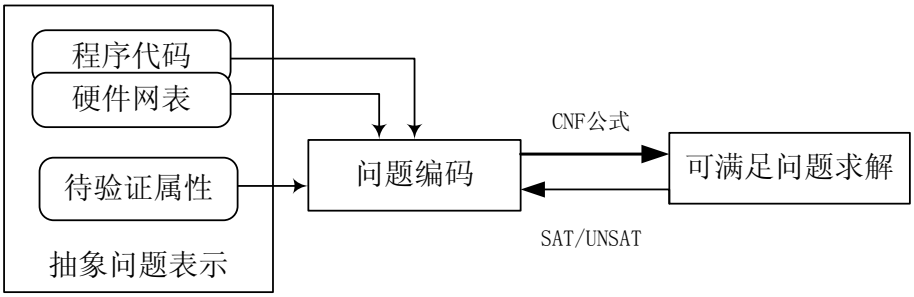


图 1.5　基于 SAT 求解的软硬件验证流程

3.BDD 和 SAT 推理引擎：这两个模块负责具体的推理工作。绝大多数 EDA 工具和软件验证工具的核心推理引擎为二叉决策图 (BDD) 和可满足求解器 (SAT)。其中 BDD 受到归一化表示方式导致的状态空间爆炸问题的困扰，通常仅用于需要归一化特性的小规模推理问题，如抽象谓词的表示等。而 SAT 则通常较少受到状态空间爆炸的影响，且天生具有内在的并行性和可扩展性；另一方面 SAT 问题是 NP 难问题，其求解时间和问题结构相关，对计算资源的需求也随具体问题而不同。

软件程序验证工具也具有类似于硬件设计验证工具的核心框架。SAT 求解器通常是作为核心推理引擎集成到具体问题求解器中。因此，从逻辑流程上看，硬件和软件形式化验证通常都是通过抽象问题表示之后再进行问题编码，而后将问题的可满足性求解过程交给 SAT 求解器完成，流程如图1.5所示。

因此将软硬件设计验证中产生的复杂 SAT 问题外包到云或网格环境下，利用其提供的弹性计算资源，成为一种有吸引的解决方案。科研机构和商业公司也已经开展了在云和网格环境下进行 SAT 问题求解的研究 [5–7, 13, 14]。如美国华盛顿大学研究小组推出支持云 SAT 求解的 sTile 系统[13]；芬兰阿尔托大学研究小组推出基于 NorduGrid 的 SAT 求解器[5]；美国圣巴巴拉大学研究小组推出 GridSAT 系统[14]；形式化验证服务提供商 OneSpin 公司和 Plunify 公司于 2013 年，联合推出了面向云平台的基于 SAT 求解的硬件验证商业服务[7]。

1.3　开放环境下数据隐私的安全威胁

云计算和网格依托于互联网，与互联网这种开放环境的便利快捷相伴而生的是，安全威胁也无处不在。一方面，公有云和网格计算节点均直接部署在广域网上，与用户处于不同的安全域。用户的服务请求可能面临来自网络中的多重威胁；另一方面，云服务提供商或是网格计算节点无法证明其内部行为可以信任。

由于网格计算是由松散耦合的高端计算设施组成[5]，网格环境下恶意计算节点是客观存在的[8]；而在云计算环境下，虽然云硬件平台提供商及其基础设施(虚拟层）是可被信赖，在其上运行的虚拟机却不总是可以信赖的。文献[9] 指出，

著名的云计算提供商亚马逊的 EC2 受到了虚拟机影像滥用的困扰, 被污染虚拟机映像会迅速扩散到整个社区；而文献[10] 则指出了处于同一台物理机器上的虚拟机之间攻击的可能性。与此相照应，早在 2007 年，《华盛顿邮报》就披露了客户关系管理领域著名的云服务提供商 Saleforce.com 由于受到安全攻击而导致大量租户数据泄露与丢失[43]；2010 年下半年谷歌解雇了两名入侵租户的私有账户以获取隐私数据的员工[44]。Gartner 公司发布的研究报告[45] 显示，所采访的企业中 70% 以上认为出于对数据安全性与隐私保护的怀疑，在近期内不会采用云计算技术。RSA 首席技术官也指出[46]，在企业将现有的应用向第三方云服务提供商提供的云环境迁移过程中，要考虑的首要问题是对云计算的数据安全问题。此处的安全不仅指数据的可用，更加注重的是数据的隐私保护。针对 OneSpin 公司推出的硬件形式化验证云服务，新闻评论[47] 指出验证数据的隐私是用户最为关心的因素。

而针对计算结果的安全性，对早期的志愿计算 SETI@home 项目的统计发现[8]，这类基于网格的开放计算环境存在三类威胁：**私心的计算参与者**：由于计算结果具有稀缺性，私心的计算参与者会出现奇货可居的意识，他们会完全遵照协议的规定，尽力计算出正确结果，但会出于利益原因，将有价值的结果信息透露给第三方，从而损害用户的隐私安全。**懒惰的计算参与者**：由于大计算量会耗费很多计算资源，出于节约计算成本的考虑，计算参与者可能不按照约定来进行足量的计算，以此来节省开销，使用部分结果来作为最终结果。**恶意的计算参与者**：可能出于某种目的，计算参与者完全违反协议规定，随意返回错误结果来欺骗用户，误导用户决策。而在 2009 年对云计算模型和网格计算模型的比较一文[48] 中，Ian Foster 指出云计算在安全措施设计成熟度还远不及网格，这就使得网格计算下影响结果正确性的威胁也很可能对云计算环境造成影响。

这些事实指出，外包到云计算或网格这类开放环境下的 SAT 问题, 由于计算模式将数据和处理的控制权从用户转移至云服务方，导致具体的处理过程用户不可控；其输入和输出数据可能会被未授权的第三方访问，这些潜在的威胁者可能会从这些数据中获取有价值的信息。糟糕的是，即使发生了上述信息泄露的情况，如果不辅助以技术手段，用户难以察觉和追踪；更为恶劣的情况是，部署在网格或云环境下的 SAT 求解器可能会被迫使返回错误的结果。

来源于软硬件验证的 SAT 问题，可能遭受硬件结构信息泄露的问题；Roy[11] 和 Fu[12] 的工作指出了从 CNF 公式中抽取电路结构信息的可能性。Zvika[49]、Yuriy Brun[13] 等人的工作也指出在云计算环境下进行 SAT 问题求解需要解决隐私保护问题。另一方面，Du[8] 将某些复杂 SAT 问题的解称作高价值稀有事件，指出 SAT 问题的解也应该被视作为隐私；如来源于密码破解的 SAT 问题，奇货可居的计算参与者可能会因利益问题而将其泄露给第三方。

1.4 开放计算环境下可满足问题求解的隐私保护问题

开放环境下的这些威胁将 SAT 计算服务的潜在用户置于进退维谷的境地：使用公共的云计算或网格计算基础设施在系统维护性和可用性上面具有较高的性价比，但却会面临隐私泄露和错误结果等安全问题的困扰。在开放计算环境下，计算数据的隐私保护问题看起来是一个不可能完成的问题：由于计算是在开放环境下完成的，未经加密处理的原始输入和输出数据势必会引发泄露的风险，而经过传统加密算法处理之后的计算数据由于对计算不再透明，因而丧失了可计算性。因此在保持可计算性的前提下，讨论 SAT 数据的隐私保护，成为了开放计算环境下的最大挑战。

2009 年，Gentry 等人针对开放计算环境下的数据隐私保护问题，提出完全同态加密的概念。这一概念描绘了开放环境下计算外包的美好愿景：经过完全同态加密，在保持数据隐私性的同时，仍然可保持原有数据的可计算性。完全同态加密的理论基础是，由于任何计算都可以分解为一系列微观加乘计算，并且在有限步内对加乘计算透明的加密算法确实存在。因而任意的计算都可以分解为一系列的有限次针对加密数据的加乘计算。这无疑从理论上扫清了开放环境下计算外包的安全障碍。但是，由于完全同态加密需要将计算分解为细粒度的有限步加乘计算。所带来的昂贵计算开销，使其距离实用化还有相当的距离。

同样为了解决外包计算的数据隐私保护，Atallah 等提出了针对具体问题，进行计算数据伪装的概念。例如针对矩阵求解类计算，将有待外包的数据与随机对角矩阵进行矩阵乘，对外包的矩阵数据进行伪装加密。由于矩阵计算的可逆性，伪装加密后结果可以通过可逆的矩阵运算得到。数据伪装方法充分利用了问题的特点，不改变原有计算过程和数据的外在形式，是一种直接实用的方案。但目前针对 SAT 问题的数据伪装的研究还处于空白。

作为一种基础的计算引擎，软硬件设计问题中的 SAT 问题具有其内在的特点：任何硬件设计和软件程序都可以表示为与或非等门的集合。本文从实用化的角度出发，希望在复用原有求解器的前提下，探讨软硬件设计、验证领域中 SAT 问题的隐私保护方法。

1.4.1 CNF 公式中的结构信息

CNF 公式是 SAT 求解的输入数据，来源于软硬件验证及设计中的 CNF 公式中会包含硬件电路结构信息；文献[11, 12] 给出了从 CNF 公式中获取电路结构信息的算法细节，首先了解算法中用到的概念。

定义 1.3 (CNF **标记**): 门 g 的 CNF 标记就是它的 Tseitin 编码 $Tseitin(g)$。CNF 标记中的每个子句称为门的特征子句。包含门中所有变量的特征子句称为关键子句。对应于门输出的变量称为输出变量。

公式 (1.2) 中的 AND2 门，$\neg e \vee c$ 是它的一个特征子句，$e \vee \neg c \vee \neg d$ 是它的关键子句。e 是输出变量。

文献[11] 指出，在一种编码规则下，具有相同特征函数的门必然会被编码成为相同的 CNF 标记，也就是相同的子句集合。通过探索这种结构特征可以恢复电路结构，已知的结构检测算法基于以下定义的有向超图和二分图概念。

定义 1.4 (**超图**): 以 CNF 公式中的子句为节点、变量为边，形成的图称为超图（Hypergraph）。超图 $G(V,E)$ 中：V 中每个节点对应 F 中一个子句；E 中每条边对应 F 中一个变量。如果两个子句包含相同的变量，就在两个子句之间连接一条边，并用变量标注。

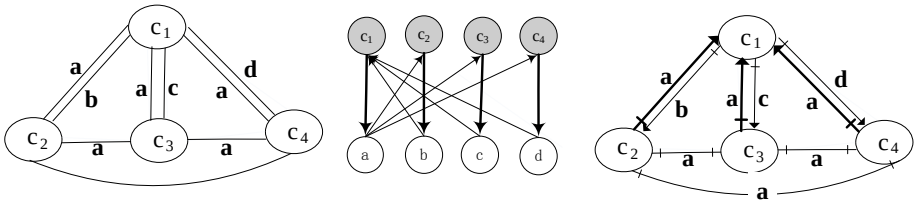
在超图表示方式下，存在具有不同 CNF 标记的两个门，却具有相同超图表示的情况，如 AND3 和 OR3，均对应图1.6a) 中的超图。为了克服该问题，在电路结构检测算法[11] 中，使用有向超图进行区分。

定义 1.5 (**有向超图**): 在定义1.4给出的超图基础上，根据子句中文字的正负、为边添加标记，形成的图称为有向超图（Directed Hypergraph）。

定义 1.6 (**二分图**): 将子句和变量均视为节点，同时将变量和子句的从属关系视为边，形成的图称为二分图（Bipartite Graph）。在二分图 $G(VE)$ 中：V 中每个顶点对应于集合中的一个子句或一个变量，即 $V = V_{cls} \bigcap V_{var}$，其中 V_{cls} 为子句集合、V_{var} 为变量集合。E 中的每条边对应于一个子句 / 变量对，如果变量出现在子句中，就在变量和子句之间连接一条边；变量为负值则对应一条负边，反之为正边。

以 AND 门为例，AND3 门的超图如图1.6a) 所示；其有向超图对应于图1.6b)，其中使用 ↑ 表示正，┼ 表示负；其二分图对应于图1.6c) 所示的二分图。

基于上述定义，CNF 公式可以表示为包含多种 CNF 特征子图的超图或二分图。这种图结构使得利用基于子图同构和模式匹配技术来恢复出电路结构和程序结构信息成为可能。



a) AND3超图　　*c)*AND3二分图　　*b)*AND3有向超图

图 1.6　超图和二分图

1.4.2　SAT 问题隐私保护的对象

基于上述概念, CNF 公式可以转化超图 *G*, 在图中匹配常用的 CNF 标记，通过同构子图的方式即可恢复出 CNF 公式携带的门信息。进一步，可用最大无关集来表示恢复出来的电路信息。基于关键子句和 CNF 标记的模式匹配还可以检测出所有门，并构建最大匹配门的子集。除了门的结构信息，在来源于软硬件验证的 CNF 公式还包含了状态迁移关系。Roy[11] 和 Fu[12] 给出了具体的实现技术。潜在的攻击者可以利用这些技术手段恢复出电路结构，并得到电路的状态迁移关系。因此，CNF 标记和关键子句是特别需要保护的重要信息。

另一方面，在验证领域，某些 SAT 问题的解反映了该系统的某些特性是否达到，因此解也应作为隐私加以保护。

1.5　本文的主要工作

来源于软硬件验证和设计的 SAT 问题，由于其 CNF 公式和其解中包含了电路结构以及电路迁移关系等敏感信息，在开放计算环境下求解，必须防止这些敏感信息的泄露。本文的工作也围绕着保护上述敏感信息展开。

1.5.1　已有工作的局限

针对开放计算环境下的计算数据隐私保护问题，2009 年 Gentry[21] 等人开创性提出完全同态加密的概念。经过完全同态加密，在保持数据隐私性的同时，仍然可保持原有数据的可计算性。由于任何的计算都可以分解为一系列微观加乘计算，因此寻找可保持对加乘的加密算法成为了一个努力的方向。但是由于计算需要被分解为细粒度的有限步的加乘操作，因此同态加密后的计算效率一直制约着该方法的实用化。

针对 CNF 公式隐私保护方面相关的研究才刚刚开始，2013 年 Brakerski[49] 等人面向云计算环境，首次探讨了使用多线性映射和坡度编码策略对 d-CNF 进行混淆的方法。使用随机和带有噪声的编码，并提供测试过程来确保编码元素的等价。这种方法基于有限加速假设，混淆后的 CNF 使用最原始的二叉树搜索的方法进行求解，无法利用目前经典的 SAT 求解器。

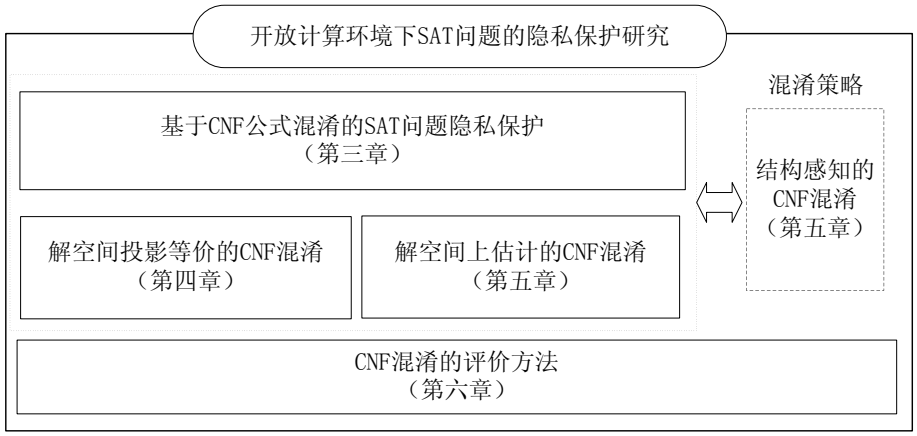


图 1.7　本文研究内容

Yuriy Brun[13] 等人则使用了 stile 数据分布的模式，通过将数据条块计算，提高攻击者获得完整 CNF 公式难度，以此降低数据被窃取的可能性。该方法目前也仅仅支持简单的二叉遍历赋值求解方法，无法利用已有的 SAT 求解加速算法。

上述的工作都试图重新构造求解器，无法利用目前已有的求解器研究成果。

1.5.2　研究内容与创新点

鉴于目前的研究现状，本文力图从 SAT 问题特性出发，寻求具有实用性的隐私保护方法。图1.7给出了本文的主要研究内容。

本文受国家高技术研究和发展计划“智能云服务与管理平台核心软件及系统”(项目编号 2013AA01A212）和国家自然科学基金项目“面向通讯应用的自动对偶综合研究”(项目编号 61070132）的支持，主要贡献和创新点如下:

1. 开放环境下基于加噪 CNF 混淆的 SAT 求解框架。CNF 公式混淆是保证开放环境下的 SAT 求解隐私性的重要手段。现有的方法基于双射或多射群加密，通过分段坡度编码对 CNF 公式进行混淆，从而隐藏 CNF 公式中携带的结构信息。但是这种方法改变了原有 CNF 公式的外部表示，需要设计新的求解算法。并且在目前仅可使用全空间遍历的方法进行求解，无法利用已有成熟的 SAT 求解算法，因此极大降低了其实用性。本文通过对 CNF 公式自身逻辑特点的分析，提出了基于加噪思路的混淆算法。通过在原始 CNF 公式中无缝的混入噪音公式，在隐藏原有的结构信息的同时，保持原有的 CNF 数据形式和解空间，从而复用目前已有的求解算法。这就在算法的实用性和隐私保护上取得了良好的折中。本文从理论上证明了算法的正确性并通过大量的仿真实验验证了算法的性能。

2. 解空间上估计的 CNF 混淆算法。由于 SAT 求解时，输出数据也包含了敏感信息。在研究了隐藏结构信息的 CNF 混淆算法之后，本文进一步研究了隐藏 CNF 解的混淆方法。针对输出信息保护，本文提出了解空间上估计的 CNF 混淆

算法。通过扩展噪音公式解空间，使得混淆后 SAT 问题的解空间为原始解空间的上估计。通过引入噪音解实现对原始解的隐藏。本文从理论上充分证明了算法的正确性，并通过大量的仿真实验验证了方法的有效性和性能。

3. 混淆后公式的求解效率是在混淆算法有效性的一个特别重要的指标，也是基于加噪的混淆算法区别于其他混淆算法的一个重要因素。由于加噪的过程改变了公式的内在结构，并且 SAT 问题自身特点，使得其问题复杂度会随着结构的变化出现跃变。针对这一情况，本文针对硬件验证中常用的 CNF 结构进行分析，提出了跃变敏感的混淆策略，使得混淆后的 CNF 公式求解难度不会大于原始公式求解难度。特别针对对偶综合这一 SAT 问题的实例算法进行了分析。从理论上验证了算法的可用性。

4. CNF 混淆算法的有效性评价。混淆算法保证混淆后的 CNF 公式可用已有的求解器求解，并且可以用较小的开销恢复出原始的解。但是除此之外，在开放环境下为保证 SAT 计算外包的顺利实施，混淆算法仍然需要满足其他的特性。结合程序混淆的有效性评价标准，本文抽象出 CNF 公式混淆的有效性评价标准。通过对混淆策略的细分，针对两种混淆策略进行定量分析和定性评价，为设计出更好的混淆策略提供了依据。

上述各部分研究内容之间的关系参见图 1.3。

1.6 论文组织结构

论文共分六章，组织结构如下：

第一章为绪论，介绍 SAT 求解的基本概念、特点、应用以及安全可验证计算的研究现状。分析 CNF 公式混淆算法的研究意义和挑战，并简述本文的研究内容和组织结构；

第二章为相关研究，对科学计算外包隐私保护、安全可验证计算以及程序混淆等相关概念进行了系统和全面的介绍，分析了现有工作的特点和适用性；

第三章研究 SAT 问题求解中输入数据的隐私保护问题，从实用性的角度出发，提出了基于加噪的 CNF 混淆算法。在保证求解算法和解空间不变的前提下，通过混入噪音变量和子句来实现 CNF 公式内部结构信息的隐藏；

第四章在前述开创性工作的基础上，针对高危险外包计算环境，针对可能出现的 ALLSAT 攻击，通过引入具有簇形解的噪声 CNF 公式，进一步提高混淆算法的鲁棒性；

第五章研究 SAT 问题求解中输出数据的隐私保护问题，提出了解空间上估计的 CNF 混淆方法，通过混入用户可剔除的噪声解来隐藏真实的解信息；另一方面，研究 SAT 问题求解中输入数据中结构信息的增强型隐藏方法，提出在感知原

有公式结构的基础之上，加入可构成合法结构的噪声变量和子句，来实现 CNF 公式内部结构的保真隐藏，以提高应对基于模式识别和同构检测等隐私攻击的防范能力；

第六章研究 SAT 问题混淆算法的有效性评价问题，希望通过对有效性标准的提取，为设计更为有效的混淆算法提供指导。

第七章总结全文并展望未来的工作。最后是致谢、博士期间撰写的论文、参加的科研工作以及参考文献。

第二章　相关研究

随着网格和云计算等开放计算环境从概念走向实用，针对科学计算在开放环境下安全外包的相关研究也逐渐兴起。科学计算的安全外包主要关注两方面的问题：一是计算数据的隐私性，二是计算结果的正确性。

计算的隐私保护、可验证计算和混淆是其中三个重要支撑技术。面向计算的隐私保护和可验证计算，聚焦于开放计算环境中输入输出数据的隐私和计算结果完整性保护；而混淆着眼于对部署于开放环境下的计算程序和电路自身的隐私保护。

2.1　外包计算的隐私保护研究

由于计算数据的隐私性涉及到商业秘密，成为了影响用户是否采用开放计算环境的决定性因素。近几年，随着云计算和网格计算在商业模式上的逐渐推广和普及，外包计算的隐私保护问题更加引起了学术界和工业界的关注。

按照采用的方法不同，计算数据的隐私保护可以分为基于同态加密 (homomorphic encryption) 的方法和基于数据伪装 (disguising) 的方法。伪装和加密的区别在于，伪装并不改变被外包的算法，而只是改变被外包的数据。而被外包的数据在经过伪装之后，仍然能够保持问题本身所需的某些特性，以便获得最终解。伪装方法依赖于特定问题和特定伪装算法的选择，对数据的保护程度缺乏一个统一的框架，但是对被外包算法性能影响小。数据转换和数据分割是两种最为常用的伪装方法，数据转换方法是通过一个可逆的运算将数据转换为外包数据，并使用逆运算从外包计算结果中恢复真实结果；而数据分割方法则通过将数据分散到不同的计算单元，使得第三方无法了解被外包问题的全部。

而同态加密方法则将被外包的数据和算法都映射到加密空间，并保证这种映射是同态的；该方法能够提供一个统一的框架来保证数据的安全性。但是同态性要求使得其实现开销非常高。下面将分别介绍这几种方法的相关研究。

2.1.1　基于同态加密的方法

同态加密技术对密文进行某些特定代数运算，而且保证对明文的运算结果进行加密得到的结果与该运算结果相同。同态加密技术一般包括密钥生成 (keygen)、加密 (encrypt)、求值 (evaluate) 和解密 (decrypt) 四个步骤：

1. 密钥生成算法：该算法输入安全参数，输出用户的公钥和私钥。

2. 加密算法：该算法输入用户的公钥和明文数据，输出相应的密文。

3. 求值算法：该算法输入用户的公钥、一个函数和一组密文，输出一个新密文。

4. 解密算法：该算法输入用户的私钥和密文，输出对应的明文数据。

根据可支持代数运算种类的不同，同态加密技术又可以分为部分同态加密和完全同态加密两种。其中，部分同态加密只能对密文进行某些运算，而完全同态加密方案则能够对密文进行任意运算。

部分同态加密系统在非加密空间 D 和加密空间 D' 之间，针对加法或者乘法函数 f，相应地构造加密空间上的运算函数 f'，以及对应的加密映射 E 和解密映射 E^{-1}，使得 $y = E^{-1}(f'(E(x))$。此时称 E 是 f 的一个部分同态加密映射。常见的部分同态加密系统包括 Unpadded RSA[50]，ElGamal[51]，Goldwasser-Micali[52] 和 Paillier[53] 等。

在实际应用需要对密文进行较为复杂的操作时，以上部分同态加密方案就无法满足这种需求。针对这一问题，Gentry[21] 提出了非常优雅的理论框架，以构造针对乘法和加法同时保持的全同态加密算法。该方法的基本思想在于使用一连串的基于噪声的近似同态加密。近似同态意味着，每一次加密仅能够在一定步数 S 内，而不是任意步数内，对加法和乘法运算同时保持同态特性。当在加密空间 D_i 中的运算步数达到 S 时，为了防止累积的噪声导致解码失败，该方法将中间计算结果 r_i，以及解密该结果的函数 d_i，一起映射到新的加密空间 D_{i+1}，并在其中运行解密函数 d_i 以得到中间结果 r_i。此时由于 r_i 处于 D_{i+1} 中，所以运行该运算的云代理将在不知道 r_i 的情况下，完成后续 S 步的运算，以得到新的中间结果 r_{i+1}。如此迭代，即可最终得到能够对加法和乘法同时同态的加密系统。

Gentry 等人在[54] 中给出了上述理论框架一个完整的参考实现，但运行结果显示该方案需要较大的时间和空间开销。Scholl 等人[55] 和 Stehle 等人[56] 分别改进实现方案，得到运行效率更高的完全同态加密方案。在上述方案的基础上，Smart 等人[57] 根据剩余定理，设计了密钥和消息长度都较小的新方案。Gentry 在文献[58] 中设计了新的密钥生成算法，将完全同态加密方案的安全性，建立在稀疏子集求和问题和理想格中最坏情况下的困难 (hardness) 问题之上。

为了有效地对不同用户的加密数据进行计算，Lopez-Alt 等人[59] 基于理想格，设计了一种允许多个密钥参与的完全同态加密方案。该方案比传统的完全同态加密方案更加灵活和实用，但其安全性依赖于一个非标准的假设。Bos 等人[60] 采用 Brakerski 等人[61] 提出的技术，消除了该假设。

上面的方案均是基于理想格或类似方法来构造的，存在不易理解的问题。因此 Van 等人[62] 设计了基于整数环的完全同态加密方案。Coron 等人[63, 64] 和 Chen

等人[65] 针对 Van 方案中的问题，提出多种的改进方案。Gu 等人[66–68] 也进行相关的研究。

LWE(Learning with errors)[69] 通过引入数量较小的错误作为噪音值，增加高斯消去法的求解难度。其难度为基于理想格的最坏情况。而基于环的 R-LWE(Ring-Learning with errors) 是一种构造复杂性适中，安全性和 LWE 相当的新算法。打包技术 (packed) 是通过 SIMD 方式对明文向量而不是对单个明文进行加密，以提高同态加密效率的一种方法。鉴于性能是完全同态实现方案中存在的最大问题，Gentry 等人在文献[70] 中采用将明文打包的方法，基于 R-LWE[71] 设计了一个时间开销仅为多项式对数的完全同态加密方案。随后在文献[72] 中又通过将模设置为 2 的幂的近似值，得到了一个效率更高的方案。Brakerski 等人[73–75] 利用 Peikert 等人[76] 的打包技术，设计了一种基于标准 LWE 问题和 R-LWE 问题的简单且安全性较高的完全同态加密方案，并在安全性和效率上进行了一系列改进的工作[77]。

针对 SAT 问题的隐私保护，Brakerski[49] 等人探讨了使用多线性映射方法和坡度编码策略对 d-CNF 进行混淆。该方法使用随机和带有噪声的编码，并提供测试过程来确保编码元素的等价。这种方法基于有限加速假设，并使用最原始的二叉树搜索方法求解混淆后的 CNF。由于无法利用目前经典的 SAT 求解器，因此计算开销仍然是制约其实用化的重要因素。

2.1.2 基于数据伪装的方法

按照对数据域的转换方式，数据伪装可分为数据域扩展和数据域分片两种。

数据域扩展使原始数据无缝地包含于伪装后数据中。典型的如基于可逆矩阵乘法的伪装。其原理是，将原始矩阵数据与可逆矩阵相乘。由于 n 阶矩阵都有 $n!$ 个置换矩阵，因此还原出原始矩阵的可能性为 $1/n!$。M. J. Atallah[78] 针对科学计算中常见的多种线性代数算法，将有待外包的数据与随机对角矩阵，进行矩阵乘。结果可以通过可逆的矩阵运算得到。该论文还讨论了问题域的扩展和缩减，以进一步伪装计算的真正企图。该方法的一个问题是没有讨论如何验证返回的结果的正确性。C. Wang[79] 继承了上述工作中对等式的伪装方法，并创造性的针对线性规划问题中的不等式和优化目标找到了安全而有效的伪装算法。同时该论文还讨论了如何验证返回结果的正确性。他们的方法是基于问题转换的，不需要引入额外的开销。但是，这些技术需要花费与计算负载成立方关系的时间，不适用于弱客户端系统，无法处理大规模问题。C. Wang 等人[80] 在大规模线性方程组的外包求解问题中，利用矩阵 -向量乘的代数属性，对结果以批处理方式进行验证。针对 SAT 问题的特点，Qin 在文献[81, 82] 提出了基于 CNF 混淆的 SAT 计算隐私保护算法。文献[81] 讨论了 SAT 计算中输入数据的隐私保护算法。通过在输入数据

中按照特定的规则混入噪音数据，在保持解空间不变的前提下，隐藏真实输入数据信息。文献[82] 进一步讨论了输出的隐私保护。通过对混淆后解空间的上估计扩展实现输出数据的隐藏，并从理论上证明了混淆算法的正确性。

对数据域进行分片计算，使得计算者无法获得全局数据也是一种有效的数据伪装方法。M. J. Atallah[78, 83] 部分的借鉴了加密算法的思想。同时分别针对序列运算和代数计算的特点，将被外包的问题划分为两个子集，并外包到多个不存在合作关系的代理中，保证每个代理都无法了解被外包问题的全部。在[84] 中，M. J. Atallah 放弃了非合作代理假设，允许多个敌意代理之间通过使用机密共享机制 [85] 交换信息，进行协同求解。而每一个代理都不能获得待求解问题的整体信息。然而该机制也导致了通讯开销的急剧增长。另外，这些协议都是在假定非共谋 (non-colluding) 服务器的情况下作出，无法防御共谋 (colluding) 攻击。Yuriy Brun[13] 等人则使用了 stile 数据分布的模式，通过将数据条块计算，提高攻击者获得完整计算数据的难度，以此降低数据被窃取的可能性。该研究主要针对 SAT 问题的求解，通过将 CNF 子公式分布在多个计算节点上，防止 CNF 公式泄露。该方法目前也仅仅支持简单的二叉遍历赋值求解方法，无法利用已有的 SAT 求解加速算法，因此计算开销仍然是进行大规模的 SAT 问题求解的主要障碍。

2.2 计算完整性验证研究

如果说计算数据的隐私性是外包之前需要关注的重要问题，外包计算结果的正确性则是计算外包结束之后必须确认的事情。由于网格计算和云计算环境下，计算外包至云端来执行，主要计算过程均在客户无法掌控的环境中进行。网格计算节点和云端服务器恶意或无心的错误都会对用户的计算结果造成影响。

Du 等人[8] 指出在早期出现的开放计算环境——志愿计算模式下，部分参与计算的志愿者会给出错误结果。大规模的统计表明，给出错误结果的志愿计算者可分为三类：懒惰欺骗者，私心欺骗者和恶意欺骗者。其中懒惰欺骗者希望少干活多获得报酬，为节约计算成本，不会进行全部计算，因而仅能给出部分结果甚至不会返回结果；私心欺骗者会进行全部计算，但会希望私留部分结果，对用户进行瞒报或仅仅将部分正确结果返回给用户；恶意欺骗者则不进行计算或者进行错误计算，而将错误结果返回给用户。

开放计算环境下这些威胁，催生了可验证计算的研究。可验证计算的目标是确保一个弱计算能力的客户，都可以验证不可信服务器上计算的完整性。其中计算完整性不仅包括计算结果的在数值上是准确的，还包括计算结果在数量上是完整的。

多副本技术是容错的传统技术，主要利用冗余计算的思想。该技术通过将同一个任务部署在多个计算节点同时进行，并对计算结果进行评估，超过半数的相

同结果将被采纳。由于该类方法简单易行，在实际应用中最为广泛。但是多副本计算存在开销大的问题，并且无法防止节点共谋的情况。

为此，研究者提出了基于抽样验证的技术。抽样验证技术是指用户指定抽测样本，由计算者进行计算，而后用户检测样本的计算结果是否正确，以此来判断此节点所有结果的正确性。Du 等人[86] 提出了基于提交的抽样，在计算者提交结果之后进行抽样复算。由于此时计算者已经不能改变计算结果，这就迫使计算者要提供完整的正确结果。

Golle 等人[87] 利用单向函数的特点，在参与者所处理的数据域中随机选取部分数值并计算其单向函数值，要求参与者给出这些函数值对应的数值。由于单向函数的难解性，参与者无法由函数值直接推出数值，因此必须遍历数据域，完成所有的计算作业。通过上述方法防止恶意以及懒惰工人的欺骗行为。Szajda 等人[88] 扩展了上述策略，针对任务优化的计算外包以及 MonteCarlo 方法的计算外包，通过检测返回结果中探针结果的正确性，来检测是否存在恶意欺骗和懒惰欺骗的情况。M Blanton[89] 在生物实验数据中随机插入用于结果校验的特定数据模式，并在计算完成后检查结果是否包含由这些模式导致的特定要求。

Du 等人[8] 为了防止私心欺骗者，在待计算的实例中混入一定比例的 chaff 实例。由于 chaff 与计算实例在形式上难以区分，使得计算节点无法确定计算结果是真实结果还是 chaff 结果。由于 chaff 实例结果预先已知，因而迫使计算节点返回所有的计算结果。Du 针对子图同构检测和 SAT 问题分别给出了 chaff 实例的构造方法。

云计算作为一种近期出现的开放计算环境新模式，其在大规模数据上的出色表现，也对计算结果的完整性提出了更高的要求。Wang 等[90] 针对大规模数据处理架构 Mapreduce 的运算特点，结合复算和验算技术，实现计算结果完整性检测。结合云计算模式的细分和大数据处理的需求，Wang 等针对混合云下的大数据处理计算的结果完整性验证进行了一系列研究[91, 92]。但是上述的研究未考虑到云计算中的数据隐私保护问题。

针对云计算大规模商业普及所带来的隐私保护和结果可信性的要求，R. Gennaro[93] 提出安全可验证计算的概念，给出了基于完全同态加密[94] 和加密电路 (Garbled-Circuit)[95] 实现可验证的安全计算外包的方案。方案不仅考虑对结果正确性的要求，数据的隐私性更是关注的重点。但是这种方案只是给出了理论上的可行性，其构造的效率受限于完全同态加密方案的效率，因此具体的应用实现还处于探索中。

上述的工作主要是用于结果验证的目的，但加入加密电路 (garbled circuit) 或是 chaff 实例等探针进行隐藏计算对本文的工作有一定的启发。

2.3 混淆技术研究

混淆[96, 97]是一种可隐藏设计中的隐私，并使得设计更加难以理解的技术，在软硬件防盗版、防信息泄露等领域得到广泛的应用。具体说来，混淆通过将程序或电路变形，并保证变形前后程序或电路的功能不变，达到隐藏或嵌入敏感信息的目的。根据其应用领域，混淆又可细分为程序混淆、电路混淆。

2.3.1 混淆理论研究

混淆理论的研究始于 2001 年，Barak 等人在[98]中首次给出了混淆的概念。其思想是将程序或电路变形，同时保证变形前后程序和电路的输入输出不变，并使变形后的程序或电路看起来类似一个虚拟黑盒 (virtual black box)，这一概念被称为黑盒混淆 (Blackbox obfuscation)。在他们确定的最初定义中，黑盒混淆需要满足下列特性：

1. 被混淆的电路和原始电路的功能相同，且至多有多项式的开销损耗。

2. 除了黑盒功能 (输入输出)，被混淆的电路不能泄露任何信息。

从形式上看，Barak 提出的黑盒混淆具有如下表现：混淆后电路可以进行有效计算，但仅可以通过电路的输入输出访问计算。悲观的消息是，Babark 等人的研究表明：不存在通用的黑盒混淆算法。

结合黑盒混淆的困境，Barak 等人[96, 98]提出了不可区分混淆 (indistinguishability obfuscation) 的概念。不可区分混淆给出了如下的要求：给定两个尺寸相同的等价电路 C_0 和 C_1，他们的混淆结果计算时不可区分。随后，Lynn[99]等人给出了关于程序混淆的好消息：点和多点类的函数可以通过随机 oracle 模型进行混淆。

混淆最常被应用于知识产权保护领域，如防止对软件程序进行反向工程，防止硬件 IP/IC 核非法使用和复制。在 IP 核的保护中，由于混淆后网表会被交付出去，并且其功能也是公开的，将其看作是一个黑盒显然不合适。基于上述实际的情况，Goldwasser 等人[100, 101]提出基于放松要求的最佳可能混淆 (Best Possiable Obfuscation)，并研究了其中的特性。最佳可能混淆定义放松了 Barak 定义中要求的第二个特性，即放弃了黑盒假设，而将其表述为不会比其他任何相同功能的电路泄露更多信息。直观的看，最佳可能混淆保证任何不被混淆电路隐藏的信息，也无法被其他具有相同功能的电路隐藏；这个放松条件并不绝对保证混淆电路可以隐藏信息，但保证该混淆电路是所有电路中在信息隐藏方面做得最好的。

随着云计算的发展，对外包计算的加密需求进一步激发了实用性混淆技术的研究。在混淆理论研究的基础上，Sanjam 等人[102]构造了一个不可区分混淆的参

考实现，该参考实现可以用于所有电路；他们还研究了该参考实现在函数加密上的应用。Brakerski 等人[103] 结合多线性映射和坡度编码，给出了 DNF 公式混淆的参考实现。采用相同的编码方法和线性映射，Brakerski 等人[49] 还构造了 d-CNF 公式黑盒混淆的参考实现，并证明了其安全性。

混淆理论研究上的进展，已经由从对概念与特性的界定，发展到参考实现的设计。但是，目前给出的参考实现均基于加密原语来构造，开销较大。因此，理论研究上的成果还未在程序和电路保护中得到实际的广泛应用。

2.3.2 程序混淆

在实际应用中，程序混淆的主要目标是增加反向工程的难度，因此更多采用一些轻量级的混淆策略。通过保持语义的程序转换使得代码难以理解，尽可能迫使攻击者因解混淆开销过大而放弃对代码的剽窃。在程序混淆方面，最普遍的应用是保护 Java 字节码。其目的是，通过混淆提高 Java 代码的理解难度[104]。

程序混淆的典型方法[97] 包括:

1. 词法混淆 (Lexical obfuscation)：对程序中的标示符使用重命名技术，降低其可读性。

2. 数据混淆 (Data obfuscation)：使用数据编码、变量数组的划分和融合 (split and merge)，变量重排序以及继承关系修改等手段，改变程序中的数据结构。

3. 控制混淆 (Control Obfuscation)：改变程序中的语句、循环和控制语句的顺序，使用无关的条件语句隐藏控制流的实际走向。

4. 妨碍混淆 (Prevention obfuscation)：使用动态派发 (dynamic dispatching) 方法，防止解编译器 (decompilers) 编译出原始程序。针对 Java 代码，具体的方法有：将分支转换为跳转指令，不遵循构造器规范，将 try 语句和 catch 语句块融合，以及混淆字节码解释器。

前三类混淆方法主要专注于降低程序的可阅读性，针对的是人类攻击者；而第四类混淆方法主要用于预防自动解混淆 (automatic deobfuscation) 的攻击。

针对混淆的有效性，文献[105] 针对代码混淆的最初要求，指出评价混淆有效性的四个维度：效能 (potency)，适应力 (resilience)，开销 (cost) 和隐形性 (stealth)。其中，效能衡量混淆引入的复杂性，复杂性越高，意味着程序越难以理解；适应力衡量抵抗自动解混淆器攻击的能力；开销衡量混淆程序的时空复杂性；隐形性衡量了混淆前后程序的相似性。该评价标准对我们的工作起到了指引的作用。

由于 CNF 公式是电路结构和软件程序的抽象，因此针对衡量程序混淆有效性的部分指标，也适用于衡量 CNF 混淆。特别指出的是，效能标准针对程序的可理解性，而 CNF 公式本身就具有难以理解的特点，因此我们采用后三个评价标准。

2.3.3 电路混淆

硬件计量 (Hardware meter)[106] 和硬件水印 (Hardware watermark)[107] 是防止硬件盗版的两类电路混淆技术。和程序混淆主要用于隐藏程序自身信息的目标不同，电路混淆主要目标是在硬件 IC 或 IP 中植入要隐藏的信息 (例如水印)，防止非法使用或非法复制。

2.3.3.1 硬件计量

硬件计量亦称为集成电路计量，是由 Koushanfar 等人[108] 于 2001 年首次提出的概念。其思想是通过将一小部分设计变为可编程状态，从而为集成电路的功能赋予一个唯一的标记。其目的是使设计者 (知识产权拥有者) 对集成电路设计的后期投片仍然具有控制权。硬件计量的研究工作主要集中于，利用制造可变性，为每个集成电路产生随机的身份标记，以便于获得计量。

按照是否对功能进行主动控制，硬件计量可分为主动方法和被动方法两类。早期研究[106, 108–110] 关注被动方法，主要是研究如何为集成电路植入标记，而不涉及对电路功能的改变。Alkabani 等人[111, 112] 给出了主动的硬件计量策略。该策略利用了物理上不可克隆函数 (PUF) 为每一个集成电路产生唯一初始 FF 值 (开机状态)。开机状态有较高的可能性成为增强有限状态机的一部分，并导致集成电路处于锁住状态；仅有增强有效状态机的设计者或被授权者，可以使用密钥 (转化为合法的 reset 状态) 来解锁集成电路。Li 等人[113] 提出利用 retiming、resynthesis、sweep 和 conditional stuttering 四类电路结构变换操作，来实现顺序电路的最佳可能混淆。上述基于嵌入锁的方法，被称为内部主动集成电路计量。相对于内部主动方法，外部主动集成电路计量方法[114–116] 将锁嵌入到设计的物理层，由外部的加密函数进行控制的。由于加锁使用了复杂的加密模块，因此通常会导致较大的功耗开销和面积开销。

2.3.3.2 硬件水印

硬件水印技术[107] 的研究最早始于 1999 年，其目标是标注硬件所有权。由于水印会导致某些非水印电路中不会出现的输入转换行为，硬件设计者可以此来证实所有权。和硬件计量不同的是，水印的加入对集成电路的功能没有影响。Oliveira 等人[117] 首先给出了在顺序电路中隐藏秘密水印的方法。在该方法中，水印是通过修改状态迁移图 (STG) 实现的。使用特定输入集合 (密钥) 来穿透状态转换选择路径，达到加入特定不可觉察信息的目的。Koushanfar[118] 给出了加入多

个水印来加强安全性的方法。他们的工作同时表明，在状态迁移图中隐藏多个水印，是使用一般化输出来混淆多点函数的一个实例。Yuan 等人[119] 给出了使用冗余的有限状态机来隐藏水印的方法。他们还设计了一个基于 SAT 的算法，来发现给定最小化 FSM 中最大冗余转换集合，并且利用这些冗余来隐藏 FSM 信息，并且不改变给定最小 FSM。

硬件水印和被动硬件计量看上去相似，但有一些本质区别：计量通常为每个集成电路赋予一个特殊的标记，同一个产品的所有集成电路中的水印是相同的。因此，水印无法跟踪由同一模具产生出来的拷贝。

上述的技术对不受控制的软件产品和硬件设计进行主动保护或被动追踪，对于 CNF 公式的保护也有借鉴价值。

2.4 本章小结

本章综述了 SAT 问题及其隐私的基本概念和相关定义。对与本文研究工作相关的计算外包隐私保护、可验证计算、以及程序电路混淆等问题进行了概述，并对这些问题中已有的研究工作进行了具体的介绍和分析。

第三章　基于余因子和 Craig 插值的迭代特征化算法

3.1　引言

在形式化验证和综合领域，对于两个存在某种内在联系的逻辑向量 $\vec{a}$ 和 $\vec{b}$，有两种不同的方式表达他们之间的联系：关系和函数。

其中，关系 $R(\vec{a},\vec{b})$ 更具一般性，能够表达 $\vec{a}$ 和 $\vec{b}$ 之间的任意对应，尤其是一对多的对应，这是关系比函数具有更强描述能力的地方。这种一般性在形式化验证中广泛用于描述非确定性行为以扩展描述能力[120]，以及构造抽象模型[121] 以削减计算复杂性等。

而另一方面，函数是关系的一种受限形式。如果 $R(\vec{a},\vec{b})$ 满足以下要求，则能将其转换为相应的函数 $\vec{b} := f(\vec{a})$：对 $\vec{a}$ 的任意取值 $x \in [\![\vec{a}]\!]$，均存在且仅存在唯一的 $y \in [\![\vec{b}]\!]$，使得 $R(x,y)$。

以图3.1为例。对于图3.1a) 中的一对一映射，我们可以使用布尔函数 $y_1 = x_1 \wedge x_2$ 和 $y_2 = \neg x_1 \wedge \neg x_2$ 表示。而另一方面，对于图3.1b) 中的布尔关系，并不存在相应的布尔函数，因为 $(x_1, x_2) = (0,1)$ 的情形被映射到了多个 (y_1, y_2) 的组合。而这种情况可以使用如下布尔关系表示：

$$R = \left\{ \begin{array}{ll} & (\neg x_1 \wedge \neg x_2 \wedge \neg y_1 \wedge y_2) \\ \vee & (\neg x_1 \wedge x_2 \wedge \neg y_1 \wedge \neg y_2) \\ \vee & (\neg x_1 \wedge x_2 \wedge y_1 \wedge y_2) \\ \vee & (x_1 \wedge \neg x_2 \wedge \neg y_1 \wedge \neg y_2) \\ \vee & (x_1 \wedge x_2 \wedge y_1 \wedge \neg y_2) \end{array} \right\} \tag{3.1}$$

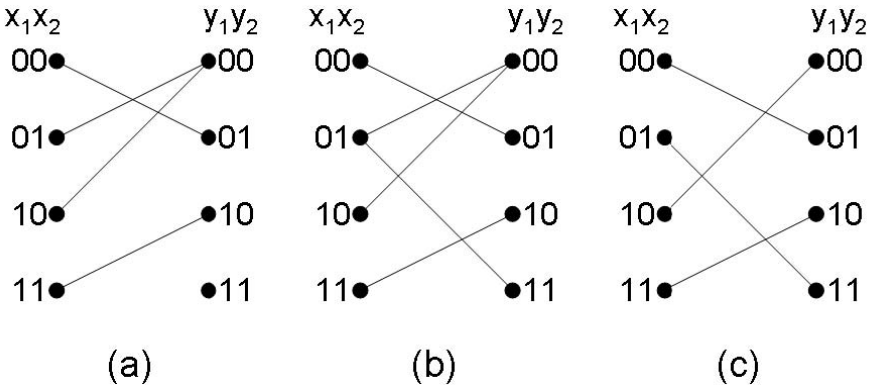


图 3.1　关系和函数的布尔映射

在实际的软硬件设计与验证领域，存在大量的情况需要从一个关系中获得相应的函数。如在自动激励生成算法中从约束描述产生相应的激励函数[122]，证明导引抽象中的抽象模型构造[123]，以及本文中推导控制流谓词和特征化解码器等。

为此，我们在本章中提出了基于余因子 (Cofactor)[34] 和 Craig 插值[37] 的迭代特征化算法，以从布尔关系 $R(\vec{a},\vec{b},t)$ 中特征化函数 $t = f(\vec{a})$。

在本章中，我们首先在小节3.2描述 Craig 插值的基本原理，然后在小节3.3中将其应用至一种受限的特殊情况下以求解特征化函数。最后在小节3.4中通过迭代调用上述受限算法，以将其扩展到包含额外变量的更一般情形。

3.2 Craig 插值的原理和实现

注意本小节描述的 Craig 插值原理不是我们的原创，而是为了方便上下文的叙述从[38] 移植至此。

3.2.1 相关背景知识和记法

在通常的 SAT 求解器，包括本文使用的 MiniSat[25] 中，要求待求解的公式被表示为 CNF 格式。其中一个公式是多个短句的合取 (conjunction)，而每一个短句是多个文字的析取 (disjunction)，而每个文字是一个布尔变量 v 或者其反 $\neg v$。如公式 $(v_0 \vee \neg v_1 \vee v_2) \wedge (v_1 \vee v_2) \wedge (\neg v_0 \vee v_2)$，包含短句 $v_0 \vee \neg v_1 \vee v_2$，$v_1 \vee v_2$ 和 $\neg v_0 \vee v_2$。而短句 $v_0 \vee \neg v_1 \vee v_2$ 包含文字 v_0，$\neg v_1$ 和 v_2。

当存在一个变量 v，使得一个短句 c 中同时包含两个文字 v 和 $\neg v$，则称 c 为 tautological 的。我们通常假设有待 SAT 求解器求解的公式中所有的短句都是非 tautological 的。

假设公式 F 的布尔变量全集为 V。若存在对 V 的赋值函数 $A : V \to \{0,1\}$，使得 F 中的每个短句均能取值为 1，则称 F 是可满足的，此时 SAT 求解器能够找到赋值函数 A。否则称 F 为不可满足的，此时 SAT 求解器能够产生如下一小节所述的不可满足证明。

3.2.2 不可满足证明

对于两个短句 $c_1 = v \vee B$ 和 $c_2 = \neg v \vee C$，当 $A \vee B$ 是非 tautological 的时，$A \vee B$ 称为它们的 **resolvant**。而 v 称为它们的 **pivot**。易知以下事实：

$$\begin{aligned} resolvant(c_1, c_2) &= \exists v, c_1 \wedge c_2 \\ c_1 \wedge c_2 &\to resolvant(c_1, c_2) \end{aligned} \tag{3.2}$$

定义 3.1: 对于不可满足公式 F，假设其短句集合为 C，则其不可满足证明 Π 是一个有向无环图 (V_Π, E_Π)，其中 V_Π 是短句集合，而 E_Π 是连接 V_Π 中短句的有向边集合。Π 满足如下要求：

1. 对于节点 $c \in V_\Pi$：

   (a) 要么 $c \in C$，此时称 c 为 Π 的根

   (b) 或者 c 有且仅有两个扇入边 $c_1 \rightarrow c$ 和 $c_2 \rightarrow c$，使得 c 是 c_1 和 c_2 的 resolvant。

2. 空短句是 Π 的唯一一个叶节点。

直观的说，Π 就是一棵树，以短句集合 C 的子集为根，以空短句为唯一叶节点。而每个节点 c 的两个扇入边 $c_1 \rightarrow c$ 和 $c_2 \rightarrow c$ 代表了一个 resolving 关系 $c := resolvant(c_1, c_2)$。

包括本文使用的 MiniSat 求解器[25] 在内的许多 SAT 求解器，当公式不可满足时都将产生一个不可满足证明 Π。

3.2.3 Craig 插值算法

根据文献[37]，给定两个布尔逻辑公式 A 和 B，若 $A \wedge B$ 不可满足，则存在仅使用了 A 和 B 共同变量的公式 I，使得 $A \Rightarrow I$ 且 $I \wedge B$ 不可满足。I 被称为 A 针对 B 的 Craig 插值[37]。

目前最常见且最高效的产生 Craig 插值的算法是 McMillan 算法[38]。其基本原理描述如下。

对于上述公式 A 和 B，已知 $A \wedge B$ 不可满足，而 Π 是 SAT 求解器给出的不可满足证明。当一个变量 v 同时出现在 A 和 B 中时，我们称其为全局变量。若 v 只出现在 A 中，则称其为 A 本地变量。

对于文字 v 或者 $\neg v$，当变量 v 是全局变量或者 A 本地变量时，称该文字为全局文字或者 A 本地文字。

对于短句 c，令 $g(c)$ 为 c 中所有全局文字的析取，而 $l(c)$ 为 c 中所有 A 本地文字的析取。

例如，假设有两个短句 $c_1 = (a \vee b \vee \neg c)$ 和 $c_2 = (b \vee c \vee \neg d)$。并假设 $A = \{c_1\}$ 和 $B = \{c_2\}$。则 $g(c_1) = (b \vee \neg c)$，$l(c_1) = (a)$，$g(c_2) = (b \vee c)$，$l(c_2) = FALSE$。

定义 3.2: 令 (A, B) 为一对公式，而 Π 是 $A \wedge B$ 的不可满足证明，且其唯一叶节点是空短句 $FALSE$。对于每一个节点 $c \in V_\Pi$，令 $p(c)$ 为如下定义的一个公式：

1. 如果 c 是根节点则

   (a) 如果 $c \in A$ 则 $p(c) = g(c)$

   (b) 否则 $p(c) = TRUE$

2. 否则令 c_1 和 c_2 分别是 c 的两个扇入节点，而 v 是他们的 pivot 变量

   (a) 如果 v 是 A 本地变量，则 $p(c) = p(c_1) \vee p(c_2)$。

   (b) 否则 $p(c) = p(c_1) \wedge p(c_2)$。

上述定义3.2是构造性的，已经给出了从不可满足证明 Π 得到最终的 Craig 插值的算法，即以 Π 的根节点为起点，为每一个 c 计算相应的 $p(c)$，直至到达最终的唯一叶节点 $FALSE$。我们有以下定理：

定理 3.1: 定义3.2为唯一叶节点 $FALSE$ 产生的 $p(FALSE)$ 即为 A 相对于 B 的 Craig 插值。

该定理的详细证明可见文献[39]。

计算 A 相对于 B 的 Craig 插值的时间复杂性为 $O(N + L)$，其中 N 是 Π 中包含的节点个数 $|V_\Pi|$，而 L 是 Π 中的文字个数 $\Sigma_{c \in V_\Pi}|c|$。而所产生的插值可以视为一个电路，其空间复杂性为 $|O(N + L)|$。当然，Π 的尺寸在最坏情况下也是 $A \wedge B$ 的尺寸的指数。

3.3 非迭代的特征化算法

假设有布尔关系 $R(\vec{a}, t)$ 使得 $R(\vec{a}, 1) \wedge R(\vec{a}, 0)$ 不可满足。而我们需要从 R 中特征化函数 f，使得 $t = f(\vec{a})$。则根据上述的讨论，可以简单的令 $A = R(\vec{a}, 1)$ 而 $B = R(\vec{a}, 0)$。此时 A 相对于 B 的 Craig 插值 Π 具有以下性质：

1. $R(\vec{a}, 1) \rightarrow \Pi$，这说明 Π 覆盖了所有能够使得 $R(\vec{a}, 1)$ 可满足的 $[\![\vec{a}]\!]$。

2. $R(\vec{a}, 0) \wedge \Pi$ 不可满足，这说明 Π 没有覆盖任何使得 $R(\vec{a}, 0)$ 可满足的 $[\![\vec{a}]\!]$。

3. Π 仅引用了 $R(\vec{a}, 0)$ 和 $R(\vec{a}, 1)$ 的共同变量集合 $\vec{a}$。这说明 Π 是一个定义在 $\vec{a}$ 上的函数。

因此，Π 即为函数 f。

该算法在本文的后继章节中被广泛应用于构造解码器的布尔函数。

算法 3.1 *CharacterizingFormulaSAT*$(R, \vec{a}, \vec{b}, t)$: 特征化使得 $R(\vec{a}, \vec{b}, 1)$ 可满足的 $\vec{a}$ 集合

$FSAT_R(\vec{a}) := 0$;
while $R(\vec{a}, \vec{b}, 1) \wedge \neg FSAT_R(\vec{a})$ 是可满足的 **do**
    假设 $A : \vec{a} \cup \vec{b} \cup \{t\} \rightarrow \{0, 1\}$ 是可满足赋值函数;
    $\phi_A(\vec{a}) := R(\vec{a}, A(\vec{b}), 1)$;
    $\phi_B(\vec{a}) := R(\vec{a}, A(\vec{b}), 0)$;
    假设 $ITP(\vec{a})$ 是 ϕ_A 针对 ϕ_B 的 Craig 插值;
    $FSAT_R(\vec{a}) := ITP(\vec{a}) \vee FSAT_R(\vec{a})$;
end while
return $FSAT_R(\vec{a})$

3.4 迭代的特征化算法

上一小节讨论了如何从关系 $R(\vec{a}, t)$ 特征化函数 $t = f(\vec{a})$。然而在更一般的情形下，我们需要从 $R(\vec{a}, \vec{b}, t)$ 特征化函数 $t = f(\vec{a})$。相比之下，此时多了一个需要进行存在性量化的 $\vec{b}$。为此我们需要将上述算法进行以下扩展。

假设 $R(\vec{a}, \vec{b}, t)$ 是一个使得 $R(\vec{a}, \vec{b}, 0) \wedge R(\vec{a}, \vec{b}, 1)$ 不可满足的布尔公式。

其中 $\vec{a}$ 和 $\vec{b}$ 被分别称为重要和非重要变量子集。而 t 是目标变量。我们进一步假设 $R(\vec{a}, \vec{b}, t)$ 是可满足的。

我们需要特征化一个布尔函数 $FSAT_R(\vec{a})$，覆盖且仅覆盖了所有能够使得 $R(\vec{a}, \vec{b}, 1)$ 可满足的 $\vec{a}$。形式化的定义是：

$$FSAT_R(\vec{a}) := \begin{cases} 1 & \exists \vec{b}.R(\vec{a}, \vec{b}, 1) \\ 0 & otherwise \end{cases} \tag{3.3}$$

因此，一个计算 $FSAT_R(\vec{a})$ 的简单算法是：逐一遍历并收集所有使得 $R(\vec{a}, \vec{b}, 1)$ 可满足的 $\vec{a}$ 的赋值。然而该算法需要处理 $2^{|\vec{a}|}$ 种情况。对于很长的 $\vec{a}$，时间开销将会很大。

使用余因子化 (cofactoring) [34] 和 Craig 插值[38]，可以将每一个 $\vec{a}$ 扩展为一个更大的集合，从而极大的提高算法运行速度。直观的，假设 $R(\vec{a}, \vec{b}, 1)$ 的一个满足赋值是 $A : \vec{a} \cup \vec{b} \cup \{t\} \rightarrow \{0, 1\}$，通过 cofactoring[34] 可以构造以下公式：

$$R(\vec{a}, A(\vec{b}), 1) := R(\vec{a}, \vec{b}, 1) \wedge b \equiv A(b) \tag{3.4}$$

$$R(\vec{a}, A(\vec{b}), 0) := R(\vec{a}, \vec{b}, 0) \wedge b \equiv A(b) \tag{3.5}$$

因为 $R(\vec{a}, A(\vec{b}), 0) \wedge R(\vec{a}, A(\vec{b}), 1)$ 是不可满足的，$R(\vec{a}, A(\vec{b}), 1)$ 针对 $R(\vec{a}, A(\vec{b}), 0)$ 的 Craig 插值 $ITP(\vec{a})$ 可以用作 $\vec{a}$ 使得 $R(\vec{a}, A(\vec{b}), 1)$ 可满足的上估计。同时，$ITP(\vec{a}) \wedge R(\vec{a}, A(\vec{b}), 0)$ 是不可满足的，因此 $ITP(\vec{a})$ 没有覆盖任何使得 $R(\vec{a}, A(\vec{b}), 0)$ 可满足的情况。因此，$ITP(\vec{a})$ 覆盖且仅覆盖了所有使得 $R(\vec{a}, A(\vec{b}), 1)$ 可满足的 $\vec{a}$。

基于上述讨论，我们提出了算法3.1 以特征化等式 (3.3) 中的 $FSAT_R(\vec{a})$。行2检测是否仍然存在尚未被 $FSAT_R(\vec{a})$ 覆盖，且使得 $R(\vec{a}, \vec{b}, 1)$ 可满足的 $\vec{a}$ 。行4 和5 将可满足赋值中 $\vec{b}$ 的取值分别赋予 $R(\vec{a}, \vec{b}, 1)$ 和 $R(\vec{a}, \vec{b}, 0)$ 。这将使得 $\vec{b}$ 不再出现在这两个公式中。

因此，$\phi_A \wedge \phi_B$ 在行6 是不可满足的。且 ϕ_A 和 ϕ_B 的共同变量是 $\vec{a}$。因此可以使用 McMillian 算法[38] 计算 Craig 插值 $ITP(\vec{a})$。

$ITP(\vec{a})$ 将在行7被加入 $FSAT_R(\vec{a})$ 并在行2 被排除。

算法3.1 的每一个循环将向 $FSAT_R(\vec{a})$ 中加入至少一个 $\vec{a}$ 的赋值。这意味着 $FSAT_R(\vec{a})$ 覆盖了 $\vec{a}$ 的一个有界并且单调增长的赋值集合。因此算法3.1 是停机的。

3.5 可选的 BDD 整理和化简

上述由迭代特征化算法产生的函数，本质上是一系列 Craig 插值结果的析取。然而，正如小节3.3所指出的，Craig 插值结果是一个包含大量自由组合的与门、或门和反相器的电路，结构非常不规则。

对于不同的应用需求，上述问题的影响各不相同。

1. 对于特征化解码器的情形，由于这些解码器在进行物理实现的时候，还需要经由 Design Compiler[124] 这样的逻辑综合器进行处理，而这些工具具有非常强大的优化能力。因此不规则的 Craig 插值结果并不会对此类应用产生不良影响。

2. 然而对于特征化控制流谓词这样的应用，产生的结果需要由人类工程师手工检查，因此对可读性的要求非常高。

针对后一种应用，我们提出了基于 BDD[42] 的整理算法。利用 BDD 本身的规范性 (canonical)，首先将 Craig 插值结果转换为 BDD，然后使用 CUDD 工具包[125] 提供的遍历功能，遍历并导出每个 Cube，形成多个合取项的析取形式。

该方法的具有其独特的优点和缺点，并进而导致了对其应用场合的不同选择：

1. 该算法能够极大的化简 Craig 插值结果的表达式。因此通常用于特征化下一章的控制流谓词。

2. 另一方面，在处理包含大量异或门的电路时，BDD 很容易出现组合爆炸问题，从而导致运行算法的机器内存溢出。因此，我们在特征化解码器的时候并不调用该算法。

3.6 本章小结

本章综述了 Craig 插值算法的原理及其实现，以及基于其实现的迭代式特征化算法。相对于传统的状态空间遍历算法，该算法能够极大的提升遍历状态空间的能力。

本章算法将在本文的剩余部分被其他算法频繁调用，包括解码器特征化和流控谓词推导。

第四章　面向流控机制的对偶综合

4.1　引言

在通讯和多媒体芯片设计中，一个最为困难且容易出错的工作就是设计该协议的的编码器和解码器。其中编码器将输入变量 $\vec{i}$ 映射到输出变量 $\vec{o}$. 而解码器则从 $\vec{o}$ 中恢复 $\vec{i}$。对偶综合算法 [126–134] 通过自动生成特定编码器的解码器以降低该工作的复杂度并提高结果的可靠性。该算法的一个前提假设是，编码器的输入变量 $\vec{i}$ 总能够被输出变量 $\vec{o}$ 的一个有限长度序列唯一决定。

然而，许多高速通讯系统的编码器带有流控机制[135]，而该功能直接违反了上述假设。图4.1a) 展示了一个带有流控机制的通讯系统的结构。其中一个传输器 (transmitter) 和一个接收器 (receiver) 通过一个编码器 (encoder) 和一个解码器 (decoder) 连接在一起。从传输器到编码器有两个输入变量：有待编码的数据位 d，和代表 d 的有效性的有效位 f。图4.1b) 给出了编码器如何将 f 和 d 映射到输出变量 $\vec{o}$ 的编码表。

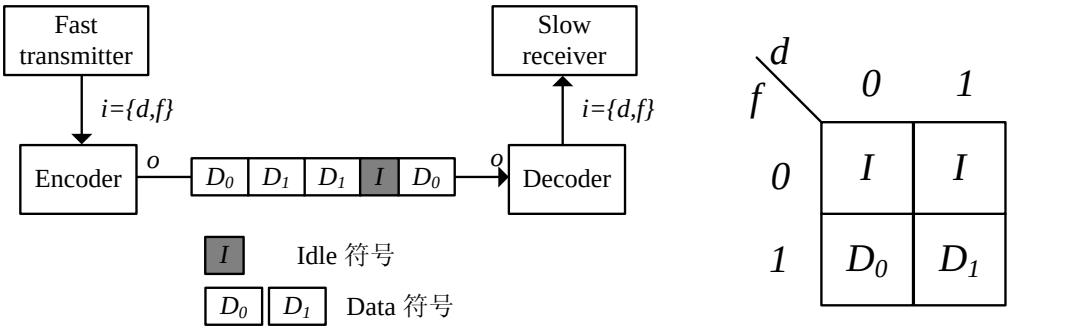
该流控机制的工作原理为：

1. 当接收器能够跟上发送器的速度时，发送器将 f 设为 1，这使得编码器按照 d 的值发送 D_d。从图4.1b) 的编码表可知，解码器总能够根据 D_d 恢复 f 和 d。

2. 而当接收器无法跟上发送器的速度时，发送器将 f 设为 0 以阻止编码器继续发送 D_d，转而在不考虑 d 的情况下发送空闲符号 I。而解码器应当识别并淘汰 I，并将 $f \equiv 0$ 发送给接收器。此时 d 的具体值并不重要。

上述流控机制能够防止快速发送器发送过多数据以至于接收器无法处理。然而该机制违反了迄今为止所有对偶综合算法 [126–134] 的基本假设，因为 d 无法被 I 唯一决定。很明显，为了解决该问题，我们只需考虑 $f \equiv 1$ 的情形，因为在此情况下 d 是能够被唯一决定的。而对于 $f \equiv 0$ 的情况，d 是无意义的，可以无需考虑。

基于上述讨论，本文提出了首个能够处理流控机制的对偶综合算法。该算法分为三步：首先，该算法使用算法4.2发掘所有能够被唯一决定的输入变量，并将它们作为流控向量 $\vec{f}$。其次，该算法在 $\vec{f}$ 上特征化一个流控谓词 $valid(\vec{f})$，使得剩余的所有输入变量均能被输出唯一决定。最后，该算法调用 Craig 插值算法[38] 产生解码器的布尔函数。



a) 快速transmitter和慢速的receiver被编码器和解码器链接起来

b) 编码器的映射表将 *d*和 *f* 映射到*o*

图 4.1　带有流控机制的通讯系统及其编码器

该算法的第二步类似于 Shen et al.[131]，因为他们都试图得到使得 $\vec{d}$ 或者 $\vec{i}$ 被唯一决定的谓词。然而他们的根本区别在于文献 [131] 的算法推导的是一个全局谓词，必须在整个展开的迁移关系序列上都成立。而本文算法得到的是仅应用于当前步的谓词，以恢复 $\vec{d}$。因此本文算法可以视为文献[131] 的一般化。

实验结果表明，对于多个来自于工业界的复杂编码器 (如以太网 [136] 和 PCI Express [137])，本文算法总能够正确的识别流控变量 $\vec{f}$，推导谓词 $valid(\vec{f})$，并产生解码器。同时，我们也和现有算法进行了运行时间，电路面积和时序方面的比较。

本章剩余部分按照以下方式组织。第4.2节介绍了背景知识；第4.4节给出了识别流控变量的算法，而第4.5节推导使得 $\vec{d}$ 能够被 $\vec{o}$ 唯一决定的谓词；第4.6节压缩迁移关系展开的长度以降低运行时间并减小电路面积和延时，而第4.7节给出了特征化解码器电路的算法；第4.8和4.9节分别给出了实验结果和结论。

4.2　背景知识

4.2.1　有限状态机

如图4.2a) 所示，本文中编码器使用有限状态机 $M = (\vec{s}, \vec{i}, \vec{o}, T)$ 作为模型。该模型包括状态向量 $\vec{s}$，输入向量 $\vec{i}$，输出向量 $\vec{o}$，和迁移函数 $T : [\![\vec{s}]\!] \times [\![\vec{i}]\!] \rightarrow [\![\vec{s}]\!] \times [\![\vec{o}]\!]$。其中 T 用于从当前状态向量 $\vec{s}$ 和输入向量 $\vec{i}$ 计算出下一状态向量 $\vec{s}$ 和输出向量 $\vec{o}$。

如图4.2b) 所示，有限状态机 M 的行为可以通过将迁移函数展开多步得到。具体的步骤如下：

1. 首先，使用在小节1.1.1.2中描述的 Tseitin 编码方法，将迁移函数 T 转化成为对应的 CNF 公式。而如图4.2b) 所示，我们可以实例化 T 的 CNF 公式的任意多个实例 T_0，...，T_n，同时保证任意两个实例所使用的变量集合不重叠。在第 j 步上，状态变量 $s \in \vec{s}$，输入变量 $i \in \vec{i}$ 和输出变量 $o \in \vec{o}$ 分别表示为 s_j，i_j 和 o_j。进一步的，在第 j 步的状态变量向量，输入变量向量和输出变量向量分别记为 $\vec{s}_j$，$\vec{i}_j$ 和 $\vec{o}_j$。

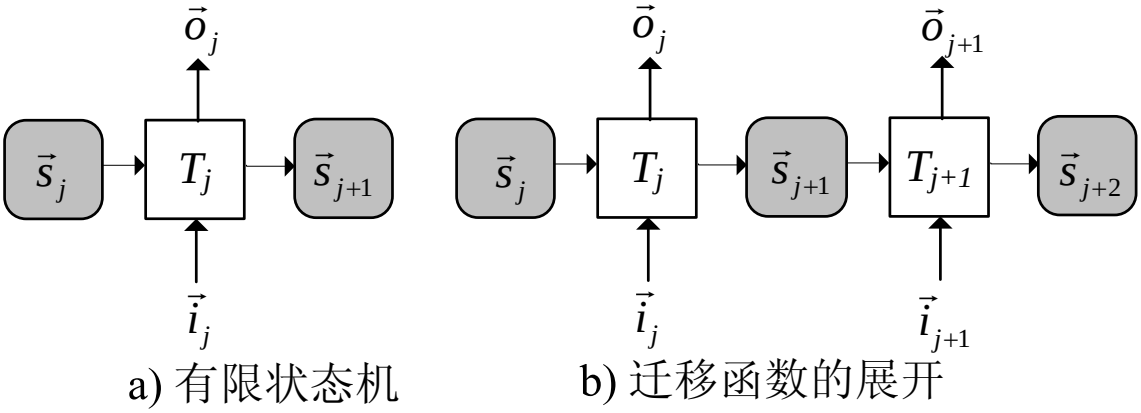


图 4.2　有限状态机及其迁移关系的展开

2. 其次，对于任意两个相邻的迁移函数 T_j 和 T_{j+1}，我们约束 T_j 的下一状态等于 T_{j+1} 的当前状态，即都等于图4.2b) 的 $\vec{s}_{j+1}$。这样任意 n 个迁移函数 T 的实例就能够被连接起来，形成一个如图4.2b) 所示长度为 $n+1$ 的序列。将这些 CNF 公式的合取形成的新的 CNF 公式，送入 SAT 求解器求解，所得到的对状态向量序列 $<\vec{s}_0,\ldots,\vec{s}_n>$ 的赋值，就代表了该有限状态机的一个长度为 n 的合法行为。

一条**路径**是一个对所有 $n \le j < m$ 均有 $\exists\vec{i}_j\vec{o}_j(\vec{s}_{j+1},\vec{o}_j) \equiv T(\vec{s}_j,\vec{i}_j)$ 的状态序列 $<\vec{s}_n,\ldots,\vec{s}_m>$。一个**环**是一条使得 $\vec{s}_n \equiv \vec{s}_m$ 的路径 $<\vec{s}_n,\ldots,\vec{s}_m>$。

4.2.2　基于迁移关系 (函数) 展开的形式化验证算法的一般性原理

我们已经在上一小节描述了如何将迁移关系 (函数) 的 CNF 公式展开成为任意长度的序列。这种展开方法被广泛应用于绝大多数形式化验证方法中，以对该状态机的行为进行推理。

假设有待验证的状态机是一个 2 位计数器，初始状态为 0，每一步的行为是将当前状态加 1。再假设当计数器为 3 时会发生一些不希望发生的行为，所以我们需要验证该计数器永远不会到达 3。因此有待证明的断言是“计数器永远不等于 3”。

最简单，同时也是最典型的限界模型检验[138] 方法，通过逐步的增加迁移关系 (函数) 展开的长度，以试图找到在展开序列上违反有待证明断言的状态。比如针对上述“计数器永远不等于 3”的断言，则违反该断言意味着计数器的值等于 3.

如图4.3所示，限界模型检验依次将迁移函数展开了 1 次，2 次和 3 次。其中在前两次都没有找到对断言的违反。而在第 3 次中则找到了计数器等于 3 的情况。在这种情况下，限界模型检验停止展开，并给出最后一个展开序列上每个状态 $\vec{s}^i$ 的值，作为反例，以证明上述“计数器永远不等于 3”的断言是错误。

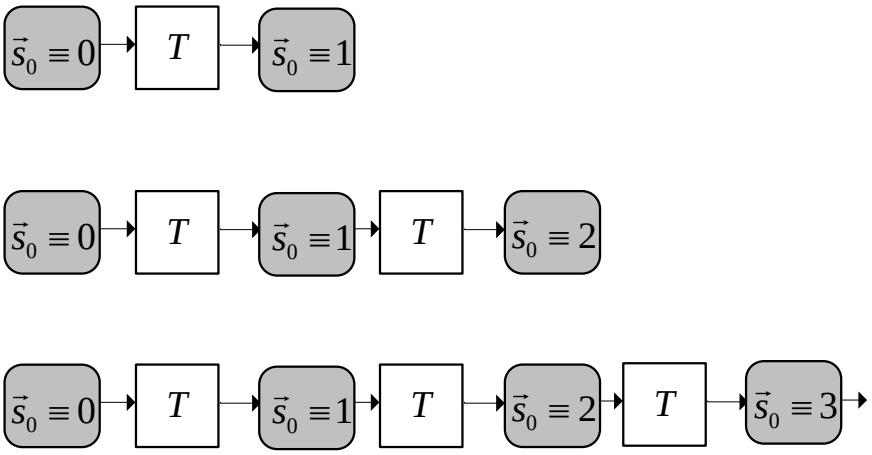


图 4.3　一个简单计数器的迁移函数展开序列

当然，限界模型检验的算法是不停机的。即当断言成立的情况下，无法退出算法。存在其他方法来部分的改善这一点，如完备性停机条件[139] 等。不过这些方法也都基于上述的基本框架，而且也都有他们自身的缺陷。

上述逐步的增加迁移关系 (函数) 展开长度的框架，也广泛应用于本文的对偶综合算法中。详情将在本文的后继部分详细描述。

4.3　对偶综合研究现状

本节将简述对偶综合领域取得的研究成果。由于其中的若干算法也将在本论文的主要章节中被使用，因此其描述方式根据本论文的需要进行了修改。因此有可能和原始论文的描述方式有微小差别。在阅读本论文时应以本小节的描述为准。

4.3.1　早期的充分非完备算法

国防科技大学的 Shen et al.[126] 首次提出了对偶综合的概念和初步算法实现。该算法是充分非完备的，这意味着当特定编码器对应的解码器存在时，该算法总能找到其实现；而当解码器不存在时，该算法不停机。

该算法类似于小节4.2.2中所描述的限界模型检验，同样是通过逐步的增加迁移函数展开序列的长度，并在每一个特定长度的展开序列上检查是否存在解码器。

针对特定展开长度，检查解码器是否存在的方法如下：如图4.4所示，对于每一个 $i \in \vec{i}$，如果存在三个参数 p，l 和 r，使得在长度为 $p+l+r$ 的迁移函数展开序列上，对于输出序列 $<\vec{o}_p,\ldots,\vec{o}_{p+l+r}>$ 的任意取值，i_{p+l} 不能同时取值为 0 和 1，

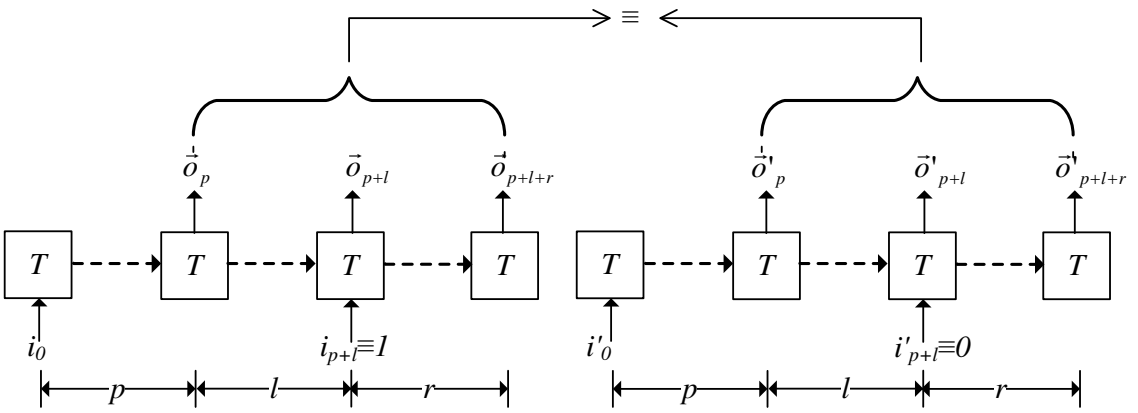


图 4.4　用于检查 i_{p+l} 是否能够被唯一决定的充分非完备算法

则输入变量 $i \in \vec{i}$ 可以被唯一决定。这等价于等式 (4.1) 中的公式 $F_{PC}(p,l,r)$ 的不可满足。

$$F_{PC}(p,l,r) := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r}\vec{o}_m \equiv \vec{o'}_m \\ \wedge & i_{p+l} \equiv 1 \wedge i'_{p+l} \equiv 0 \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.1}$$

这里，p 是前置迁移函数序列的长度，l 和 r 则是两个用于唯一决定 i_{p+l} 的输出序列 $<\vec{o}_{p+1},\ldots,\vec{o}_{p+l}>$ 和 $<\vec{o}_{p+l+1},\ldots,\vec{o}_{p+l+r}>$ 的长度。等式 (4.1) 的行 1 对应于图4.4左边的路径，而行 2 对应于图4.4右边的路径。他们的长度是相同的。行 3 强制这两条路径的输出是相同的。而行 4 要求他们的输入 i_{p+l} 是不同的。行 5 和 6 则是用户给出的断言，用于约束合法的输入模式。F_{PC} 中的 PC 是"parameterized complementary" 的缩写，意味着 $F_{PC}(p,l,r)$ 被用于检查在三个参数 p，l 和 r 的情况下，i_{p+l} 能否被唯一决定。

从图4.4可知，等式 (4.1) 的前三行代表了两个具有相同输出的迁移函数展开序列。因此他们总是可满足的。而最后两行是对合法输入模式的约束。我们将在算法开始前检查他们的可满足性。所以 $F_{PC}(p,l,r)$ 的不可满足意味着 $i_{p+l} \equiv i'_{p+l}$，即输入被唯一决定。

从图4.4可知，如果 $F_{PC}(p,l,r)$ 不可满足，则 $F_{PC}(p',l',r')$ 对于更大的 $p' \geq p$，$l' \geq l$ 和 $r' \geq r$ 也是不可满足的。从等式 (4.1) 中可知，$F_{PC}(p',l',r')$ 的短句集合是 $F_{PC}(p,l,r)$ 的超集。这也指向了同一个结论。

算法 4.1 *CheckUniquenessSound*(i): 用于检测 $i \in \vec{i}$ 是否能够被 $\vec{o}$ 的有限长度序列唯一决定的充分算法

```
1:  p:= 0;
2:  l:= 0;
3:  r:= 0;
4:  while 1 do
5:      p++;
6:      l++;
7:      r++ ;
8:      if F_PC(p,l,r) 不可满足 then
9:          return (1,p,l,r);
10:     end if
11: end while
```

这意味着，$F_{PC}(p,l,r)$ 的不可满足性的限界证明可以被扩展到任意更大的 p，l 和 r 上，从而成为非限界的证明。

命题 4.1: 如果 $F_{PC}(p,l,r)$ 不可满足，则对于任意更大的 p，l 和 r，i_{p+l} 能够被 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 唯一决定。

基于上述讨论，算法4.1即为我们所提出的充分算法。该算法首先初始化 p，l 和 r 为全 0。然后使用一个循环来持续的增加 p，l 和 r。在每一个循环中，将迁移关系展开至长度为 $p+l+r$ 的序列。若 $F_{PC}(p,l,r)$ 不可满足，则返回当前的 p，l 和 r。

上述算法仅考虑了输入向量 $\vec{i}$ 中的一个输入变量 i。当需要讨论整个 $\vec{i}$ 时，我们需要将整体的 p，l 和 r 设置为最大值。即，假设对于 $i \in \vec{i}$，其对应的唯一决定参数为 p_i，l_i 和 r_i，则对于整个 $\vec{i}$，其对应的唯一决定参数为：

$$\begin{aligned} p &:= max_{i\in\vec{i}}\{p_i\} \\ l &:= max_{i\in\vec{i}}\{l_i\} \\ r &:= max_{i\in\vec{i}}\{r_i\} \end{aligned} \tag{4.2}$$

等式 (4.1) 不包含初始状态，相反使用一个长度为 p 步的前置状态序列 $< \vec{s_0}, \ldots, \vec{s_{p-1}} >$ 以将约束 $assertion(\vec{i})$ 传播到状态序列 $< \vec{s_p}, \ldots, \vec{s_{p+l+r}} >$。从而将在 $assertion(\vec{i})$ 约束下不可达的状态集合剔除。相比考虑初始状态的传统方法，这带来了两个主要的好处：首先，通过不计算可达状态，本文算法可以得到极大的简化和加速。而相比之下，目前唯一能够计算可达状态的对偶综合算法[134] 则无法处理最为复杂的 XFI 编码器[140]。而我们的算法[129] 则始终可以处理。第二，通过

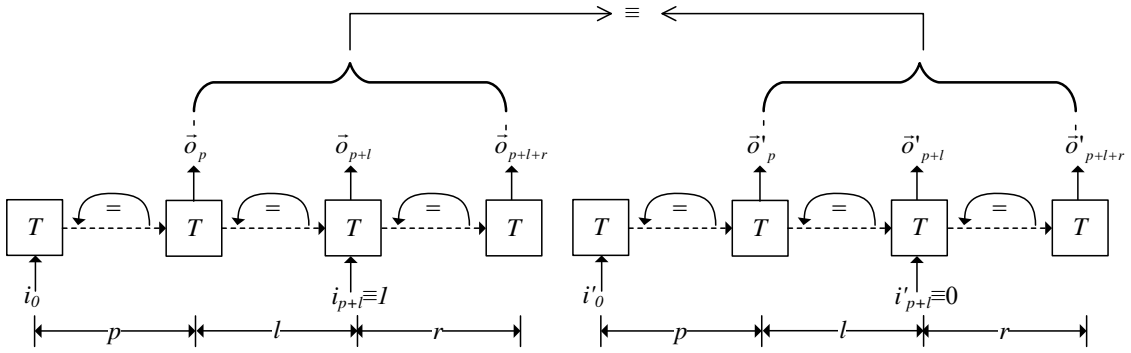


图 4.5　用于检查 i_{p+l} 是否不能被唯一决定的上估计算法

忽略初始状态，本文算法可以产生出更加可靠的解码器。因为这样可以使得解码器的状态和输出仅仅依赖于有限的输入历史。因此任何被传输中的错误破坏的 $\vec{o}$ 只能对解码器产生有限步数的影响。

当然忽略初始状态有一个缺点在于，它使得判断条件比必须的情况稍微强一些。也就是说，它要求 $\vec{i}$ 必须在一个更大的状态集合 R^p 上被唯一决定。其中 R^p 代表了由任意状态在 p 步之内能够到达的状态集合。而必要条件是从初始条件出发在任意步数内可以到达的可达状态集合 R。因此在某些情况下，我们的算法有可能无法处理正确设计的编码器。不过对于目前使用的所有编码器，即使是那些来自于真实工业应用的编码器，这种极端情况也没有出现过。

4.3.2　完备停机算法

在上一小节，我讨论了当 $F_{PC}(p,l,r)$ 不可满足时，i_{p+l} 能够在任意更大的 p，l 和 r 下被唯一决定。另一方面，如果 $F_{PC}(p,l,r)$ 是可满足的，则 i_{p+l} 不能在特定的 p，l 和 r 情况下被 $<\vec{o}_p,\ldots,\vec{o}_{p+l+r}>$ 唯一决定。此时存在两种可能性：

1. 在某个更大的 p'，l' 和 r' 情况下，$i_{p'+l'}$ 能够被 $<\vec{o}_{p'},\ldots,\vec{o}_{p'+l'+r'}>$ 唯一决定。

2. 对任意 p，l 和 r，i_{p+l} 都不能够被 $<\vec{o}_p,\ldots,\vec{o}_{p+l+r}>$ 唯一决定。

如果是第一种情况，则通过迭代的增长 p，l 和 r，$F_{PC}(p,l,r)$ 总能够变成不可满足的。然而对于第二种情况，迭代的递增 p，l 和 r 将导致不停机。

因此，为了得到一个停机算法，我们需要区分上述两种情形的手段。国防科技大学的 Shen et al.[129] 和 Liu et al.[132] 分别独立提出了类似的全新解决方案。该方案如图4.5所示，该图类似于图4.4，但是增加了三个约束用于检测三个路径

算法 4.2 *CheckUniqueness*(i): 用于检测 $i \in \vec{i}$ 是否能够被 $\vec{o}$ 的有限长度序列唯一决定的停机算法

```
1:  p:= 0;
2:  l:= 0;
3:  r:= 0;
4:  while 1 do
5:      p++;
6:      l++;
7:      r++ ;
8:      if F_PC(p,l,r) 不可满足 then
9:          return (1,p,l,r);
10:     else if F_LN(p,l,r) 可满足 then
11:         return (0,p,l,r);
12:     end if
13: end while
```

$< \vec{s}_0, \ldots, \vec{s}_p >$，$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ 和 $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$ 上的环。该方法被形式化的定义于等式 (4.3) 中。

$$F_{LN}(p,l,r) := \left\{ \begin{array}{ll} & F_{PC}(p,l,r) \\ \wedge & \bigvee_{x=0}^{p-1} \bigvee_{y=x+1}^{p} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+1}^{p+l-1} \bigvee_{y=x+1}^{p+l} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+l+1}^{p+l+r-1} \bigvee_{y=x+1}^{p+l+r} \{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \end{array} \right\} \tag{4.3}$$

F_{LN} 中的 LN 意味着环形非对偶。这表明 $F_{LN}(p,l,r)$ 将使用这三个环检测约束来检测 i_{p+l} 是否不能够被唯一决定。

当 $F_{LN}(p,l,r)$ 可满足时，i_{p+l} 不能被 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 唯一决定。更重要的是，这个可满足结果意味着在三个路径 $< \vec{s}_0, \ldots, \vec{s}_p >$，$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ 和 $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$ 上分别各存在一个环。通过展开这三个环，我们可以把上述三个路径变得更长，从而得到针对更大的 p，l 和 r 的新公式 $F_{LN}(p,l,r)$。而该公式仍然是可满足的。这意味着:

命题 4.2: *当 $F_{LN}(p,l,r)$ 可满足时，i_{p+l} 针对任意 p，l 和 r 都不能被 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 唯一决定。*

根据命题4.1 和4.2，我们能将针对特定 p，l 和 r 的限界证明扩展到针对任意 p，l 和 r 的非限界情形。这使得我们得到停机算法4.2，用于检测 $i \in \vec{i}$ 是否能被 $\vec{o}$ 的有限长度序列唯一决定。

1. 一方面，如果确实存在 p，l 和 r，使得输入能被输出唯一决定，令 $p' := max(p,l,r)$，$l' := max(p,l,r)$ 和 $r' := max(p,l,r)$。从命题4.1，可知 $F_{PC}(p',l',r')$ 是不可满足的。因此 $F_{PC}(p,l,r)$ 总能够在算法4.2行8成为不可满足的并退出循环;

2. 另一方面，如果不存在这样的 p，l 和 r，则 p，l 和 r 在不断的递增之后最终总能够大于有限状态机的最大无环路径长度。这意味着在 $< \vec{s}_0, \ldots, \vec{s}_p >$，$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ 和 $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$ 上都存在环。这将使得 $F_{LN}(p,l,r)$ 在行10被满足。这也将导致退出循环。

因此该算法是停机的。

类似于等式4.2，我们需要将整体的 p，l 和 r 设置为最大值。

$$\begin{aligned} p &:= max_{i \in \vec{i}}\{p_i\} \\ l &:= max_{i \in \vec{i}}\{l_i\} \\ r &:= max_{i \in \vec{i}}\{r_i\} \end{aligned} \tag{4.4}$$

4.4 识别流控变量

本节将介绍如何识别流控变量 $\vec{f}$。这将首先在小节4.4.1中介绍。然后小节4.4.2将介绍如何使用增量求解算法加速该算法。

4.4.1 识别流控变量 $\vec{f}$

为了便于描述，我们假设 $\vec{i}$ 可以被划分为两个向量: 流控向量 $\vec{f}$ 和数据向量 $\vec{d}$。

流控向量 $\vec{f}$ 用于表达 $\vec{d}$ 的有效性。所以对于一个正确设计的编码器，$\vec{f}$ 总能够被输出变量向量 $\vec{o}$ 的一个有限长度序列唯一决定。否则解码器无法识别 $\vec{d}$ 的有效性。

所以我们提出算法4.3用于识别 $\vec{f}$。其中 $F_{PC}(p,l,r)$ 和 $F_{LN}(p,l,r)$ 分别定义于公式 (4.1) 和 (4.3)。

在行2，f 和 d 被设为空向量。在行5，p，l 和 r 的初始值被设为 0。

在行6，一个 while 循环被用于遍历 $i \in \vec{i}$。

在行11，能够被唯一决定的输入变量 i 将被加入 $\vec{f}$。

另一方面，当 $\vec{i}$ 非常长时，逐一测试 $i \in \vec{i}$ 的时间开销将会很大。为了加速该算法，当 $F_{LN}(p,l,r)$ 在行14是可满足的时候，每个在 j_{p+l} 和 j'_{p+l} 上取不同值的 $j \in \vec{i}$ 也将被在行15加入 $\vec{d}$，因为他们自己的 $F_{LN}(p,l,r)$ 也是可满足的。

在某些特殊情况下，一些 $d \in \vec{d}$ 也能够像 $f \in \vec{f}$ 那样被唯一决定。在这种情况下，d 也将被算法4.3识别为流控变量。但是这不会对我们的算法产生不利影响，因为这些 d 的解码器函数也将在节4.7中被正确特征化。

4.4.2 使用增量 SAT 求解器加速识别算法

通过小节1.1.2.4中的 MiniSat 增量求解机制，我们能进一步加速算法4.3 。

算法 4.3 *FindFlow*($\vec{i}$): 识别 $\vec{f}$

```
1:  f⃗ := {};
2:  d⃗ := {};
3:  p := 0 ;
4:  l := 0 ;
5:  r := 0 ;
6:  while i⃗ ≠ {} do
7:      假设 i ∈ i⃗;
8:      p + +;
9:      l + +;
10:     r + +;
11:     if F_PC(p, l, r) 对于 i 不可满足 then
12:         f⃗ := i ∪ f⃗;
13:         i⃗ := i⃗ − i;
14:     else if F_LN(p, l, r) 对于 i 可满足且赋值为 A then
15:         for j ∈ i⃗ do
16:             if A(j_{p+l}) ≠ A(j'_{p+l}) then
17:                 i⃗ := i⃗ − j;
18:                 d⃗ := j ∪ d⃗;
19:             end if
20:         end for
21:     end if
22: end while
23: return (f⃗,p,l,r);
```

通过将等式 (4.1) 的第三行移动到等式 (4.6)，等式 (4.1) 中的 $F_{PC}(p,l,r)$ 可以被划分为以下两个等式：

$$C_{PC}(p,l,r) := \left\{ \begin{array}{cc} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.5}$$

$$A_{PC}(p,l,r) := \left\{ \; i_{p+l} \equiv 1 \wedge i'_{p+l} \equiv 0 \; \right\} \tag{4.6}$$

类似的我们可以将等式 (4.3) 中的 $F_{LN}(p,l,r)$ 划分为下列两个等式：

$$C_{LN}(p,l,r) := \left\{ \begin{array}{cc} & C_{PC}(p,l,r) \\ \wedge & \bigvee_{x=0}^{p-1}\bigvee_{y=x+1}^{p}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+1}^{p+l-1}\bigvee_{y=x+1}^{p+l}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+l+1}^{p+l+r-1}\bigvee_{y=x+1}^{p+l+r}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \end{array} \right\} \tag{4.7}$$

$$A_{LN}(p,l,r) := \left\{ \; i_{p+l} \equiv 1 \wedge i'_{p+l} \equiv 0 \; \right\} \tag{4.8}$$

很明显 C_{PC} 和 C_{LN} 是独立于特定 $i \in \vec{i}$ 的，所以可以使用 $addClause(C_{PC})$ 或 $addClause(C_{LN})$ 来将它们一次性添加进入 MiniSat 的短句数据库。于此同时，A_{PC} 和 A_{LN} 中的短句只包含单个文字，因此它们可以作为调用 *solve* 函数时的假设集合。

因此，基于上述等式，我们可以使用增量求解机制将算法4.3 修改成为算法4.4 。主要的修改在于行11 和18上的两个 *addClause* ，以及行13 和20上的两个 *solve*。这些新加的函数来自于在小节1.1.2.4中描述的 MiniSat 的增量求解机制接口。

4.5 推导使得数据变量向量被唯一决定的谓词

在本小节中，我们使用小节3.4的迭代特征化算法算法推导 $valid(\vec{f})$，也就是使得 $\vec{d}$ 能够被 $\vec{o}$ 的有限长度序列唯一决定的谓词。

如图4.6所示，我们将首先在小节4.5.1中定义 $valid(\vec{f})$ 的一个单调增长的下估计。然后我们将在小节4.5.2中定义 $valid(\vec{f})$ 的一个单调递减的上估计。然后在

小节4.5.3中我们指出这两个估计将收敛至 $valid(\vec{f})$ 。小节4.5.4将证明该算法的正确性。

算法 4.4 *FindFlowIncSAT*($\vec{i}$): 基于增量求解识别流控变量

```
1:  f⃗ := {};
2:  d⃗ := {} ;
3:  p := 0 ;
4:  l := 0 ;
5:  r := 0 ;
6:
7:  while i⃗ ≠ {} do
8:      p + +;
9:      l + +;
10:     r + +;
11:     addClause(C_PC(p, l, r));
12:     for i ∈ i⃗ do
13:         if solve(A_PC(p, l, r)) 不可满足 i then
14:             i⃗ := i⃗ − i;
15:             f⃗ := i ∪ f⃗;
16:         end if
17:     end for
18:     addClause(C_LN(p, l, r));
19:     for i ∈ i⃗ do
20:         if solve(A_LN(p, l, r)) 可满足 i 且赋值为 A then
21:             for j ∈ i⃗ do
22:                 if A(j_{p+l}) ≠ A(j'_{p+l}) then
23:                     i⃗ := i⃗ − j;
24:                     d⃗ := j ∪ d⃗;
25:                 end if
26:             end for
27:         end if
28:     end for
29: end while
30: return (f⃗,p,l,r)
```

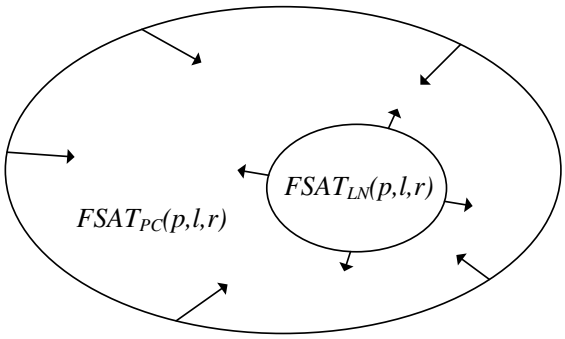


图 4.6　$FSAT_{PC}(p,l,r)$ 和 $FSAT_{LN}(p,l,r)$ 的单调性

4.5.1　计算 $valid(\vec{f})$ 的单调增长的下估计

通过将等式 (4.1) 中的 i 替换为 $\vec{d}$，我们得到：

$$F^{d}_{PC}(p,l,r) := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \\ \wedge & \vec{d}_{p+l} \neq \vec{d'}_{p+l} \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.9}$$

如果 $F^{d}_{PC}(p,l,r)$ 是可满足的，则 $\vec{d}_{p+l}$ 无法被 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 唯一决定。通过收集等式 (4.9) 的第三行，我们得到 $T_{PC}(p,l,r)$：

$$T_{PC}(p,l,r) := \left\{ \quad \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \right\} \tag{4.10}$$

通过将 $T_{PC}(p,l,r)$ 代入到 $F^{d}_{PC}(p,l,r)$，我们得到一个新的公式：

$$F'^{d}_{PC}(p,l,r,t) := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & t \equiv T_{PC}(p,l,r) \\ \wedge & \vec{d}_{p+l} \neq \vec{d'}_{p+l} \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.11}$$

很明显 $F^{d}_{PC}(p,l,r)$ 和 $F'^{d}_{PC}(p,l,r,1)$ 是等价的。我们进一步定义:

$$\vec{a} := \vec{f}_{p+l} \tag{4.12}$$

$$\vec{b} := \vec{d}_{p+l} \cup \vec{d'}_{p+l} \cup \vec{s}_0 \cup \vec{s'}_0 \cup \bigcup_{0\le x\le p+l+r, x\neq(p+l)} (\vec{i}_x \cup \vec{i'}_x) \tag{4.13}$$

则 $\vec{a}\cup\vec{b}$ 包含了两个迁移函数展开序列上的所有输入状态向量 $<\vec{i}_0,\ldots,\vec{i}_{p+l+r}>$ and $<\vec{i'}_0,\ldots,\vec{i'}_{p+l+r}>$。它同时也包含了两个展开序列的初始状态 $\vec{s}_0$ 和 $\vec{s'}_0$。进一步的，等式 (4.11) 前两行的迁移关系序列能够从输入序列和初始状态唯一的计算出输出序列。因此 $\vec{a}$ 和 $\vec{b}$ 能够唯一决定 $F'^{d}_{PC}(p,l,r,t)$ 中 t 的取值。因此，对于特定 p,l 和 r，以 $\vec{f}_{p+l}$ 为输入并使得 $F'^{d}_{PC}(p,l,r,1)$ 可满足的函数可以通过以 $F'^{d}_{PC}(p,l,r,t)$, $\vec{a}$ 和 $\vec{b}$ 为参数调用算法3.1得到:

$$FSAT_{PC}(p,l,r) := CharacterizingFormulaSAT(F'^{d}_{PC}(p,l,r,t),\vec{a},\vec{b},t) \tag{4.14}$$

因此 $FSAT_{PC}(p,l,r)$ 覆盖了使得 $F^{d}_{PC}(p,l,r)$ 可满足的 $\vec{f}_{p+l}$ 赋值集合。因此，其反 $\neg FSAT_{PC}(p,l,r)$ 是使得 $F^{d}_{PC}(p,l,r)$ 不可满足的 $\vec{f}_{p+l}$ 集合。

从命题4.1可知，$F^{d}_{PC}(p,l,r)$ 的不可满足证明可以推广到任意更大的 p，l 和 r 上。任意被 $\neg FSAT_{PC}(p,l,r)$ 覆盖的 $\vec{f}$ 也仍然能够使 $F^{d}_{PC}(p,l,r)$ 对于任意更大的 p，l 和 r 不可满足。

所以我们有:

命题 4.3: $\neg FSAT_{PC}(p,l,r)$ 是 $valid(\vec{f})$ 针对 p，l 和 r 单调递增的一个下估计。

这直观的展示在了图4.6中。

4.5.2 计算 $valid(\vec{f})$ 的单调递减上估计

.

类似的，通过将公式 (4.3) 中 $F_{LN}(p,l,r)$ 的 i 替换为 $\vec{d}$，我们有：

$$F^{d}_{LN}(p,l,r) := \left\{ \begin{array}{cc} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r}\vec{o}_m \equiv \vec{o'}_m \\ \wedge & \vec{d}_{p+l} \neq \vec{d'}_{p+l} \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \\ \wedge & \bigvee_{x=0}^{p-1}\bigvee_{y=x+1}^{p}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+1}^{p+l-1}\bigvee_{y=x+1}^{p+l}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+l+1}^{p+l+r-1}\bigvee_{y=x+1}^{p+l+r}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \end{array} \right\} \tag{4.15}$$

如果 $F^{d}_{LN}(p,l,r)$ 可满足，则 $\vec{d}_{p+l}$ 不能被 $<\vec{o}_p,\ldots,\vec{o}_{p+l+r}>$ 唯一决定。进一步的，类似于命题4.2，通过展开等式 (4.15) 中最后三行的环，我们能够证明 $\vec{d}_{p+l}$ 对于任意更大的 p，l 和 r 都不能被唯一决定。通过收集等式 (4.15) 的第三行和最后三行，我们进一步定义了 $T_{LN}(p,l,r)$：

$$T_{LN}(p,l,r) := \left\{ \begin{array}{cc} & \bigwedge_{m=p}^{p+l+r}\vec{o}_m \equiv \vec{o'}_m \\ \wedge & \bigvee_{x=0}^{p-1}\bigvee_{y=x+1}^{p}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+1}^{p+l-1}\bigvee_{y=x+1}^{p+l}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \\ \wedge & \bigvee_{x=p+l+1}^{p+l+r-1}\bigvee_{y=x+1}^{p+l+r}\{\vec{s}_x \equiv \vec{s}_y \wedge \vec{s'}_x \equiv \vec{s'}_y\} \end{array} \right\} \tag{4.16}$$

通过将等式 (4.15) 的第三行和最后三行替换为 $T_{LN}(p,l,r)$，我们得到：

$$F'^{d}_{LN}(p,l,r,t) := \left\{ \begin{array}{cc} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1},\vec{o}_m) \equiv T(\vec{s}_m,\vec{i}_m)\} \\ \wedge & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1},\vec{o'}_m) \equiv T(\vec{s'}_m,\vec{i'}_m)\} \\ \wedge & t \equiv T_{LN}(p,l,r) \\ \wedge & \vec{d}_{p+l} \neq \vec{d'}_{p+l} \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.17}$$

很显然 $F^{d}_{LN}(p,l,r)$ 和 $F'^{d}_{LN}(p,l,r,1)$ 等价。因此，对于特定的 p，l 和 r，定义在 $\vec{f}_{p+l}$ 上且能够使得 $F^{d}_{LN}(p,l,r)$ 可满足的函数可以通过下式计算：

$$FSAT_{LN}(p,l,r) := CharacterizingFormulaSAT(F'^{d}_{LN}(p,l,r,t),\vec{d},\vec{b},t) \tag{4.18}$$

算法 4.5 *InferringUniqueFormula*: 推导使得 $\vec{d}_{p+l}$ 能够被唯一决定的 $valid(\vec{f}_{p+l})$

1: $p := p_{max}$; $l := l_{max}$; $r := r_{max}$;
2: **while** $\neg FSAT_{LN}(p,l,r) \wedge FSAT_{PC}(p,l,r)$ is satisfiable **do**
3: p ++ ;
4: l ++ ;
5: r ++ ;
6: **end while**
7: **return** $\neg FSAT_{LN}(p,l,r)$

再次参见命题4.2，$F^d_{LN}(p,l,r)$ 的可满足证明可以扩展到任意更大的 p，l 和 r 上。因此任意被 $FSAT_{LN}(p,l,r)$ 覆盖的 $\vec{f}$ 仍然能够对所有更大的 p，l 和 r 使得 $F^d_{LN}(p,l,r)$ 可满足。因此 $FSAT_{LN}(p,l,r)$ 单调增长且是 $\neg valid(\vec{f})$ 的子集。

因此我们下列命题:

命题 4.4: $\neg FSAT_{LN}(p,l,r)$ 是 $valid(\vec{f}))$ 的单调递减上估计。

这被直观的展示在了图4.6中。

4.5.3 计算 $valid(\vec{f})$ 的算法

.

基于命题4.3 和4.4，我们给出了算法4.5以推导 $valid(\vec{f}_{p+l})$。该算法迭代的增加 p，l 和 r，直到 $\neg FSAT_{LN}(p,l,r) \wedge FSAT_{PC}(p,l,r)$ 不可满足。这意味着 $FSAT_{PC}(p,l,r)$ 和 $FSAT_{LN}(p,l,r)$ 收敛到一个确定的集合上。在此情况下，$\neg FSAT_{PC}(p,l,r)$ 即为 $valid(\vec{f})$。

该算法的正确性证明在下一小节给出。

4.5.4 停机和正确性证明

首先我们需要证明以下三个引理:

引理 4.1: 算法4.5 中的 $\mathbf{FSAT_{PC}(p,l,r)}$ 针对 p，l 和 r 单调递减。

证明: 对于任意 $p' > p$，$l' > l$ 和 $r' > l$，假设 $A : \vec{f}_{p'+l'} \rightarrow B$ 是流控变量在第 $(p'+l')$ 步的取值。进一步假设 A 被 $FSAT_{PC}(p',l',r')$ 覆盖。

根据等式 (4.14) 算法3.1，易知 A 能够使得 $F'^d_{PC}(p',l',r',1)$ 可满足。假设 $F'^d_{PC}(p',l',r',1)$ 的满足赋值为 A'。易知 $A(\vec{f}_{p'+l'}) \equiv A'(\vec{f}_{p'+l'})$。

直观的，如图4.7所示，通过将第 $(p'+l')$ 和 $(p+l)$ 步对准，我们能够将 $F'^d_{PC}(p',l',r',1)$ 的状态变量，输入变量和输出变量赋值映射到 $F'^d_{PC}(p,l,r,1)$。并淘

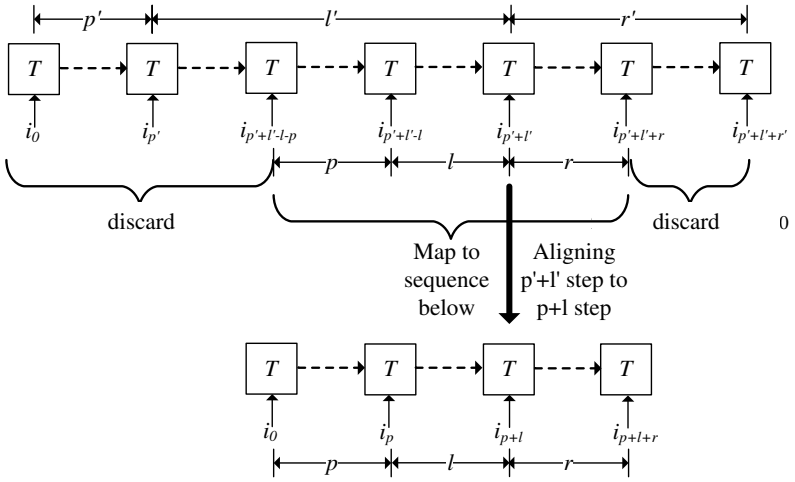


图 4.7　通过将第 $(p'+l')$ 步对准第 $(p+l)$ 步映射 $F'^{d}_{PC}(p',l',r',1)$ 到 $F'^{d}_{PC}(p,l,r,1)$。

汰前置和后置的状态序列。形式化的，对于 $p'+l'-l-p \le n \le p'+l'+r$，我们映射 $F'^{d}_{PC}(p',l',r',1)$ 中的 s_n 到 $F'^{d}_{PC}(p,l,r,1)$ 中的 $s_{n-p'-l'+l+p}$。i_n 和 o_n 的映射类似。

基于该映射，我们能将 $F'^{d}_{PC}(p',l',r',1)$ 的 A' 映射为另一个 $F'^{d}_{PC}(p,l,r,1)$ 的可满足赋值 A''。

通过将 A'' 的定义域限制为 $\vec{f}_{p+l}$，我们得到第四个可满足赋值 $A''' : \vec{f}_{p+l} \to B$。从上述构造过程可知，$A''' \equiv A$。

因此，任意被 $FSAT_{PC}(p',l',r')$ 覆盖的 A，都能够被 $FSAT_{PC}(p,l,r)$ 覆盖。

因此，$FSAT_{PC}(p,l,r)$ 针对 p，l 和 r 单调递减。 ■

引理 4.2: 算法4.5 中的 $\mathbf{FSAT_{LN}(p,l,r)}$ 针对 p, l 和 r 单调递增。

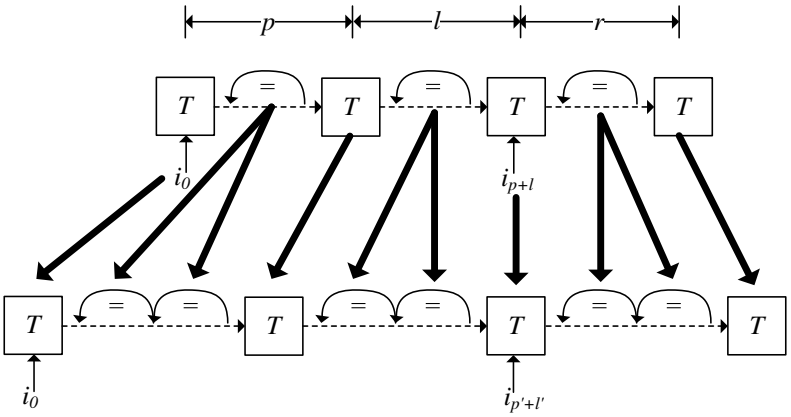


图 4.8　通过将第 $(p+l)$ 步对准 $(p'+l')$ 步并展开三个环，从而映射 $F'^{d}_{LN}(p,l,r,1)$ 到 $F'^{d}_{LN}(p',l',r',1)$。

证明：对于任意 $p' > p$，$l' > l$ 和 $r' > l$，假设 $A : \vec{f}_{p+l} \rightarrow B$ 是流控变量在第 $(p+l)$ 步的赋值。进一步假设 A 被 $FSAT_{LN}(p,l,r)$ 覆盖。

因此 A 能够使 $F'^{d}_{LN}(p,l,r,1)$ 可满足。假设 $F'^{d}_{LN}(p,l,r,1)$ 的可满足赋值为 A'。可知 $A(\vec{f}_{p+l}) \equiv A'(\vec{f}_{p+l})$。

从图4.8可知，通过对齐第 $(p+l)$ 步到第 $(p'+l')$ 步，并展开三个环，我们能将 $F'^{d}_{LN}(p,l,r,1)$ 映射 $F'^{d}_{LN}(p',l',r',1)$。如此可得到 $F'^{d}_{LN}(p',l',r',1)$ 的可满足赋值 A''。

通过将 A'' 的定义域限制为 $\vec{f}_{p'+l'}$，我们可以得到第四个可满足赋值 A'''：$\vec{f}_{p'+l'} \rightarrow B$。很明显 $A \equiv A'''$。

这意味着每个被 $FSAT_{LN}(p,l,r)$ 覆盖的赋值也同时被 $FSAT_{LN}(p',l',r')$ 覆盖。因此 $FSAT_{LN}(p,l,r)$ 针对 p，l 和 r 单调递增。 ■

引理 4.3：$FSAT_{LN}(p,l,r) \Rightarrow FSAT_{PC}(p,l,r)$

证明：很明显 $F'^{d}_{LN}(p,l,r,1)$ 的短句集合是 $F'^{d}_{PC}(p,l,r,1)$ 的超集。所以 $F'^{d}_{LN}(p,l,r,1)$ 的每个可满足赋值也能够使得 $F'^{d}_{PC}(p,l,r,1)$ 可满足。因此 $FSAT_{LN}(p,l,r) \Rightarrow FSAT_{PC}(p,l,r)$ 成立。 ■

这三个引理直观的展示在图4.6中。很明显 $\neg FSAT_{LN}(p,l,r) \wedge FSAT_{PC}(p,l,r)$ 在算法4.5中是单调递减的。基于这些引理，我们首先证明4.5 是停机的：

定理 4.1：算法4.5 是停机算法。

证明：编码器是一个有限状态机，其最长的无环路径的长度是有限的。如果算法4.5 不停机，则 p，l 和 r 总会大于该长度。这意味着在下面的三个状态序列 $< \vec{s}_0, \ldots, \vec{s}_p >$，$< \vec{s}_{p+1}, \ldots, \vec{s}_{p+l} >$ 和 $< \vec{s}_{p+l+1}, \ldots, \vec{s}_{p+l+r} >$ 上必然存在环。因此，每个 $F'^{d}_{PC}(p,l,r,1)$ 的可满足赋值必然也能够满足 $F'^{d}_{LN}(p,l,r,1)$。这意味着 $\neg FSAT_{LN}(p,l,r) \wedge FSAT_{PC}(p,l,r)$ 是不可满足的。这将导致算法4.5的停机。所以得证。 ■

其次，我们将证明算法4.5的正确性。

定理 4.2：从算法4.5返回的 $\neg FSAT_{LN}(p,l,r)$ 覆盖且仅覆盖了所有能够使 $\vec{d}$ 被 $\vec{o}$ 的有限长度序列唯一决定的 $\vec{f}$。

证明：首先证明覆盖的情况。$FSAT_{LN}(p,l,r)$ 覆盖了一个 $\vec{f}$ 集合使得 $\vec{d}$ 对特定的 p，l 和 r 不能被唯一决定。因此 $\neg FSAT_{LN}(p,l,r)$ 排除了该集合，从而包含了所有能够使得 $\vec{d}$ 被唯一决定的 $\vec{f}$。

然后我们证明仅覆盖的情形。如果 A 是被 $\neg FSAT_{LN}(p,l,r)$ 覆盖的 $\vec{f}$ 的赋值，且能够使得 $\vec{d}$ 针对特定 p'，l' 和 r' 不被唯一决定，则有：

1. 一方面 $FSAT_{LN}(p',l',r')$ 也覆盖 A。则从引理4.2可知，对所有 $p'' > max(p',p)$, $l'' > max(l',l)$ 和 $r'' > max(r',r)$，$FSAT_{LN}(p'',l'',r'')$ 也覆盖 A。

2. 同时 $FSAT_{LN}(p,l,r)$ 不覆盖 A。$FSAT_{LN}(p,l,r)$ 是算法4.5 结束前计算出来的最后一个。这意味着 $valid(\vec{f})$ 的上估计和下估计收敛了。因此对于所有 $p'' > max(p',p)$，$l'' > max(l',l)$ 和 $r'' > max(r',r)$，$FSAT_{LN}(p'',l'',r'')$ 必然等于 $FSAT_{LN}(p,l,r)$。因此 $FSAT_{LN}(p'',l'',r'')$ 也不覆盖 A。

这导致了冲突，因此仅覆盖的情形得证。 ■

4.6 压缩迁移关系展开序列的长度

我们首先在小节4.6.1中介绍为什么和如何削减 l 和 r 的长度。然后在小节4.6.2中给出我们算法的另一种可能结构。并讨论为什么我们选择了小节4.6.1 中的算法而不是小节4.6.2中的算法。

4.6.1 压缩 l 和 r

由于我们在算法4.5中同步增长 p，l 和 r 的值。这导致它们的值有可能包含冗余。这将导致产生的解码器的面积和时序上有不必要的额外开销。

例如，假设一个编码器仅包含一个简单的 buffer，其功能为 $o := i$。当 $p \equiv 0$，$l \equiv 0$ 和 $r \equiv 0$ 时，我们能够得到最简单的解码器 $i := o$。该解码器只包含一个 buffer，而不包含状态变量。而对于 $p \equiv 0$，$l \equiv 0$ 和 $r \equiv 1$，我们则需要一个额外的状态变量以将 o 延迟一步，然后从延迟的 o 中回复 i_i。

根据图4.4，很明显 r 影响解码器的电路面积和延迟，l 仅影响解码器的电路面积，而 p 并不对解码器的上述特性带来影响。

因此如算法4.6所示，我们选择首先压缩 r，然后压缩 l。

为了简化描述，我们将仅介绍 r 的情况。在行2，当 $F_{PC}(p,l,r'-1) \wedge valid(\vec{f}_{p+l})$ 可满足，则 r' 是最后一个使其不可满足的，我们将其直接返回 r'。另一方面，如果 $r' \equiv 0$ 仍能够使行4的 $F_{PC}(p,l,r') \wedge valid(\vec{f}_{p+l})$ 不可满足，我们直接返回 0。

4.6.2 另一种可能的算法结构

在上述讨论中，我们在算法4.4中同步增加 p，l 和 r 以找到流控变量，然后在算法4.6中压缩他们的值。该算法需要调用 SAT 求解器的次数是 $O(n)$ 。其中 $n = max(p,l,r)$。

还有另一种可能的方法：即使用三个嵌套的环逐一增加 p，l 和 r。该算法需要调用 SAT 求解器的次数是 $O(n^3)$ 。

算法 4.6 *RemoveRedundancy*(p,l,r)

for $r' := r \rightarrow 1$ **do**
  if $F_{PC}(p,l,r'-1) \wedge valid(\vec{f}_{p+l})$ 是可满足的 **then**
    break;
  else if $r' \equiv 1$ **then**
    $r' := r' - 1$;
    break ;
  end if
end for
for $l' := l \rightarrow 1$ **do**
  if $F_{PC}(p,l'-1,r') \wedge valid(\vec{f}_{p+l'-1})$ 是可满足的 **then**
    break;
  else if $l' \equiv 1$ **then**
    $l' := l' - 1$;
    break;
  end if
end for
return $< l', r' >$

我们将在小节4.8.6 中指出，同步增加 p，l 和 r 然后使用算法4.6 进行压缩比单独增加 p，l 和 r 要更有优势。我们将在那里对该现象进行解释。

4.7 产生解码器函数

在小节4.4中，解码器的输入 $\vec{i}$ 被划分为两个向量：流控向量 $\vec{f}$ 和数据向量 $\vec{d}$。为这两个向量分别产生解码器函数的算法是不同的，在下面两个小节中将分别描述。

4.7.1 产生 $\vec{f}$ 的解码器函数

每个 $f \in \vec{f}$ 都能够被输出向量 $\vec{o}$ 的有限长度序列唯一决定。所以，对于每个特定的输出向量序列 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$，$f_{p+l}$ 不能同时取值为 1 和 0。因此，计算 f_{p+l} 的解码器函数可以视为 ϕ_A 相对于 ϕ_B 的 Craig 插值，其中 ϕ_A 和 ϕ_B 分别定义如下：

$$\phi_A := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r} \{(\vec{s}_{m+1}, \vec{o}_m) \equiv T(\vec{s}_m, \vec{i}_m)\} \\ \wedge & f_{p+l} \equiv 1 \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \end{array} \right\} \tag{4.19}$$

$$\phi_B := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1}, \vec{o'}_m) \equiv T(\vec{s'}_m, \vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \\ \wedge & f'_{p+l} \equiv 0 \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.20}$$

显然 $\phi_A \wedge \phi_B$ 等价于等式 (4.1) 中的 $F_{PC}(p,l,r)$，因此它是不可满足的。ϕ_A 和 ϕ_B 的共同变量集合为 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$。因此，Craig 插值 ITP 可以使用 McMillian 算法[38] 从 $\phi_A \wedge \phi_B$ 的不可满足证明序列中产生出来。ITP 覆盖了所有使得 $f_{p+l} \equiv 1$ 的 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 的赋值。同时，$ITP \wedge \phi_B$ 是不可满足的，因此 ITP 没有覆盖任何使 $f_{p+l} \equiv 0$ 成立的 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 的赋值。因此，ITP 是计算 $f \in \vec{f}$ 的解码函数。

为了进一步提高产生 $f \in \vec{f}$ 的解码函数的速度，我们通过以下方式使用 MiniSat 的递增求解机制：

1. 从 ϕ_A 中移除 $f_{p+l} \equiv 1$，从 ϕ_B 中移除 $f'_{p+l} \equiv 0$。
2. 将 $\phi_A \wedge \phi_B$ 加入 MiniSat 的短句数据库。
3. 针对每一个 $f \in \vec{f}$，使用 $f_{p+l} \equiv 1$ 和 $f'_{p+l} \equiv 0$ 作为求解的假设。并从不可满足证明中产生 Craig 插值。

4.7.2 产生 $\vec{d}$ 的解码函数

假设 $valid(\vec{f})$ 是被算法4.5推导出来的谓词。为每个 $d \in \vec{d}$，我们定义以下两个公式：

$$\phi'_A := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s}_{m+1}, \vec{o}_m) \equiv T(\vec{s}_m, \vec{i}_m)\} \\ \wedge & d_{p+l} \equiv 1 \\ \wedge & valid(\vec{f}_{p+l}) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i}_m) \end{array} \right\} \tag{4.21}$$

$$\phi'_B := \left\{ \begin{array}{ll} & \bigwedge_{m=0}^{p+l+r}\{(\vec{s'}_{m+1}, \vec{o'}_m) \equiv T(\vec{s'}_m, \vec{i'}_m)\} \\ \wedge & \bigwedge_{m=p}^{p+l+r} \vec{o}_m \equiv \vec{o'}_m \\ \wedge & d'_{p+l} \equiv 0 \\ \wedge & valid(\vec{f'}_{p+l}) \\ \wedge & \bigwedge_{m=0}^{p+l+r} assertion(\vec{i'}_m) \end{array} \right\} \tag{4.22}$$

当 $valid(\vec{f})$ 成立时，每个 $d \in \vec{d}$ 均能够被唯一决定。因此，如果 $valid(\vec{f}_{p+l})$ 成立，对于每个特定的 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$，$d_{p+l}$ 不能同时为 1 和 0，因此，$\phi'_A \wedge \phi'_B$ 是不可满足的。因此，可以使用 McMillian 算法[38] 从 $\phi'_A \wedge \phi'_B$ 的不可满足证明中产生 Craig 插值 ITP。ITP 覆盖且仅覆盖使得 $d_{p+l} \equiv 1$ 的 $< \vec{o}_p, \ldots, \vec{o}_{p+l+r} >$ 的赋值。因此，ITP 是 $d \in \vec{d}$ 的解码器函数。

进一步的当 $valid(\vec{f}_{p+l})$ 不成立时，数据变量 $d \in \vec{d}_{p+l}$ 不能被唯一决定。因此，不存在计算它的解码器函数。不过这并不影响我们的算法的正确性，因为在这种情况下解码器只需恢复 $\vec{f}$，并忽略 $\vec{d}$。

类似的，我们也使用小节4.7.1 中的增量求解机制加速该算法。

4.8 实验结果

我们使用 OCaml 语言[141] 实现了所有算法，并使用 MiniSat 1.14[25] 求解所有的 CNF 公式。所有的实验使用一台包含 16 个 Intel Xeon E5648 2.67GHz 处理器，192GB 存储器，和 CentOS 5.4 Linux 操作系统的服务器进行。

4.8.1 测试集

表4.1 给出了测试集的信息。他们在自于两个来源:

1. 我们以前的论文[131].
2. Liu et al. [133].

表4.1 的每一列依次给出了每个实验对象的输入位数、输出位数、状态位数、映射到 mcnc.genlib 标准单元库后的门数和面积。映射使用 ABC 综合工具[142]，脚本为”strash; dsd; strash; dc2; dc2; dch; map”。该脚本来自于[133]。本文剩余部分给出的所有电路面积和延时都使用同样的设置产生。这使得我们的结果可以被用于和[133] 作比较。

表4.1 的最后一列也给出了我们将如何描述每一个测试电路的实验结果。

1. 对于来自于我们以前论文[131] 的 5 个测试电路，我们发现他们中的大多数都有流控机制。这并不奇怪，因为这些测试电路都来自于实际的工业项目。他们的实验结果将分别在小节 4.8.2，4.8.3 和4.8.4中描述。
2. 对于其他不包含流控机制的测试电路，如果他们的输入都能够被输出唯一决定，则我们的算法能够将他们所有的输入都识别成流控变量，并直接生成他们的解码器函数。他们的实验结果将在小节4.8.5中描述。

我们还进行了下列额外的实验：

在小节4.8.6中，我们将比较下列两种不同算法的时间开销：

1. 在算法4.5中同时增长 p，l 和 r，然后在算法4.6中压缩他们的值。

2. 在算法4.5中使用三个嵌套的循环分别增长 p，l 和 r。

在小节4.8.7中，我们将比较在算法4.6中是否压缩 p，l 和 r，导致的在算法运行时间、解码器面积和延时方面的区别。

最后在小节 4.8.8中，我们将比较我们算法产生的解码器和手工书写的解码器在电路面积和延迟方面的差别。

4.8.2 PCI Express 2.0 编码器

该编码器遵从 PCI Express 2.0 标准[137]。在删除了所有的注释和空行之后，其源代码包含 259 行 verilog。

表 4.1 Benchmarks

	名字	个数 in/ouy	个数 reg/gate	电路 面积	编码器 描述	处理 方法
来自于[131]并有流控的测试集	PCIE2	10 / 11	22 / 149	326	PCIE 2.0[137]	小节4.8.2
	XGXS	10 / 10	17 / 249	572	10Gb 以太网 clause 48[136]	小节4.8.3
	T2Eth	14 / 14	53 / 427	947	UltraSPARC T2 的以太网模块	小节4.8.4
来自于[131]但没有流控的测试集	XFI	72 / 66	72 / 5086	12381	10Gb 以太网 clause 49[136]	在小节4.8.5中比较我们的算法和 Liu[133]
	SCRAM-BLER	64/64	58/353	1034	增加数据中的 01 翻转	
来自于[133]的测试集	CC_3	1/3	6/22	54	长度为 3 的卷机码	
	CC_4	1/3	7/26	63	长度为 4 的卷机码	
	HM(7,4)	4/7	3/38	103	输入 4 输出 7 的汉明码	
	HM(15,11)	11/15	4/102	317	输入 11 输出 15 的汉明码	

表4.2给出了所有输入和输出的描述。根据 8b/10b 编码机制的描述[143]，当 $TXDATAK \equiv 0$ 时，$TXDATA$ 可以为任何值。而当 $TXDATAK \equiv 1$ 时，$TXDATA$ 只能是 1C，3C，5C，7C，9C，BC，DC，FC，F7，FB，FD 和 FE。因此，我们写了一个断言以剔除不在编码表内的情形，并将其嵌入迁移函数 T。

算法4.3 在 0.475 秒内识别出了流控向量 $\vec{f} := CNTL\_TXEnable\_P0$。而后算法4.5 在 1.22 秒内推导出了 $valid(\vec{f}) := CNTL\_TXEnable\_P0$。最后算法4.6 在 0.69 秒内得到压缩后的 $p := 4$，$l := 0$ 和 $r := 2$。最后产生解码器函数使用了 0.26 秒。最终的解码器包含 156 个门和 0 个状态变量，面积为 366，延迟为 7.6。

本文算法的创新之处在于其处理流控机制的能力。谓词我们将展示编码器如何将无效的数据向量映射到输出向量 $\vec{o}$。通过研究编码器的代码，我们发现，当且仅当 $CNTL\_TXEnable\_P0 \equiv 0$ 成立，也就是 $TXDATA$ 和 $TXDATAK$ 无效时，输出 $HSS\_TXELECIDLE$ 为 1。因此，解码器将使用空闲字符 $HSS\_TXELECIDLE$ 来唯一决定流控向量 $CNTL\_TXEnable\_P0$。

4.8.3 10G 以太网编码器 XGXS

该编码器 XGXS 遵从 IEEE 802.3 标准的[136] 的 clause 48 。在删除空行和注释后，其包含 214 行 verilog。

表4.3给出了输入和输出变量列表。该编码器也使用 8b/10b 编码机制[143] ，它包含两个输入：8 位的有待编码数据 *encode_data_in* ，和 1 位的控制字符标志位 *konstant* 。根据编码表[143]，当 $konstant \equiv 0$ 时，*encode_data_in* 可以是任何值。而当 $konstant \equiv 1$ 时，*encode_data_in* 只能是 1C，3C，5C，7C，9C，BC，DC，FC，F7，FB，FD 和 FE。因此，我们将一个手工给出的断言嵌入 T 以剔除不在编码表内的情形。

算法4.3 在 0.31 秒内识别了流控向量 $\vec{f} := bad\_code$。然后算法4.5 在 0.95 秒内推导了谓词 $valid(\vec{f}) := bad\_code$。再次算法4.6 在 0.52 秒内得到压缩结果 $p := 4$，

表 4.2 PCI Express 2.0 编码器的输入和输出变量描述

	变量名	宽度	描述
输入	$TXDATA$	8	有待编码的数据
	$TXDATAK$	1	1 意味着 $TXDATA$ 是一个控制字符 0 意味着 $TXDATA$ 是普通数据
	$CNTL\_TXEnable\_P0$	1	1 意味着 $TXDATA$ 和 $TXDATAK$ 的有效性
输出	HSS_TXD	10	被编码的数据
	$HSS\_TXELECIDLE$	1	电磁空闲状态

表 4.3　10G 以太网编码器 XGXS 的输入和输出

	变量名	宽度	描述
输入	*encode_data_in*	8	将被编码的数据
	konstant	1	1 意味着 *encode_data_in* 是一个控制字符 0 意味着 *encode_data_in* 时普通数据
	bad_code	1	1 意味着 *konstant* 和 *encode_data_in* 是无效的
输出	*encode_data_out*	10	编码结果

$l := 0$ 和 $r := 1$。最后产生解码器使用了 0.17 秒。该解码器包含 163 个门和 0 个状态变量。面积为 370 ，延迟为 8.1。

虽然该算法使用了和上述 PCI Express 2.0 编码器相同的编码机制。然而它使用了完全不同的方法处理流控机制。该编码器并没有单独的输出用于表明输出的有效性。相反，所有输入的具体值及其有效性都统一编码在 *encode_data_out* 中。通过研究该编码器的源代码，我们发现当且仅当 $bad\_code \equiv 1$，既 *encode_data_in* 和 *konstant* 均无效时，输出变量 *encode_data_out* 将成为 0010111101。因此解码器能够使用 *encode_data_out* 来唯一决定 *bad_code*。

4.8.4　UltraSPARC T2 以太网编码器

该编码器来自于 UltraSPARC T2 开源处理器。它遵从于 IEEE 802.3 标准[136]的 clause 36。在删除空行和注释之后，它包含 864 行 verilog 源代码。

表4.4给出了输入和输出变量的列表。该编码器同样使用 8b/10b 编码机制[143]，但是采用了另外一种流控机制，完全不同于上述两个编码器。有待编码的数据仍然是 8 位的 *txd*。然而并不存在单独的有效位，而是在一个 4 位的 *tx_enc_ctrl_sel*

表 4.4　UltraSPARC T2 以太网编码器的输入输出列表

	变量名字	宽度	描述
输入	*txd*	8	有待编码的数据
	tx_enc_ctrl_sel	1	参见表4.5
	tx_en	1	传输使能
	tx_er	1	传输一个错误字符
输出	*tx_10bdata*	10	编码结果
	txd_eq_crs_ext	10	传输一个特殊错误字符 其中 $tx\_er \equiv 1$ 且 $txd \equiv 8'h0F$
	tx_er_d	1	传输一个错误字符
	tx_en_d	1	传输使能
	pos_disp_tx_p	1	正向 parity

表 4.5　UltraSPARC T2 以太网编码器的动作列表

动作名称	动作含义
‘PCS_ENC_K285	发送 K28.5 控制字符
‘PCS_ENC_SOP	发送 K27.7 控制字符
‘PCS_ENC_T_CHAR	发送 K29.7 控制字符
‘PCS_ENC_R_CHAR	发送 K23.7 控制字符
‘PCS_ENC_H_CHAR	发送 K30.7 控制字符
‘PCS_ENC_DATA	发送编码后的 txd
‘PCS_ENC_IDLE2	发送 K28.5 D16.2 序列
‘PCS_ENC_IDLE1	发送 D5.6 数据符号
‘PCS_ENC_LINK_CONFA	发送 K28.5 D21.5 序列
‘PCS_ENC_LINK_CONFB	发送 K28.5 D2.2 序列

中定义执行什么样的动作。细节如表4.5所示。很明显控制字符和流控机制被混合在 *tx_enc_ctrl_sel* 中。表4.5 的最后四种情形不能被唯一决定，因为他们无法和‘PCS_ENC_DATA 区分开来。因此我们使用一个断言将他们剔除。

算法4.3 在 3.76 秒内识别了流控信号 $\vec{f} := \{tx\_enc\_ctrl\_sel, tx\_en, tx\_er\}$。然后算法4.5 在 21.53 秒内推导了谓词 $valid(\vec{f}) := tx\_enc\_ctrl\_sel \equiv ‘PCS\_ENC\_DATA$。再次算法4.6 在 6.15 秒内得到压缩结果 $p := 5$，$l := 0$ 和 $r := 4$。最后产生解码器花费了 3.40 秒。解码器包含 401 个门和 9 个状态变量。面积为 920 ，延迟 10.2。

如表4.5的最后一列所示，前 5 种情况都有各自特殊的控制字符被赋予 *tx*_10*bdata*。因此解码器总能从 *tx*_10*bdata* 恢复出 *tx_enc_ctrl_sel*。

4.8.5　针对不具备流控机制的编码器比较我们的算法和 Liu et al.[133] 算法

表4.6 针对不具备流控机制的编码器比较了我们的算法和 Liu et al.[133] 的算法。

表 4.6　比较我们的算法和 Liu et al.[133] 的算法

名字	我们的算法				Liu et al.[133]		
	检查解码器存在的时间开销	产生解码器的时间开销	解码器面积	解码器延迟	检查解码器存在和产生解码器的时间开销	解码器面积	解码器延迟
XFI	13.24	6.13	3878	13.8	8.59	3913	12.5
SCRAMBLER	1.80	0.55	698	3.8	0.42	640	3.8
CC_3	0.06	0.03	116	8.5	0.21	104	9.1
CC_4	0.16	0.09	365	12.5	0.20	129	9.0
HM(7,4)	0.09	0.03	258	8.1	0.05	255	7.3
HM(15,11)	1.49	2.23	5277	13.7	2.02	3279	13.2

表 4.7　比较两种可能性：同时增长 p，l 和 r 或者单独增长

benchmarks	A1: 同时增长					A2: 单独增长				
	p,l,r	时间识别 $\vec{f}$	时间推导 $valid(\vec{f})$	时间压缩 p,l,r	整体时间开下	p,l,r	时间识别 $\vec{f}$	时间推导 $valid(\vec{f})$	时间压缩 p,l,r	整体时间开销
PCIE2	3,0,2	0.49	1.21	0.68	2.38	3,0,2	0.38	0.80	0.38	1.60
XGXS	3,0,1	0.31	0.88	0.52	1.71	3,0,1	0.23	0.58	0.30	1.11
T2Eth	4,0,4	4.28	15.17	6.25	25.70	4,0,4	15.47	13.85	6.19	35.51
XFI	2,1,0	4.59	3.60	9.55	17.74	2,1,0	3.52	2.75	10.05	16.32
SCRAMBLER	2,1,0	0.64	0.58	1.33	2.55	2,1,0	0.48	0.43	1.47	2.38
CC_3	3,2,2	0.01	0.01	0.04	0.06	3,2,2	0.01	0.01	0.01	0.03
CC_4	4,4,3	0.07	0.01	0.08	0.16	4,1,4	0.16	0.01	0.07	0.25
HM(7,4)	3,0,0	0.02	0.01	0.07	0.09	3,0,0	0.01	0.01	0.04	0.06
HM(15,11)	3,0,0	0.22	0.05	1.21	1.49	3,0,0	0.34	0.04	0.58	0.96

对于从 XFI 到 HM(15,11) 的 6 个测试电路，我们有他们的源代码，而 Liu et al.[133] 给出了他们的实验结果。因此我们能在这里比较我们的算法和 Liu et al.[133] 的结果。

通过比较第二和第三列之和和第六列，可见 Liu et al.[133] 比本文算法快很多。尤其是第二列。这种差距的主要原因在于本文算法需要逐一检查每个 $i \in \vec{i}$ 是否能够被唯一决定，而 Liu et al.[133] 可以在一次 SAT 求解之中检查所有的 $\vec{i}$。

另一个问题是本文算法的面积和延迟均大于 Liu et al.[133]，原因在于本文算法所采用的 Craig 插值算法实现仍不够优化，可以通过移植 ABC [142] 的相应代码得到改善。

4.8.6　比较两种可能性：同时增长 p，l 和 r 或者单独增长

在算法4.4中，我们同时增加 p，l 和 r ，并在算法4.6中压缩他们的冗余值。我们称其为 A1 方案。

小节4.6.2 给出了另一种可能性。它使用 3 个嵌套的循环来单独增长每一个 p，l 和 r 。我们称其为 À2 方案。

我们在表4.7中比较了这两种方案。

通过比较第 6 和 11 列中的整体时间开销，很显然 A2 在大多数情形下比 A1 快。只有 T2Eth 是一个例外。

这意味着我们应当使用 A2 而不是 A1 吗？答案是否定的。

表 4.8　在压缩和不压缩 l 和 r 的两种算法之间比较运行时间，电路面积和延迟

bench-marks	不压缩					使用算法4.6压缩					
	p,l,r	时间 生成 解码器	解码器 面积	寄存器 个数	最大 逻辑 延迟	时间 压缩 p,l,r	p,l,r	时间 生成 解码器	解码器 面积	寄存器 个数	最大 逻辑 延迟
PCIE2	3,3,3	0.44	382	11	7.5	0.68	3,0,2	0.28	366	0	7.6
XGXS	3,3,3	0.35	351	20	8.2	0.52	3,0,1	0.18	370	0	8.1
T2Eth	4,4,4	4.76	1178	9	10.9	6.25	4,0,4	3.41	920	9	10.2
XFI	2,2,2	10.67	5079	190	16.50	9.55	2,1,0	6.13	3878	58	13.8
SCRMBL	2,2,2	1.27	826	186	3.8	1.33	2,1,0	0.55	698	58	3.8
CC_3	3,3,3	0.04	117	11	9.2	0.04	3,2,2	0.03	116	9	8.5
CC_4	4,4,4	0.05	154	10	9.6	0.08	4,4,3	0.09	365	14	12.5
HM(7,4)	3,3,3	0.05	262	21	7.2	0.07	3,0,0	0.03	258	0	8.1
HM(15,11)	3,3,3	2.98	5611	45	13.5	1.21	3,0,0	2.23	5277	0	13.7

从小节4.6.2可知，A1 需要调用 SAT 求解器的次数为 $O(n)$，其中 $n = max(p,l,r)$。而 A2 需要的次数为 $O(n^3)$。对于比较小的 n，两者并没有很大的区别。而对于较大的 n，比如 T2Eth，A1 对 A2 的优势是显著的。

因此 A2 在小电路上有优势，而 A1 在大电路上有优势。因此我们仍然选择 A1。也就是首先同步增加 p，l 和 r，然后在算法4.6中压缩他们。

CC_4 是唯一一个在第 2 列和第 7 列具有不同 p，l 和 r 的测试电路。这是由于 l 和 r 的不同增长顺序导致的。对于 A1 方案，其解码器包含 14 个状态变量，206 个门，490 面积和 13.3 延迟。对于 A2 方案，其解码器包含 10 个状态变量，61 个门，154 面积和 9.6 延迟。因此 A2 方案比 A1 好很多。但是这仍然不意味着我们应当使用 A2。具体原因我们将在下一小节得到更多实验数据支撑之后进一步展开解释。

4.8.7　在压缩和不压缩 l 和 r 的两种算法之间比较运行时间、电路面积和延迟

为了改善解码器的面积和延迟，算法4.6 被用于在产生解码器之前压缩 l 和 r。表4.8展示了其效果。

第一列是测试电路名字。当算法4.6 没有被使用时，第 2 列到第 6 列分别给出了 p，l 和 r 的值、产生解码器的运行时间、解码器面积、解码器包含的状态变量个数和解码器最大的逻辑延迟。当算法4.6 被使用的时候，这些数据在最后 5 列给出。而第 7 列给出了 l 和 r。

通过比较 2-6 列和 8-12 列，很明显算法4.6显著的压缩了 l 和 r。

CC_4 再次引起我们的注意。从第 4 列到第 6 列，我们发现电路面积和延迟非常类似于上一小节的 A2 情形。而他的 p，l 和 r 则类似于 A1 情形。这意味着分 CC_4 的解码器有至少两种差别很大的实现方式。而具体哪一种被选中取决于 SAT 求解器和 Craig 插值算法内部的某些不稳定因素。这回答了我们在上一小节的疑惑，即 A1 方案中的电路质量下降并不是由 A1 导致的。我们仍然应当使用 À1。

4.8.8 在我们的算法和手工书写的解码器之间比较电路面积和延迟

表4.9 在我们的算法和手工书写的解码器之间比较电路面积和延迟。CC_3，CC_4，HM(7,4) and HM(15,11) 没有在该表中是因为我们没有他们的手写解码器。

很明显在大多数情况下我们的解码器比手工解码器更小也更快。少数的两个例外是 T2Eth 和 XFI ，我们产生的解码器比手工解码器稍微大一些。

4.9 结论

本文提出了第一个能够处理流控机制的对偶综合算法。实验结果表明本文算法能够为多个来自于实际工业项目的复杂编码器，包括 PCI Express [137] 和以太网[136]，正确的生成包含流控机制的解码器。

表 4.9 在我们的算法和手工书写的解码器之间比较电路面积和延迟

bench-marks	本文算法		手工书写的解码器	
	面积	最大逻辑演出	面积	最大逻辑延迟
PCIE2	366	7.6	594	9.7
XGXS	370	8.1	593	11.0
T2Eth	920	10.2	764	11.7
XFI	3878	13.8	3324	28.1
SCRAMBLER	698	3.8	1035	6.4

第五章　结构感知的 CNF 混淆算法

5.1　引言

软硬件设计验证过程中，通过 Tseitin 编码[23] 将硬件电路、软件代码及待验证属性转化为统一的 CNF 公式，并进行 SAT 求解，以判断系统是否具有指定特性或是符合规定要求。SAT 求解问题是面向位级的，解空间随软硬件系统规模成指数级增长，对计算能力的扩展性要求高，适合部署在能够提供弹性计算资源的开放环境下。但是，研究表明，虽然电路在 Tseitin 编码过程中损失了很多有用的逻辑信息，编码后生成的 CNF 公式中仍然携带了硬件电路结构等敏感信息；另一方面，由于 SAT 问题的结果反映待验证属性是否符合预期，直接影响设计方案的成败，因此错误结果也是不可容忍的。

为避免开放环境下可能出现上述问题，本章以电路设计验证中 SAT 问题为研究对象，对系统进行建模，分析其中存在的安全威胁，并进而在此基础上提出了基于 CNF 混淆的安全可验证 SAT 求解方法。在从理论上证明混淆算法的正确性之后，本文选取 ISCAS89 中的典型电路，对算法进行了性能测试和评估。

5.2　问题描述

本节将以云计算环境为例，来建立开放环境下 SAT 求解的系统模型，并分析其中的安全威胁。在研究中，存在两种类型的云，私有云和公共云：

1. 私有云是可信的，表现在两方面，一是数据的隐私性可以得到保证，二是计算结果的正确性可以确认；但私有云的计算及存储能力受限，仅仅适合处理对计算资源需求量小且相对稳定的简单计算。

2. 公共云可以提供弹性的计算及存储资源，适合承担计算资源需求量大的复杂计算任务；但由于公共云处于互联网环境下，这种开放计算环境存在中存在诸多安全威胁，如“诚实但好奇”的计算参与者，这些计算参与者会尽力完成所有的计算任务，获得正确的结果，但是它们会试图获取计算数据中隐藏的信息。

传统的综合与验证工具的核心框架通常包含以下主要功能模块：

1. 抽象问题表示：该模块用于管理与特定推理过程和引擎无关，但是与问题本身密切相关的数据结构，如 Reduced Boolean circuits[40] 和 And-Inverter Graph[41] 等。

2. 问题编码：该模块用于将特定的抽象问题表示，转换为满足特定推理引擎，如二叉决策图 (BDD) 或可满足问题（SAT）要求的数据结构，以便进行高效的推理工作。针对 SAT，该模块通常使用在空间和时间方面均具有多项式复杂性的 Tseitin[23] 编码。

3. 二叉决策图 (BDD) 和 SAT 推理引擎：这两个模块负责具体的推理工作。

基于对形式化验证[144–148] 和对偶电路综合[149, 150, 150–153] 方面的研究，发现一个通常的综合验证问题可以简述为：将硬件网表和需要验证的属性通过抽象问题表示和问题编码转换为 BDD 或 CNF 公式，然后使用 BDD 推理引擎或 SAT 推理引擎对公式进行推理并返回结果。

抽象问题表示和问题编码转换通常是多项式复杂度的，计算开销相对稳定；BDD 受到归一化表示方式导致的状态空间爆炸问题的困扰，通常仅用于需要归一化特性的小规模推理问题，如抽象迁移关系的构造和精炼[3] 等；因此这两类计算适合部署在私有云环境下。SAT 则通常较少受到状态空间爆炸的影响，且天生具有内在的并行性和可扩展性，因此较为适合运行于公有云的核心推理引擎。

结合这些情况，CNF 公式将在私有云中产生，SAT 求解器部署在公共云，用于求解 CNF 公式，并将结果返回给私有云。由于公共云环境下存在“诚实但好奇”的计算参与者，他们会尽力完成所有的计算任务以便获得正确的 CNF 公式的解，但是它们也试图从 CNF 公式或解中获得他们需要的信息，例如电路结构信息。文献[11, 12, 154, 155] 研究了抽取和利用 CNF 公式中电路结构信息的方法。Roy 等人[11] 和 Fu 等人[12] 给出了基于子图同构以及模式匹配的电路结构抽取算法，并能最大限度的恢复出 CNF 公式中的电路结构。这些技术可能会被潜在的攻击者利用来抽取敏感的电路结构信息。因此，CNF 公式包含的结构信息应该被当做隐私加以保护。

5.3 基于 CNF 混淆的 SAT 求解方法

5.3.1 SAT 求解框架

设计基于开放环境下 SAT 求解框架，将考虑下面三个因素：

1. 可移植性：目前的 SAT 求解器集成了冲突检测[32] 等高效机制，因此我们希望可以将其作为黑盒直接使用，而不是像[49] 等基于全同态的方案那样，需要重新构造问题并试图使用新的求解算法。

2. 隐形性[97]：求解框架应该保护 CNF 公式中的电路结构不会被获取。

3. 结果可验证：计算结果可以验证。

4. 开销：框架不应该引入太多的开销。

上面的要求具体说来可以归结为这样一个有趣的问题：将 SAT 问题编码为 CNF 公式外包到云中，由未经修改的经典 SAT 求解器求解；并且不希望 SAT 求解器知道实际 SAT 问题的原始结构。

直觉来看，一个简单的方法就是将原始 CNF 公式 F_C 和一个与其无变量交集的可满足公式 F_H，简单混合在一起，并将混合后的 CNF 公式 $F_C \wedge F_H$ 外包的云上。由于可满足公式 F_H 必然存在解，那么如果原始公式 F_C 有解，其解将会混杂在混合后的解中。这种方法非常简单易行. 但是，基于分区[156] 的方法可以将两个不相关的公式分离开来，从而得到原始 CNF 公式 F_C。

再来考虑一个改进的方法: 任意公式 F_C 和一个可满足公式 F_H，F_C 和 F_H 没有公共变量，也即：$V_{F_C} \cap V_{F_H} = \phi$。将 F_C 和 F_H 无缝混合以便于隐藏 F_C；同时，保持 F_C 中的所有解都包含在新的公式中。而要实现 F_C 和 F_H 无缝混合，需要实现两个公式子句的交织，需要存在既包含 F_H 变量，同时又包含 F_C 变量的子句。根据 CNF 公式的定义可知其具有如下特性：对任意 CNF 公式 F_C，在其子句中加入新的文字可能会扩展解空间；而另一方面，在 F_C 中加入包含 F_C 中变量的子句则会缩减解空间。

那么，如何在此情况下保证 F_C 所有的解仍旧保留在新的公式中? 为了实现该目标，本章给出了安全可验证的 SAT 求解框架。为了混淆 CNF 公式 F_C，根据后续章节介绍的 SSH 规则策略，采用随机的方式在公式 F_C 的子句中加入新的文字，并根据加入的文字在公式中加入新的子句。新加入的文字和一部分新的子句来自于被称为单一 Husk 公式 F_H（见定义5.1）的噪音公式。SSH 规则保证原始 CNF 公式 F_C 可以和单一 Husk 公式 F_H 无缝地混合在一起，达到2) 和 4) 的要求。

定义 5.1 (**单一 Husk 公式**): 单一 Husk 公式是仅有一个可满足解的 CNF 公式，并且解中变量的赋值是非特异的，即他们不是全 T 或全 F。

框架的详细实现在图 5.1中给出。在该框架下，SAT 问题的求解包含了以下 4 步。

- **步骤 1**：产生器产生一个单一 Husk 公式 F_H 和它的一个解 R_H。
- **步骤 2**：混淆器将原始 CNF 公式 F_C 和单一 Husk 公式 F_H 混合，并产生一个新的 CNF 公式 $F_O := F_H \wedge F_C$。
- **步骤 3**：F_O 由位于公共云上的 SAT 求解器求解，并返回结果 S_O。

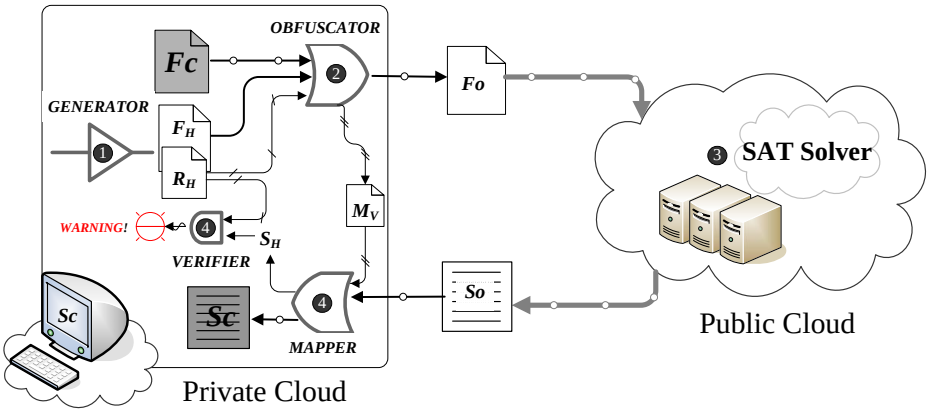


图 5.1　基于 CNF 公式混淆的安全可验证 SAT 求解框架

算法 5.1 GENERATOR

1: input : NULL;
2: output : Husks CNF F_H and Husks result R_H;
3: Generating prime numbers p_A;
4: $\Phi = M(I_1 \neq 1, I_2 \neq 1, O = p_A * p_A)$;
5: $F_H = Tseitin(\Phi)$;
6: $R_H = \{I_1 \equiv p_A, I_2 \equiv p_A\}$;

- **步骤 4**：映射器从 S_O 中抽取出 S_C，验证器确认 S_C 是 F_C 的解。

步骤 3在开放环境下的公共云中运行，其余的步骤在可信的私有云上运行。产生器，混淆器，映射器和验证器算法将在5.3.2，5.3.3和5.3.4 小节中分别介绍。

5.3.2　单一 Husk 公式产生算法

产生可满足 CNF 公式的算法有多种[157, 158]. 仅有一个可满足解的单一 Husk 公式 (见定义5.1) 采用质因数的方法构造[158]，其中的具体实现在算法5.1中描述。

首先，给定质数 p_A(第 3行)，将 $p_A * p_A$ 赋值给乘法器 M 的输出，并且限制 $I_1 \neq 1$ and $I_2 \neq 1$ (第4行)。其中，I_1 和 I_2 是 M 的输入。

第二，将乘法器 M 编码为 CNF 公式 $Tseitin(M)$(第5行)。

为了满足 $Tseitin(M)$，M 的两个输入的唯一解一定是 $R_H := \{I_1 \equiv p_A, I_2 \equiv p_A\}$。

5.3.3　解空间保持的混淆算法

让我们继续考虑开篇提到的有趣问题：如何将 SAT 问题编码为 CNF 公式外包到云中，由云端的 SAT 求解器求解；并且不希望 SAT 求解器获知原始 SAT 问题的结构。对来源于硬件验证的 SAT 问题而言，这个问题就具化为：一方面，希望求解器无法获取原 CNF 公式携带的电路结构信息，可使用“抗结构感知”来描

述这一特性；而另一方面，希望求解器还能够给出正确解。这不仅包括解的取值是正确的，还包括解的个数也是正确的。本文的后续使用“保持解空间”来描述这一特性。

5.3.3.1 解空间保持规则

一个能同时实现“抗结构感知”和“解空间保持”的方法，是将真实 CNF 公式 F_C 和一个在变量上和其无交集的可满足公式 F_H，混合在一起，将混合后的 CNF 公式 F_O 外包的云上。由于 SAT 求解器拿到的不是真实 CNF 公式，原始公式中包含的结构信息自然无从获得，这就实现了“抗结构感知”。由于可满足公式 F_H 必然存在解，那么如果真实公式 F_C 有解，其解将会混杂在混合后公式 F_O 的解中，混合后公式就是可满足的 (SAT)。而如果如果原始公式 F_C 没有解，混合后公式 F_O 也将是不可满足的 (UNSAT)。因此这种混合也是解空间保持的。唯一不足的是，简单混合无法防止基于分区方法[156] 的攻击。

改进的方法是将原始 CNF 公式 F_C 和可满足公式 F_H 无缝混合，以便于隐藏 F_C。为实现 F_C 和 F_H 无缝混合，直觉的方法是将 F_H 中的变量加入到 F_C 的子句中，或使用 F_C 和 F_H 中的变量产生新的子句，这样就可以构造同时包含 F_C 和 F_H 中文字的子句。但是根据 CNF 公式的定义可知其具有如下特性：对任意 CNF 公式 F_C，在子句中加入新的文字可能会扩展解空间；而另一方面，在 F_C 中加入包含 F_C 中变量的子句则会缩减解空间。那如何在此情况下保证 F_C 的解仍旧保留在新的公式中? 在给出具体答案之前，首先澄清下面的概念。

定义 5.2 (**解包含关系** $S_C \subseteq S_O$): CNF 公式式 F_C 和 F_O 具有 n_{F_C} 个公共变量 $x_1, ..., x_{n_{F_C}}$。并且有 $|V_{F_C}| \equiv n_{F_C}$，$|V_{F_O}| \equiv n_{F_O}$，$n_{F_O} \geqslant n_{F_C} > 0$。$S_C$ 和 S_O 分别是 F_C 和 F_O 的解。在 S_C 和 S_O 中，n_{F_C} 个公共变量的赋值是相同的，也就是 $S_C = \{x_1 = B_1, ..., x_{n_{F_C}} = B_{n_{F_C}} | B_i \in \{T, F\},\ 1 \leqslant i \leqslant n_{F_C}\}$, $S_O = \{x_1 = B_1, ..., x_{n_{F_C}} = B_{n_{F_C}}, ..., x_{n_{F_O}} = B_{n_{F_O}} | B_i \in \{T, F\},\ 1 \leqslant i \leqslant n_{F_O}\}$。称解 S_C 包含于 S_O，记为 $S_C \subseteq S_O$。

定义 5.3 (**解空间等价** (SSE)): CNF 公式 F_C 有 n 个解 $\{S_{C_1}, ..., S_{C_n}\}$; CNF 公式 F_O 也有 n 个解 $\{S_{O_1}, ..., S_{O_n}\}$，并且对于任意 $i \in [1, n]$, $S_{C_i} \subseteq S_{O_i}$。称 F_O 解空间等价于 F_C，记为 $F_C \equiv_{SSE} F_O$。

为了在新公式中保持 F_C 公式中的所有解, 我们将在下面给出解空间保持 (SSH) 规则。本章介绍的混淆器将使用 SSH 规则，将 Husks 公式 F_H 嵌入到原始公式 F_C 中，产生一个新的 CNF 公式 F_O。这将防止攻击者通过模式识别或是子图同构技术获取公式中携带的电路结构信息。SSH 规则同时还将保证混淆后的解空间与原公式解空间等价。

解空间保持（SSH）规则:

规则 1：子句替换规则: 对任一子句 $c \in F_C$，从 R_H 中任取出变量 v，并按照下列规则插入到子句 c 中：如果在 R_H 中变量 v 的赋值是 T，则插入负文字 $\neg v$；如果在 R_H 中变量 v 的赋值是 F，则插入正文字 v；用新生成的子句代替原始子句 c。

规则 2：子句生成规则: 任取 R_H 中的文字和 F_C 中的变量创建新的子句。

定义 5.4 (**Obf**(F_C, F_H, R_H)): 对任意公式 F_C，和可满足公式 F_H 及其一个赋值 R_H，$Obf(F_C, F_H, R_H)$ 是在基于 SSH 规则将 F_C 和 F_H 混合后得到的公式。

定义 5.5 (SSE **混淆**): 如果 F_H 是单一 Husk 公式并且 R_H 是它的唯一解，就称 $Obf(F_C, F_H, R_H)$ 为解空间等价的混淆，称为 **SSE** 混淆。

对基于 SSH 规则的 **SSE 混淆**，下列定理成立。

定理 5.1 (SSE Obfuscation): 对任意 CNF 公式 F_C 和单一 Husk 公式 F_{sH}，如果 $V_{F_C} \cap V_{F_{sH}} = \phi$，并且 R_{sH} 是 F_{sH} 的唯一解，则 $Obf(F_C, F_{sH}, R_{sH}) \equiv F_C \wedge F_{sH}$。

基于定理5.1 和定义5.3，有:

定理 5.2: 对任意 CNF 公式 F_C 和单一 Husk 公式 F_{sH}，如果有 $V_{F_C} \cap V_{F_{sH}} \equiv \phi$，而且 R_{sH} 是 F_{sH} 的唯一解，则 $Obf(F_C, F_{sH}, R_{sH}) \equiv_{SSE} F_C$。

定理5.1和5.2的严格证明将在5.4.1.1小节给出。

经过 SSE 混淆生成的 CNF 公式 $F_O = Obf(F_C, F_{sH}, R_H)$，包含了 F_C 和 F_{sH} 中的所有变量；并且根据 SSH**规则 1**，F_C 中的部分子句被加入了 F_{sH} 中的变量对应的文字，而成为了新的子句。由于 F_{sH} 解的非特异性，这些文字不全为正也不全为负；还有部分子句是根据 SSH**规则 2**，由分别取自 F_C 和 F_{sH} 的变量文字组合而成。从形式上看，F_O 是一个合法的 CNF 公式。而根据定理5.1，可知混合后公式 F_O 的解是 F_C 和 F_{sH} 的交集。由于公式 F_C 和 F_{sH} 无公共变量，又由于 F_{sH} 只有唯一解，这就使得 F_O 的解的个数和 F_C 的一致。可以得到如下的结论：

1. F_C 是不可满足的当且仅当 F_O 不可满足，并且 F_C 的不可满足核可以通过从 F_O 的不可满足核中删除 F_H 中的文字获得。

2. F_C 可满足当且仅当 F_O 是可满足的，并且 F_C 的解可以通过将 F_O 的解投影到 F_C 的变量集中获得。

综上所述，经过 SSH 混淆变换后，F_O 和 F_C 在形式上相似，F_O 和 F_C 可以使用同样的 SAT 求解器求解。在不知晓 R_H 的情况下，F_C 将很难被从 F_O 剥离；另一方面，F_O 的解空间和 F_C 解空间等价。通过 SSH 规则，算法5.4所示的混淆器 OBFUSCATOR 可以在子句中添加新文字并且创建新的子句，同时保证解空间不被扩展或缩减；这就实现了与全同态加密相似的混淆效果。

算法 5.2 mark

```
1: mark;
2: input : CNF formula S;
3: output : marked S ;
4: KeyClauseSet := {}
5: OutLiteralMap := {}
6: for (C ∈ S) do
7:     if |C| ≡ 2 then
8:         KeyClauseSet := KeyClauseSet ∪ C
9:         for (l ∈ C) do
10:            OutLiteralMap := OutLiteralMap ∪ (C, l)
11:        end for
12:    end if
13: end for
14: return (KeyClauseSet, OutLiteralMap)
```

算法 5.3 generate_new_clause

```
1: generate_new_clause;
2: input : key clause C, Husk literal lit, OutLiteralMap;
3: output : new clause C1;
4: 假设 (C, l) ∈ OutLiteralMap
5: C1 = lit ∪ ¬l ;
```

5.3.3.2 OBFUSCATOR 算法

OBUFSCATOR 算法遵循上述 SSH 规则，混淆 CNF 公式 F_C，从而防止 F_C 所携带的电路结构信息被潜在的攻击者获取。SSH 规则保证了解空间的不变。如何达到隐藏电路结构信息的目的，则需要确定如何在子句中添加新的文字以及构造新的子句。

为了达到上述目标，OBFUSCATOR 首先会在第 3 行使用算法5.2中的 mark 算法，得到 CNF 公式中的关键子句和输出文字集合，以确定究竟选取哪些子句加入文字；而后通过在关键子句中加入噪音文字，破坏原有的结构。OBFUSCATOR 的详细实现在算法5.4中，

5.3.4 映射和验证算法

在公共云上的求解器完成求解后，将 F_O 的解 S_O 返回给私有云。和混淆器相对应，在私有云中使用 *MAPPER_VERIFIER* 将 F_C 的解从 S_O 中过滤出来。*MAPPER_VERIFIER* 的实现在算法5.5中。

算法 5.4 OBFUSCATOR

```
1: input : The original CNF F_C，Husks CNF F_H，Husks result R_H
2: output : The obfuscated CNF F_O，variable mapping M
3: (KeyClauseSet, OutLiteralMap) := mark(F_C);
4: for c ∈ F_C do
5:     if c ∈ KeyClauseSet then
6:         lit =get literal ∈ R_H;
7:         c = c ∪ ¬lit;
8:         nc = generate_new_clause(c, lit, OutLiteralMap);
9:         F_C = F_C ∪ nc;
10:    end if
11: end for
12: for c ∈ F_C do
13:    averagelen = σ_{c'∈F_C}|c'| / |F_C| ;
14:    while |c| < averagelen do
15:        lit =get literal ∈ R_H ;
16:        while ¬lit ∈ c do
17:            lit=get literal ∈ R_H ;
18:        end while
19:        c = c ∪ ¬lit;
20:    end while
21:    M =remap all variable in F_C ∪ F_H ;
22:    F_O =reorder all clause in F_C ∪ F_H ;
23: end for
```

根据定理5.1，如果结果是 UNSAT，那么原始的 CNF 公式也是 (第3行)。如果结果是 SAT，第8-10 行将解投影到 F_C 和 F_H 的变量上，以获得 S_C 和 S_H，分别作为 F_C 和 F_H 的候选解。第12-17 行检测 S_H 是否等于 R_H。如果等于，S_C 就是 F_C 的真实解。否则，S_C 是错误解，说明云端给出的结果是不可信的。

解的投影过程依赖于变量映射表 M，它由 OBFUSCATOR(算法5.4的21 行) 创建。$M[var].variable$ (第8行) 表示了 var 的原始变量名，$M[var].formula$ (第9行) 表示 var 所属于的公式，可以是 F_C 或 F_H。

算法 5.6 MAPPER and VERIFIER

```
1: input : Obfuscated result S_O，variable mapping table M，Husk result R_H;
2: output : Result S_C;
3: if S_O is UNSAT then
4:         return UNSAT ;
5: end if
6: for lit ∈ S_O do
7:         var = abs(lit);
8:         rvar = M[var].variable;
9:         if M[var].formula is F_C then
10:                S_C[rvar] = lit > 0?rvar : ¬rvar ;
11:        else
12:                Hlit = lit > 0?rvar : ¬rvar;
13:                if R_H[rvar] ≠ Hlit then
14:                        alert(”Something wrong with Solution!!! ”);
15:                        break;
16:                end if
17:        end if
18: end for
19: print ” SAT solution is S_C ”;
```

5.4 理论分析和实验评估

5.4.1 理论分析

5.4.1.1 正确性证明

根据定理5.1，在 SSH 规则下，原始的 CNF 公式可以和单一 Husk 公式无缝混合，而不会改变解空间。在本节中，将证明这一定理。在此之前，首先给出 SSH 混淆过程的逻辑步骤描述和相关引理，这是定理证明的基础。

算法5.4实现的基于 SSH 规则的混淆过程，从逻辑上看分为如下三步：

Procedure 5.4.1.1.

输入：公式 F_C，单一 Husk 公式 F_H 及其唯一解 R_H。其中 F_C 包含了算法5.4第5行的**关键子句**和**非关键子句**，分别记为 F_{Ck} 和 F_{Cn}。

输出：公式 F_O.

步骤 1：对关键子句 $c \in F_{Ck}$，从 R_H 中取出文字 lit。根据 SSH 规则 1 将 $\neg lit$ 加入到 c(算法5.4的7，19行)。生成子句的集合记为 S_3。

步骤 2：根据 SSH 规则 2，使用 R_H 中文字 lit 和 c 中的输出变量，产生新的子句 (算法 5.4的8行，算法5.2 的5行)。新产生的子句集合记为 S_4。

步骤 3：将 S_3、S_4、F_H 和 F_{Cn} 混合产生 F_O (算法5.4的7，19，9 22 行).

end Procedure.

引理 5.1 (Singular Husk Equation(SHE))：对单一 Husk 公式 F_H，记 $|V_{F_H}| = n$。假设 F_H 的唯一解为 S_H。同时假设 B_T 为 S_H 中正文字集合，而 B_F 为 S_H 中负文字集合。令 $F_{sH} = F_H \wedge (\bigwedge_{y_i \in B_T} y_i) \wedge (\bigwedge_{y_j \in B_F} y_j)$，则有 $F_H \equiv F_{sH}$，即他们具有相同的解。

证明：因为 F_H 有唯一解 S_H，可以构造 F_{lsH}

$$F_{lsH} = (\bigwedge_{y_i \in B_T} y_i) \wedge (\bigwedge_{y_j \in B_F} y_j) \tag{5.1}$$

则有

$$F_{lsH} \equiv T \tag{5.2}$$

而 $F_{sH} = F_H \wedge F_{lsH}$，因此必有

$$F_H \equiv F_{sH} \tag{5.3}$$

根据引理5.1，一个单一 Husk 公式等价于解文字的合取。

引理 5.2 (AND Hold Obfuscation)：公式 F_C 和 $F_{RH} \wedge F_{sH}$，并且 R_H 是 $F_{RH} \wedge F_{sH}$ 的一个赋值，则有

$$Obf(F_C, F_{RH} \wedge F_{sH}, R_H) \equiv Obf(F_C, F_{RH}, R_H) \wedge Obf(F_C, F_{sH}, R_H) \tag{5.4}$$

证明：假设

1. 假设有子句 $A = a \vee X$ 和子句 $B = b$，并且 $b \notin X$，另有任意公式 F_{Cn} 和 F_{sH}；
2. 令 $F_{Ck} = A$，$F_C = F_{Ck} \wedge F_{Cn}$，$F_{RH} = B$，$F_H = F_{RH} \wedge F_{sH}$，并且 $R_H = \{b \equiv T\}$；
3. 令 $F_O = Obf(F_C, F_H, R_H)$

根据 **Procedure** 5.4.1.1，按下列 3 个步骤构造 F_O。

1. 根据 R_H 和规则 1，有子句 $C = A \vee \neg b$，并且 $S_3 = C$；

2. 根据 R_H 和规则 2，并且有文字 $a \in A$，可创建子句 $D = \neg a \vee b$；我们令 $S_4 = D$；

3. 令 $F_{dC} = S_3 \wedge S_4 \wedge F_{Cn}$，则 $F_O = F_H \wedge F_{dC}$

因此，有下列推演步骤：

$$\begin{aligned} F_O = F_H \wedge F_{dC} \qquad F_H = F_{RH} \wedge F_{SH} &\Longrightarrow \\ F_O = (F_{RH} \wedge F_{SH}) \wedge F_{dC} &\Longrightarrow \\ F_O = (F_{RH} \wedge F_{dC}) \wedge (F_{SH} \wedge F_{dC}) &\Longrightarrow \end{aligned} \tag{5.5}$$

根据 **Procedure** 5.4.1.1，F_{dC} 仅仅和 F_C、R_H 相关，因此有：

$$F_{SH} \wedge F_{dC} \equiv Obf(F_C, F_{SH}, R_H) \tag{5.6}$$

$$F_{RH} \wedge F_{dC} \equiv Obf(F_C, F_{RH}, R_H) \tag{5.7}$$

根据式5.5)、5.6）和5.7），则有

$$F_O = Obf(F_C, F_{RH}, R_H) \wedge Obf(F_C, F_{SH}, R_H) \tag{5.8}$$

证明完成。 ■

引理 5.3 (Unique Positive literal SSE Obfuscation): 对任意公式 F_C，有

$$Obf(F_C, B \equiv b, b \equiv T) \equiv F_C \wedge b \tag{5.9}$$

证明：假设

1. 对于 $A = a \vee X$ 和 $F_{Ck} = A$，有 F_C=$F_{Ck} \wedge F_{Cn}$；

2. F_H=B，$B = b$ 且 $b \notin X$，R_H=$\{b \equiv T\}$；

3. 令 $F_O = Obf(F_C, F_H, R_H)$。

根据 **Procedure** 5.4.1.1，按下列 3 个步骤构造 F_O。

1. 根据 $(b \equiv T) \in R_H$ 以及规则 1，构造子句 $C = A \vee \neg b$，并且 $S_3 = C$。

2. 根据 $(b \equiv T) \in R_H$，文字 $a \in A$ 和规则 2，构造子句 $D = \neg a \vee b$，令 $S_4 = D$.

3. 令 $F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn}$.

根据步骤3)，有以下推演步骤:

$$\begin{aligned}
F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn} \quad & S_3 = C\ S_4 = D \implies \\
F_O = F_H \wedge C \wedge D \wedge F_{Cn} \quad & F_H = B\ B = b \implies \\
F_O = b \wedge C \wedge D \wedge F_{Cn} \quad & C = A \vee \neg b \implies \\
F_O = b \wedge (A \vee \neg b) \wedge D \wedge F_{Cn} \quad & \implies \\
F_O = b \wedge A \wedge D \wedge F_{Cn} \quad & D = \neg a \vee b \implies \\
F_O = b \wedge A \wedge (\neg a \vee b) \wedge F_{Cn} \quad & \implies \\
F_O = b \wedge A \wedge F_{Cn} \quad & F_{Ck} = A \implies \\
F_O = b \wedge F_{Ck} \wedge F_{Cn} \quad & F_C = F_{Ck} \wedge F_{Cn} \implies \\
F_O = F_C \wedge b &
\end{aligned} \tag{5.10}$$

证明完成。 ■

引理 5.4 (Unique Negative literal SSE Obfuscation): 任意公式 F_C，有 $Obf(F_C, B = \neg b, b = F) = F_C \wedge \neg b$

证明：假设

1. 对于子句 $A = a \vee X$，任意公式 F_{Cn}，以及子句 $B = \neg b$ 使得 $b \notin X$；

2. 令 $F_{Ck} = A$，$F_C = F_{Ck} \wedge F_{Cn}$，$F_H = B$，且 $R_H = \{b \equiv F\}$；

3. 令 $F_O = Obf(F_C, F_H, R_H)$

可知 $F_O = F_C \wedge F_H$.

根据 **Procedure** 5.4.1.1，按照下面 3 个步骤构造公式 F_O。

1. 根据 R_H 以及规则 1，构造子句 $C = A \vee b$。令 $S_3 = C$；

2. 根据 R_H 以及规则 2，对于文字 $a \in A$，构造子句 $D = \neg a \vee \neg b$。令 $S_4 = D$；

3. 令 $F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn}$。

有以下推演步骤：

$$
\begin{aligned}
F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn} \quad & F_H = B \quad \Longrightarrow \\
F_O = B \wedge S_3 \wedge S_4 \wedge F_{Cn} \quad & S_3 = C \quad \Longrightarrow \\
F_O = B \wedge C \wedge S_4 \wedge F_{Cn} \quad & S_4 = D \quad \Longrightarrow \\
F_O = B \wedge C \wedge D \wedge F_{Cn} \quad & B = \neg b \quad \Longrightarrow \\
F_O = \neg b \wedge C \wedge D \wedge F_{Cn} \quad & C = A \vee b \quad \Longrightarrow \\
F_O = \neg b \wedge (A \vee b) \wedge D \wedge F_{Cn} \quad & \quad \Longrightarrow \\
F_O = \neg b \wedge A \wedge D \wedge F_{Cn} \quad & D = \neg a \vee \neg b \quad \Longrightarrow \\
F_O = \neg b \wedge A \wedge (\neg a \vee \neg b) \wedge F_{Cn} \quad & \quad \Longrightarrow \\
F_O = \neg b \wedge A \wedge F_{Cn} \quad & F_{Ck} = A \quad \Longrightarrow \\
F_O = \neg b \wedge F_{Ck} \wedge F_{Cn} \quad & F_C = F_{Ck} \wedge F_{Cn} \quad \Longrightarrow \\
F_O = F_C \wedge \neg b \quad & \quad
\end{aligned}
\tag{5.11}
$$

证明完成。 ■

根据引理5.3 和5.4，单文字混淆后解空间保持等价。

在上述引理的基础上，接下来讨论基于单一 Husk 公式的 SSE 混淆。

Theorem 5.1 解空间等价 (SSE) 混淆

对任意 CNF 公式 F_C，和单一 Husk 公式 F_{sH}，如果

- $V_{F_C} \cap V_{F_H} = \phi$，且 R_H 是 F_H 的唯一解。
- $F_O = Obf(F_C, F_H, R_H)$。

则 $F_C \wedge F_H \equiv F_O$。

证明：假设 B_T 时 R_H 中的正文字集合，B_F 是 R_H 中的负文字集合。

根据 **Procedure**5.4.1.1，按下列步骤构造 F_O：

1. 令 $F_{Op} = F_C$。对 $y_i \in B_T$，令

$$F_{Op} = Obf(F_{Op}, \{\{y_i\}\}, y_i) \tag{5.12}$$

其中 $\{\{y_i\}\}$ 是仅包含一个单文字子句 $\{y_i\}$ 的公式，而 y_i 是该公式的唯一解。

2. 令 $F_{On} = F_{Op}$。对 $y_j \in B_F$，令

$$F_{On} = Obf(F_{Op}, \{\{y_i\}\}, y_i) \tag{5.13}$$

3. $F_O = F_{On} \wedge F_H$.

根据引理5.2，5.3以及步骤1)，有:

$$F_{Op} \equiv F_C \wedge (\bigwedge_{y_i \in B_T} y_i) \tag{5.14}$$

根据引理 5.2，5.4以及步骤2)，有:

$$F_{On} \equiv F_{Op} \wedge (\bigwedge_{y_j \in B_F} y_j). \tag{5.15}$$

根据步骤3) 和式 (5.14) (5.15)，有:

$$F_O \equiv F_C \wedge (\bigwedge_{y_i \in B_T} y_i) \wedge (\bigwedge_{y_j \in B_F} y_j) \wedge F_H \tag{5.16}$$

因为 R_H 是 F_H 的唯一可满足解，根据引理 5.1，则有:

$$F_H \wedge (\bigwedge_{y_i \in B_T} y_i) \wedge (\bigwedge_{y_j \in B_F} y_j) \equiv F_H \tag{5.17}$$

根据式 (5.16)，(5.17)，有:

$$F_O \equiv F_C \wedge F_H \tag{5.18}$$

由于 F_H 可满足，$V_{F_C} \cap V_{F_H} = \phi$，因此有：

$$F_O \equiv_{SSE} F_C \tag{5.19}$$

证明完成。 ■

5.4.1.2 有效性分析

有效性是指混淆算法对 CNF 公式形式上的改变，会使电路结构提取工作变得困难或根本不可实现。通过增加冗余的文字和子句，OBFUSCATOR 可以将 CNF 公式中的标记改变为另一合法标记。在混淆之后，原始的 CNF 公式就被转化为混有噪音电路的另一个公式。由于在外包求解时，部署于开放环境下的 SAT 求解

器，其输入为混淆后的的 CNF 公式而不是原始的 CNF 公式，原始 CNF 公式中的电路结构就不再直接暴露给潜在的攻击者。

本节定性分析混淆算法对有向超图的改变，而其对二分图的改变和超图类似，不再赘述。

图5.2a) 和5.2b) 给出了两个 AND2 门 a 和 e 的 CNF 标记，混淆后的标记显示在图5.2c) 和5.2d) 中。其中包括三种改变：

1. 关键子句 c_1 和 c_5 的长度由 3 变为 4。这就阻止了基于关键子句模式匹配的电路结构探测算法[12] 的攻击;

2. a 的特征子句 c_1-c_3) 以及 e 的特征子句 c_5-c_7 变成了完全不同的形式，并且有新的子句加入了公式，如 c_4 和 c_8。这就阻止了基于标记子图同构的电路结构检测算法[11] 的攻击;

3. 通过加入合适的新文字以及构造新的子句，门 a 的 CNF 标记从 AND2 变为了 AND3，如图5.2a) 和5.2c)。Husk 文字 A，变成了新生成的 AND3 门的一个输入，并且与原始 AND2 门的输入 b 和 c 不可区分。这也使得区分混淆后 AND2 和真实 AND3 变得不再可能。

5.4.1.3 算法复杂性分析

根据本章提出的基于混淆的 SAT 求解框架，SAT 问题求解开销包含私有云上的计算开销和公共云中的计算开销。在私有云中，计算开销包括 CNF 公式混淆开销和结果恢复开销；在公共云中的计算开销是指对混淆后 CNF 公式的 SAT 求解开销。由于 Husk 公式可以预先产生，因此在本文中，Husk 公式的产生开销将不再被列入到每一次 SAT 求解的开销中。

混淆算法的复杂性

算法5.4实现了混淆，其中主程序仅仅包含一层循环，但是其中一个子程序 **mark**(算法5.2) 包含了 4 层循环，由于两层内循环的上界为子句长度，因此混淆算法复杂性为 $O(n^2)$。

$$a=\wedge(b,c)\rightarrow\begin{Bmatrix}c1:(a\vee\neg b\vee\neg c)\\c2:(\neg a\vee b)\\c3:(\neg a\vee c)\end{Bmatrix}$$

a)AND2 gate a

$$a=\wedge(b,\ c,\ \mathrm{A})\rightarrow\begin{Bmatrix}c1:(a\vee\neg b\vee\neg c\vee\neg\mathrm{A})\\c2:(\neg a\vee b\vee\mathrm{B})\\c3:(\neg a\vee c\vee\mathrm{C})\\c4:(\neg a\vee\mathrm{A}\vee\mathrm{D})\end{Bmatrix}$$

c)AND2 gate a after obfuscation

$$e=\wedge(f,g)\rightarrow\begin{Bmatrix}c5:(e\vee\neg f\vee\neg g)\\c6:(\neg e\vee f)\\c7:(\neg e\vee g)\end{Bmatrix}$$

b)AND2 gate e

$$e=\wedge(f,\ g,\ \neg E)\rightarrow\begin{Bmatrix}c5:(e\vee\neg f\vee\neg g\vee\mathrm{E})\\c6:(\neg e\vee g\vee\mathrm{F})\\c7:(\neg e\vee f\vee\mathrm{G})\\c8:(\neg e\vee\neg\mathrm{E}\vee\mathrm{H})\end{Bmatrix}$$

d)AND2 gate e after obfuscation

图 5.2　混淆前后 a 和 e 的 CNF 标记

解恢复算法的复杂性

解恢复在算法5.5中实现，由于仅仅包含一层循环，算法复杂度为 $O(n)$。由于算法5.5为线性复杂度，带给整个求解过程的开销较小。

5.4.2 实验评估

5.4.2.1 实验设计

本文给出的算法由 C 语言实现。实验用机器的配置为 Intel Core(TM) i7-3667U CPU @ 2.00GHz，8GB RAM。

将 ISCAS89 测试集中的部分电路，按照形式化验证时的要求分别展开 20、30、100 次并编码为 CNF 公式。产生的 Husks 公式包含的节点数和子句数分别为 $vn = 675$ 和 $cn = 2309$。使用 MiniSat[25] 作为求解器，模拟开放环境下隐私保护 SAT 求解计算。

5.4.2.2 实验结果分析

表5.1给出了实验结果，表中各个参数的意义如下所示：

1. **变量数 / 子句数（vn/cn）**：原始 CNF 公式中包含的变量数和子句数。
2. **求解时间（Solving Time）**：混淆前后的 SAT 问题得到第一个可满足解的时间。
3. **混淆时间（Obfuscation Time）**：按照混淆算法在原公式中混合入 HUSK 公式的时间。
4. **解恢复时间（Map Time）**：从混淆后的解中恢复出实际解的时间。

实验证实了算法的正确性，对取自 ISCAS89 中的 13 个示例电路生成的可满足 CNF 公式，混淆前后的 SAT 求解的结果一致。

依据下式定义的非对称加速比（Asymmetric Speedup）[79]，对全部的电路，其非对称加速比均超过了 100%。特别是某些尺寸较大的路，非对称加速比超过 500 这表明了外包复杂 SAT 求解函数的必要性。

$$Asymmetric\ Speedup = \frac{Solving\ Time}{Obfuscation\ Times + Map\ Time} \tag{5.20}$$

实验也显示出混淆后带来的 SAT 求解的开销，不同的电路具有不同的表现。

$$Solving\ OverHead = \frac{Solving\ Time\ After\ Obfuscation - Solving\ Time\ Before\ Obfuscation}{Solving\ Time\ Before\ Obfuscation} \tag{5.21}$$

90% 以上的电路，开销小于 100%。仅有少部分电路，开销超过了 100%；这表明简单的混淆策略引入的求解开销是可以接受的。

表 5.1　不同类型电路 CNF 公式混淆前后的运行时间

电路	变量数	子句数	混淆时间（单位：毫秒）	映射时间（单位：毫秒）	异构加速比	求解时间（单位：毫秒）		
						混淆前	混淆后	时间开销
s3330	10275	31973	20.001	4.000	200.00%	48.003	72.004	50.00%
s3271	11897	43969	36.002	0.000	155.56%	56.003	52.003	-7.14%
s5378	12341	41930	28.001	8.000	111.11%	40.002	56.003	40.00%
s3384	12583	42146	28.001	0.000	757.16%	212.013	544.034	156.60%
s1196	13970	50011	40.002	8.000	100.00%	48.003	80.005	66.67%
s4863	17483	64153	44.002	12.000	242.86%	136.008	160.01	17.65%
s9234	18044	67500	52.003	8.000	226.67%	136.008	256.016	88.24%
s6669	23703	81746	40.002	4.000	1072.74%	472.029	888.055	88.14%
s13207	34423	117739	72.004	8.000	275.00%	220.013	288.018	30.91%
s15850	37692	133613	88.005	16.001	276.92%	288.018	240.015	-16.67%
s38417	102556	388438	220.013	28.001	1296.78%	3216.2	4060.25	26.24%
s38584	122341	446430	252.015	28.001	605.72%	1696.11	916.057	-45.99%
s35932	1164134	4271763	2328.145	280.017	27903.83%	727777	115339	-84.15%

5.5　本章小结

在本章中，我们对 SAT 求解的安全外包进行了建模，给出了实现 SAT 问题求解输入隐私保护的有效方案，并设计了结构感知的 CNF 混淆方法，以防止敏感的输入信息如电路结构被恢复。理论和实验结果都证明了方案的实用型和有效性。

第六章　解空间投影等价的混淆算法

6.1　引言

在第三章中，假设攻击者仅仅会从分析 CNF 公式入手，获取结构信息。针对上述攻击模式，提出了 CNF 公式结构隐私保护的的混淆算法。

而在本章中，将在前一章工作的基础上再进一步，假设攻击者已经确切知道公式中夹杂了噪音变量和子句，因而试图从分析解的角度出发，还原 CNF 公式，而后再获取其中携带的结构信息。

本章针对上述的隐私保护威胁，对“解空间投影等价的混淆算法”进行了深入的研究。通过引入具有簇形解的噪音 CNF 公式，使噪音变量的取值不再唯一，消除攻击者利用 ALLSAT 还原 CNF 公式的可能性，从而实现结构信息的隐私保护。

6.2　问题描述

第三章提出的 CNF 混淆算法，通过在原始 CNF 公式中混入仅有唯一解的噪音 CNF，在隐藏原有 CNF 携带的结构信息的同时，还保证混淆前后的解空间等价。混入唯一解的噪音 CNF，也就意味着在混淆后的解中，这些噪音变量都将只有唯一的取值。从解混淆攻击者的角度看，对混淆后 CNF 公式的攻击破解点之一，就是寻找出赋值唯一的噪音变量。一旦确认噪音变量，就可能将按照嵌入规则添加到原始子句中的额外文字取出，恢复原有子句，进而获得原始的 CNF 公式，并取得其中的结构信息。

在详细描述之前，先介绍与之相关的术语。

6.2.1　ALLSAT 求解

多数 SAT 问题具有一个以上的解，求解出 SAT 问题所有解的过程称为 ALLSAT 求解。直观上看，调用一次 SAT 求解器可以获得 SAT 的一个解。将已知解中每一个变量的赋值求反，以获得其反文字，并将所有这样的反文字的合取作为阻断（block）子句并加入到待求解的 SAT 问题公式中，以引导求解器避开已搜索过的解。通过多次重复，最终可以获得 SAT 问题所有的解。

本文中给出基于 Craig 插值[35, 159] 的求解算法。Craig 插值是一阶逻辑定理证明中的一个强大工具。为了简明起见，在这里将仅讨论命题逻辑中的 Craig 插值。Craig 插值的定义是，对于两个命题逻辑公式 F_1 和 F_2，假设他们的支撑集（输入变量集合）为 V_1 和 V_2，且有 $F_1 \wedge F_2$ 不可满足，则存在 F_3，使得 $F_1 \rightarrow F_3$ 成立，

$F_3 \wedge F_2$ 不可满足，且 F_3 的支撑集为 $V_1 \cap V_2$。很显然，F_3 是 F_1 的一个抽象，既包含了 F_1 的所有情形，又具有更少的支撑集变量。F_3 就是一个 Craig 插值。

使用 Craig 插值的最常用和最高效算法是 McMillian 算法[38]：首先构造两个相互矛盾的公式，然后使用 SAT 求解器得到他们的不可满足证明，然后使用定义3.2所描述的方法，从不可满足证明中抽取 Craig 插值。而在文献[36] 中，Craig 插值的产生过程类似于传统的可满足赋值遍历算法。不过其扩展算法包含两步，分别对应于两个参与计算的公式。该算法不需要产生不可满足证明的 Craig 插值算法。

6.2.2 基于 ALLSAT 求解和分区的攻击

根据前一章给出的 CNF 公式混淆算法，可知其中 Husk 变量的赋值是唯一的。根据 SAT 问题的特点可知，对混淆后的 CNF 公式进行 ALLSAT 求解，可以得到混淆后公式的全部解。因此，攻击者可以从中筛选出赋值唯一的变量。由于原始 CNF 中也可能存在其他赋值唯一的变量，例如用于属性验证时的属性变量；因此这些具有唯一赋值的变量可作为 Husk 变量的候选者；攻击者可以通过构建仅包含这些唯一赋值变量的分区[156]，来获取 Husk 公式，并识别出 Husk 变量，从而恢复出原始公式。

假设 Husk 变量的个数是 m，原公式中取唯一赋值的变量数是 n，完全找出 m 的概率为 $1/C_{m+n}^{m}$。由于原始公式中的 n 值一般为待验证属性变量，当 n 值较小时，解混淆攻击成功的概率将会变得很大。因此必须解决上述问题。

6.3 解空间投影等价的混淆算法

本章所给出的混淆算法，在设计时，考虑下面四个因素：

1. 可移植性：目前的 SAT 求解器集成了冲突检测[32] 等高效求解机制，因此希望可以将其作为黑盒直接使用，而不是像文献[49] 试图使用新的求解算法。

2. 隐形性[97]：混淆算法可以保证电路结构信息无法通过对 CNF 公式的分析来获取。

3. 适应性[97]：求解框架应能防止第三方通过 ALLSAT 来获取求解结果，进而识别出噪音变量。

4. 开销：混淆算法不应该引入太多的开销。

根据上述目标, 为了保持混淆后的 CNF 公式对求解计算的透明性，根据第5章中给出的 SSH 规则对原始 CNF 公式进行混淆, 在 CNF 公式的子句中加入新的文

字，并在公式中加入新的子句。新加入的文字和一部分新的子句来自于具有特殊解形式的可满足 CNF 公式，这个公式称为簇形 Husks 公式。SSH 规则保证原始的 CNF 公式可以和簇形 Husks 公式无缝地混合在一起，以便于达到2) 和 3) 的要求。SSH 规则在第5章中已经给出定义，出于本章叙述的完整性，在本章的小节*6.3.2.1*中仍旧会给出详细的定义。

定义 6.1 (**簇形解**): 假设 CNF 公式有 $m+n$ 个变量，且 m 和 n 均大于 0。公式有 k 个可满足解。其中 n 个变量的赋值在 k 个解中都相同，另外 m 个变量的赋值在 k 个解中不全相同。部分变量具有完全相同赋值的这 k 个解就构成一个簇形解。

定义 6.2 (**簇形 Husks 公式**): Husk 公式是有簇形可满足解的 CNF 公式，并且解变量的赋值是非特异的（不是全 0 或全 1）。

与第三章相同，基于混淆算法的 SAT 求解隐私保护框架包含 *GENERATOR*，*OBFUSCATOR*，*MAPPER* 和 *VERIFIER* 算法，除了 *GENERATOR*，其余算法和第三章相同。新的 *GENERATOR* 在6.3.1节介绍。

6.3.1 簇形 Husk 公式的产生

产生可满足 CNF 公式的算法有多种[157, 158]。本章中，定义6.2 中指出的簇形 Husks 公式采用质因数的方法构造[158]。如算法6.1所示。

1. **首先**, 给定两个质数 $p_A \neq p_B$(第 3 行)，且两者具有如下的关系：其二进制向量表示为 $p_A =< a_1, \ldots, a_{n+m} >$ 和 $p_B =< b_1, \ldots, b_{n+m} >$。将 $p_A * p_B$ 赋值给乘法器 M 的输出，并且限制 $I_1 \neq 1$ and $I_2 \neq 1$ (第4行)。其中，I_1 和 I_2 是 M 的输入。

2. **其次**，将乘法器 M 编码为 CNF 公式 $Tseitin(M)$(第5行)。

为了满足 $Tseitin(M)$，M 的两个输入一定是 $\{I_1 = p_A, I_2 = p_B\}$ 或者 $\{I_1 = p_B, I_2 = p_A\}$。根据条件两个解具有部分相同的赋值，即存在 n 个正整数下标

算法 6.1 *GENERATOR*

1: input : NULL
2: output : Husks CNF F_H and Husks result R_H
3: Generating prime numbers p_A and p_B ;
4: $\Phi = M(I_1 \neq 1, I_2 \neq 1, O = p_A * p_B)$;
5: $F_H = Tseitin(\Phi)$;
6: $R_H = \{a_{i_j} | 1 \leq i_j \leq n, a_{i_j} \equiv b_{i_j}\}$;

$\{i_j|1 \le j \le n\}$，其中每个都满足 $1 \le i_j \le n$，使得 $a_{i_j} \equiv b_{i_j}$。直观的，$\{i_j|1 \le j \le n\}$ 是具有相同取值的 a_{i_j} 和 b_{i_j} 的下标 i_j 的集合。令 $R_H = \{a_{i_j}|1 \le i_j \le n, a_{i_j} \equiv b_{i_j}\}$。$R_H$ 也称为该公式的簇形解公共赋值变量集。

6.3.2 解空间投影等价的混淆

为了防止 CNF 公式以及解的信息泄露，给出了一个隐私保护的策略。该策略基于下面的事实和期望:

事实 1： 改变公式中的 CNF 标记和关键子句，可以使基于模式匹配或子图同构的电路结构恢复技术失效。

事实 2： 混淆后的解空间不应该被缩小，否则会误导真实应用，例如验证等。

期望 1： 鉴于事实 2, 解空间可以被扩大，以便于误导公共云上包括 SAT 求解器在内的第三方。

期望 2： 可以通过投影的方式，从混淆后公式的解中较快的恢复出原始公式的解。

本文给出的隐私保护算法，称为 *OBFUSCATOR*。该算法使用解空间投影等价算法，将 Husks 公式 F_H 嵌入到原始公式 F_C 中，产生一个新的 CNF 公式 F_O。*OBFUSCATOR* 改变子句的文字集合以及公式中的子句集合，来防止公式中的电路结构被恢复。通过 SSH 规则混淆后，原公式解空间被成倍数扩展，以防止在公共云上的 SAT 求解器获取原始公式的解。其中倍数为簇形解公式的解个数，并且原始公式的解可以通过投影的方式从混淆后公式的解中获取。

6.3.2.1 解空间投影等价（SSPE）算法

让我们在第5章的基础之上，考虑一个更复杂一些的问题：将 SAT 问题编码为 CNF 公式外包到云中，由 SAT 求解器求解；并且不希望 SAT 求解器知道原始公式以及它的解的个数。

一个简单的方法是：将原始的 CNF 公式，和具有多个解，且和原始公式无共同变量的可满足公式混合在一起，并将混合后的 CNF 公式外包的云上。原始公式的解将会混杂在混合公式的解中，并且解的个数为两个 CNF 公式解的个数的乘积。但是，这种简单混合的可识别性很高，基于分区[156] 的方法可以将两个不相关的公式分离开来，从而得到原始的 CNF 公式。

一个改进的方法是：对任意公式 F_C，产生一个可满足公式 F_H 及其一个解 R_H。F_C 和 F_H 没有公共变量，也即：$V_{F_C} \cap V_{F_H} \equiv \phi$。我们将采用如第5章所给出的方法，将 F_C 和 F_H 无缝混合，以便于隐藏 F_C。同时，保持 F_C 中的所有解都包含在新的公式中。

为实现 F_C 和 F_H 无缝混合，直觉的方法是将 F_H 中的变量加入到 F_C 的子句中，并且使用 F_C 和 F_H 中的变量产生新的子句。由于 CNF 特性，导致对任何

CNF 公式 F_C，在其子句中加入新的文字可能会扩展解空间，而加入包含 F_C 中变量的子句则会缩减解空间。那我们如何在此情况下保证 F_C 所有的解仍旧保留在新的公式中，并且新公式的解空间在原始变量上的投影和原始解空间完全等价。在给出具体答案之前，首先澄清下面的概念。

定义 6.3 (**解包含关系** $S_C \subseteq S_O$): CNF 式 F_C 和 F_O 具有 n_{F_C} 个公共变量 $x_1, ..., x_{n_{F_C}}$。并且有 $|V_{F_C}| \equiv n_{F_C}$，$|V_{F_O}| \equiv n_{F_O}$，且 $n_{F_O} \geqslant n_{F_C} > 0$。$S_C$ 和 S_O 分别是 F_C 和 F_O 的解。在 S_C 和 S_O 中，n_{F_C} 个公共变量的赋值是相同的，即对任何 $1 \leq i \leq n_{F_C}$，有 $S_C(x_i) \equiv S_O(x_i)$。我们称解 S_C 包含于 S_O，记为 $S_C \subseteq S_O$。

定义 6.4 (**解空间投影等价** (SSPE)): CNF 公式 F_C 有 n 个解 $\{S_{C_1}, ..., S_{C_n}\}$。CNF 公式 F_O 有 $n * m$ 个解 $\{S_{O_1}, ..., S_{O_{n \cdot m}}\}$。对于任意 $1 \leq i \leq n$ 和 $0 \leq j \leq m-1$，有 $S_{C_i} \subseteq S_{O_{i+n*j}}$，则称 F_O 解空间投影等价于 F_C，记为 $F_C \equiv_{SSPE} F_O$。

为了在新公式中保持 F_C 公式中的所有解，仍然需要使用第5章提出的解空间保持 (SSH) 规则。与第5章不同的是，本章使用公式 F_H 和其簇形解的公共赋值变量集合 R_H 来混淆公式 F_C，以便于保证混淆后公式具有投影等价的解空间。

定义 6.5 (**Obf**(F_C, F_H, R_H)): 对任意公式 F_C，和可满足公式 F_H 及其一个部分赋值 R_H，$Obf(F_C, F_H, R_H)$ 是在基于 SSH 规则将 F_C 和 F_H 混合后得到的公式。如果 F_H 是簇形 Husks 公式，并且 R_H 是其簇形解公共赋值变量集合，就称 $Obf(F_C, F_H, R_H)$ 为 **SSPE** 混淆。

对 SSPE 混淆，下列定理成立。

定理 6.1 (SSPE Obfuscation): 对任意 CNF 公式 F_C，以及簇形 Husks 公式 F_H，如果 $V_{F_C} \cap V_{F_H} \equiv \phi$，并且 R_H 是 F_H 的一个簇形解的公共赋值变量集，并设

$$F_O = Obf(F_C, F_H, R_H) \tag{6.1}$$

则有

$$F_O \equiv F_C \wedge F_H \tag{6.2}$$

该定理的证明见小节6.4.1。

基于定理6.1和定义6.4，我们有:

定理 6.2: 对任意 CNF 公式 F_C, 和簇形 Husk 公式 F_H，如果 $V_{F_C} \cap V_{F_H} \equiv \phi$，并且 R_H 是 F_H 的簇形解的公共赋值变量，则 $Obf(F_C, F_H, R_H) \equiv_{SSPE} F_C$。

该定理的证明见小节6.4.1。

经过 SSPE 混淆生成的 CNF 公式 $F_O = Obf(F_C, F_H, R_H)$，包含了 F_C 和 F_H 中的所有变量。

F_O 的解可被划分为 F_C 和 F_H 的解，F_C 的解可以从 F_O 的解中通过投影获得。F_O 解的个数等于 F_C 解个数和 F_H 解个数的乘积。

综上所述，F_O 的解空间是 F_C 解空间的上估计。F_O 和 F_C 可以使用同样的 SAT 求解器求解。即使通过 ALLSAT 求解出所有 F_O 的解，也仅仅可以确定 R_H，而无法推断 F_H 中除 R_H 之外其他变量的赋值。这就使噪音变量的完全识别变得不可能。

6.3.2.2 *OBFUSCATOR* 算法

OBUFSCATOR 算法遵循上述 SSH 规则，并选取具有簇形解的 Husk 公式 F_H，混淆 CNF 公式 F_C，从而防止 F_C 携带的电路结构和它的真实解被潜在的攻击者获取。

为了达到上述目标，*OBFUSCATOR* 首先会在 CNF 公式中的特征子句里随机加入新的文字，并使用 Husk 公式 F_H 中的文字生成新的子句，以改变原有公式的标记。

基于 SSH 规则的 SPE 混淆过程逻辑步骤描述如下。

Procedure 6.3.2.2.

输入：公式 F_C(其中包含了**关键子句集合** F_{Ck} 和**非关键子句集合** F_{Cn})，Husks 公式 F_H 以及其解 R_H。

输出：混淆结果 F_O。

步骤 1: 对关键子句 $c \in F_{Ck}$，从 R_H 中取出文字 *lit*。根据 SSH 规则 1，将 ¬*lit* 加入到 c。按照该规则生成的子句集合记为 S_3 .

步骤 2: 根据 SSH 规则 2，使用 R_H 中文字 *lit* 和 c 中的输出变量，产生新的子句。新产生的子句集合记为 S_4.

步骤 3: 将 S_3，S_4，F_H 和 F_{Cn} 混合产生 F_O。

end Procedure.

6.3.3 解恢复和验证算法

在公共云上的求解器完成求解，并将 F_O 的解 S_O 返回给私有云。和混淆器相对应，在私有云中使用算法5.5中的 *MAPPER* 和 *VERIFIER* 将 F_C 的解从 S_O 中过滤出来。解的恢复和验证算法和第5章相同。

6.4 理论分析和证明

6.4.1 正确性证明

根据定理6.1，在 SSH 规则下，原始的 CNF 公式可以和 Husks 公式无缝混合，并会保持解空间的投影等价。本节证明这些定理。在证明之前，首先给出以下的引理。

引理 6.1 (Cubic Husks Equation(CHE)): 对簇形 Husk 公式 F_H，记 $|V_{F_H}| = n$。其公共赋值变量集合 $R_H = \{c_i | 1 \leqslant i \leqslant n_c\}$，其中 n_c 为簇形解中公共赋值变量的个数。它的全部 m 个解记为 $\{S_{H_l} | 1 \leqslant l \leqslant m \text{ 且 } m \geqslant 2\}$。其中每个解记为 $S_{H_l} = \{c_i = B_i, y_{l_i} = B_{l_i} | B_i, B_{l_i} \in \{T, F\}, 1 \leqslant i \leqslant n_c \lessdot l_i \leqslant n\}$。

根据公共赋值变量集合 R_H，令 $F_{RH} = (\bigwedge_{1 \leqslant i \leqslant n_c}^{B_i \equiv T} c_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n_c}^{B_j \equiv F} \neg c_j)$。

则对簇形 Husk 公式的任一解 S_{H_l}，令 $F_{slH} = F_{RH} \wedge (\bigwedge_{n_c \lessdot l_i \leqslant n}^{B_{l_i} \equiv T} y_{l_i}) \wedge (\bigwedge_{n_c \lessdot l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_{l_j})$。

并且令 $F_{sH} = \bigvee_{1 \leqslant l \leqslant m} F_{slH}$。

则有 $F_{sH} \equiv F_H$。

证明：1) 由于 F_{sH} 可满足，则必然存在可满足公式：

$$F_{slH} = F_{RH} \wedge (\bigwedge_{n_c \lessdot l_i \leqslant n}^{B_{l_i} \equiv T} y_{l_i}) \wedge (\bigwedge_{n_c \lessdot l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_{l_j}) \tag{6.3}$$

则有:

$$S_{H_l} = \{c_i = T, c_j = F, y_{l_i} = T, y_{l_j} = F | B_i \equiv T, B_j \equiv F, B_{l_i} \equiv T, B_{l_j} \equiv F, 1 \leqslant i, j \leqslant n_c, n_c \lessdot l_i, l_j \leqslant n, \} \tag{6.4}$$

令 B_i, B_{l_i} 和 B_j, B_{l_j} 分别替换 $c_i = T, y_{l_i} = T$ 中的 T 和 $c_j = F$、$y_{l_j} = F$ 中的 F, 则有：

$$S_{H_l} = \{(c_i = B_i, c_j = B_j, y_{l_i} = B_{l_i}, y_{l_j} = B_{l_j}) | B_i \equiv T, B_j \equiv F, 1 \leqslant i, j \leqslant n_c, B_{l_i} \equiv T, B_{l_j} \equiv F, n_c \leqslant l_i, l_j \leqslant n\} \tag{6.5}$$

下标 i 和 j 取值范围相同，统一用 i 表示。下标 l_i 和 l_j 取值范围相同，统一用 l_i 表示。则式6.5可以简化表示为：

$$S_{H_l} = \{c_i = B_i, y_{l_i} = B_{l_i} | B_i, B_{l_j} \in \{T, F\}, 1 \leqslant i \leqslant n_c \lessdot l_i \leqslant n\} \tag{6.6}$$

因为 S_{H_l} 是 F_H 的一个解，则有 $F_H(S_H/V_{F_H}) \equiv T$。则有:

$$F_{sH} \vdash F_H \tag{6.7}$$

2) 由于 F_H 可满足，必然存在 F_H 的解，如式 (6.8) 所示, 可以使 $F_H(S_{H_1}/V_{F_H})$ 为真。

$$S_{H_1} = \{c_i = B_i, y_{l_i} = B_{l_i} | B_i, B_{l_i} \in \{T, F\}, 1 \leqslant i \leqslant n_c \lessdot l_j \leqslant n\} \tag{6.8}$$

根据式 (6.8) 构造 F_{s1H}，则有:

$$F_{s1H} = (\bigwedge_{1 \leqslant i \leqslant n_c}^{B_i \equiv T} c_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n_c}^{B_j \equiv F} \neg c_j) \wedge (\bigwedge_{n_c \lessdot l_i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{n_c \lessdot l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_j) \equiv T \tag{6.9}$$

$$F_{sH} = F_{s1H} \vee (\bigvee_{2 \leqslant l \leqslant m} F_{slH}). \tag{6.10}$$

因为 $F_{s1H} \equiv T$, 根据式 (6.9) 和 (6.10), 则有:

$$F_{sH} \equiv T \tag{6.11}$$

$$F_H \vdash F_{sH} \tag{6.12}$$

根据公式 (6.7) 和 (6.12), 则有:

$$F_{sH} \equiv F_H \tag{6.13}$$

证明完成。 ■

接下来讨论基于簇形 Husk 公式的，可实现解空间投影等价的 SSPE 混淆。

Theorem 6.1 解空间投影等价 (SSPE) 混淆

对任意 CNF 公式 F_C 和有 n 个变量 m 个解的簇形 Husk 公式 F_{sH}，如果

- $V_{F_C} \cap V_{F_H} = \phi$。
- F_H 簇形解的公共赋值集合 R_H=$\{c_i = B_i | B_i \in \{T, F\}, 1 \leqslant i \leqslant n_c\}$。其中 n_c 满足 $n_c \leqslant n - 1$。
- $F_O = Obf(F_C, F_H, R_H)$.

则 $F_C \wedge F_H \equiv F_O$，且有 $F_C \equiv_{SSPE} F_O$。

证明：根据 **Procedure**6.3.2.2, 按下列步骤构造 F_O:

1. 令 $F_{Op} = F_C$。
   对 $y_i \in \{y_i | (y_i = B_i) \in R_H \parallel B_i \equiv T)\}$, 令

   $$F_{Op}=Obf(F_{Op}, B = y_i, y_i \equiv B_i)。$$

2. 令 $F_{On} = F_{Op}$。
   对 $y_j \in \{y_j | (y_j = B_j) \in R_H \parallel B_j \equiv F)\}$, 令

   $$F_{On}= Obf(F_{On}, B = \neg y_j, y_j \equiv B_j)。$$

3. $F_O = F_{On} \wedge F_H$。 ■

根据引理5.2、5.3以及步骤1)，有：

$$F_{Op} \equiv F_C \wedge (\bigwedge_{1 \leqslant i \leqslant n_c}^{B_i \equiv T} y_i) \tag{6.14}$$

根据引理5.2、5.4 以及步骤2)，有：

$$F_{On} \equiv F_{Op} \wedge (\bigwedge_{1 \leqslant j \leqslant n_c}^{B_j \equiv F} \neg y_j). \tag{6.15}$$

根据步骤3) 和式 (6.14) (6.15)，有：

$$F_O \equiv F_C \wedge (\bigwedge_{1 \leqslant i \leqslant n_c}^{B_i \equiv T} c_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n_c}^{B_j \equiv F} \neg c_j) \wedge F_H \tag{6.16}$$

因为 R_H 是 F_H 的簇形解的公共赋值变量集合，令

$$F_{RH} = (\bigwedge_{1 \leqslant i \leqslant n_c}^{B_i \equiv T} c_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n_c}^{B_j \equiv F} \neg c_j) \tag{6.17}$$

对簇形 Husk 公式的任一解 S_{H_l}，构造公式

$$F_{slH} = F_{RH} \wedge (\bigwedge_{n_c < l_i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{n_c < l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_j) \tag{6.18}$$

构造公式 $F_{SH} = \bigvee_{1 \leqslant l \leqslant m} F_{slH}$，则有

$$F_{SH} = \bigvee_{1 \leqslant l \leqslant m} (F_{RH} \wedge (\bigwedge_{n_c < l_i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{n_c < l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_j)) \tag{6.19}$$

根据引理6.1，有：

$$F_H = F_{SH} = \bigvee_{1 \leqslant l \leqslant m} (F_{RH} \wedge (\bigwedge_{n_c < l_i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{n_c < l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_j)) \tag{6.20}$$

由式6.20，以及分配律有：

$$F_H = F_{RH} \wedge (\bigvee_{1 \leqslant l \leqslant m} ((\bigwedge_{n_c < l_i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{n_c < l_j \leqslant n}^{B_{l_j} \equiv F} \neg y_j)) \tag{6.21}$$

根据式 (6.16)，(6.17) 有：

$$F_O \equiv F_C \wedge F_{RH} \wedge F_H \tag{6.22}$$

根据式 (6.22)，(6.21)，以及吸收律有：

$$F_O \equiv F_C \wedge F_H \tag{6.23}$$

由于 F_H 可满足，$V_{F_C} \cap V_{F_H} \equiv \phi$，并且 F_H 的解的个数 n 大于 1。根据定义6.4则有下式成立：

$$F_O \equiv_{SSPE} F_C \tag{6.24}$$

6.5 本章小结

本文给出了电路结构感知的 CNF 混淆算法，可防止在 SAT 问题外包计算时，CNF 公式中的电路结构以及解被窃取。理论分析表明，算法可以有效的改变结构，同时还将扩展 CNF 公式的解空间。

第七章　解空间上估计的混淆算法

第三、四章研究了 SAT 问题求解中 CNF 公式的隐私保护，也即 SAT 问题输入数据的保护。本章把保护的范围进一步扩展到输出数据，研究可同时保护 SAT 问题输入和输出数据隐私的方法。该方法基于解空间上估计的混淆算法和解恢复算法。通过该混淆算法，CNF 公式被变换为具有不同电路结构的新公式，并且该公式的解空间为原公式的解空间的上估计。原始公式的解可通过映射方法从混淆后公式的解中恢复。理论分析证明了算法的正确性和有效性。理论分析和实验表明混淆算法具有多项式复杂度，而解恢复算法仅线性复杂度；在 ISCAS89 测试集上的实验表明，该混淆算法引入了可接受的 SAT 求解开销。

7.1　引言

命题可满足[1]（简称 SAT）问题求解在软硬件验证[2, 3] 和密码学[4] 等领域得到广泛应用。近年来，软硬件规模的日益扩大，服务于软硬件验证和密码破解的 SAT 问题规模也随之急剧膨胀。另一方面，云计算、网格计算等依托开放环境的计算模式可以根据应用规模提供弹性的计算资源，将复杂的 SAT 问题外包到云或网格环境下成为一种有吸引的解决方案[5–7]。

但是，对安全的担忧阻碍了大多数用户将其关键应用部署到开放环境中运行。由于网格计算是由松散耦合的高端计算设施组成[5]，网格环境下恶意计算节点的威胁是可预见的 [8]；在云计算环境下，虽然云硬件平台提供商及其基础设施（虚拟层）是可被信赖，在其上运行的虚拟机却不总是可以信赖的。文献[9] 指出，著名的云计算提供商亚马逊的 EC2 受到了虚拟机影像滥用的困扰，被污染虚拟机映像会迅速扩散到整个社区；而文献[10] 则指出了处于同一台物理机器上的虚拟机之间攻击的可能性。

这些事实指出，外包到云计算或网格环境下的 SAT 问题，其输入和输出数据可能会被未授权的第三方访问。这些潜在的威胁者可能会从这些数据中获取有价值的信息。例如，来源于软硬件验证的 SAT 问题，可能遭受硬件结构信息泄露的问题。Roy[11] 和 Fu[12] 的工作指出了从 CNF 公式中抽取电路结构信息的可能性。而针对计算结果的完整性，Du[8] 归纳了开放计算环境下存在的三类威胁：懒惰计算者、奇货可居者、恶意欺骗者。懒惰计算者不给或是给出不完整的结果；恶意欺骗者给出错误结果，奇货可居的计算参与者会计算出正确结果但会因利益问题将结果泄露给第三方。结果验证技术 [7][30] 可以有效制止懒惰者和恶意欺骗者的行为，但对制止奇货可居者却并不奏效，Du[8] 指出将 SAT 问题求解由显式稀有事件（ORE）转化为隐式稀有事件（CRE）是解决输出数据隐私的最终之道。

为了解决上述问题，本章给出了一个在保持求解方法不变的前提下，保护SAT 问题外包过程中输入输出隐私方法。

7.2 问题描述

7.2.1 系统假设

本章以云计算环境为例来阐述 SAT 求解外包中可能遇到的安全威胁。就威胁模型而言，网格计算和云计算类似。

在我们的研究中, 存在两种类型的云：私有云和公共云。私有云是可信的但是计算以及存储能力受限，适合处理简单计算；公共云可以提供弹性的计算和存储资源，可以承担复杂计算任务。CNF 公式将在私有云中由网表产生，SAT 求解器部署在公共云，用于求解 CNF 公式，并将结果返回给私有云。

文献[11, 12, 154, 155] 研究了抽取和利用 CNF 公式中电路结构的方法。Roy 等人[11] 和 Fu 等人[12] 给出了基于子图同构以及模式匹配的电路结构抽取算法，并能最大限度的恢复出 CNF 公式中的电路结构。这些技术可能会被潜在的攻击者利用来抽取敏感的电路结构信息。正如文献[8] 指出，由于复杂 SAT 问题的解是稀有事件，且这些解具有很重要的实用价值，恶意的计算参与者可能会保留这些解，来获取经济利益。因此，同 CNF 公式一样，公式的解也应该被当做隐私加以保护。

在我们的研究工作中，假设公共云环境下有“诚实且好奇“的计算参与者。他们会尽力完成所有的计算任务以便获得 CNF 公式的解，但是它们也试图从 CNF 公式或解中获得他们需要的信息，例如电路结构和解。

7.3 系统设计

7.3.1 基于混淆的 SAT 求解框架

结合以上的系统假设，当我们设计基于云或是网格的 SAT 求解框架时，将考虑下面四个因素：

1. 可移植性：目前的 SAT 求解器集成了冲突检测等高效机制，因此我们希望可以将其作为黑盒直接使用，而不是像[49] 那样试图使用新的求解算法。
2. 隐形性[97]：求解框架应该可以保护 CNF 公式中的电路结构不会被获取。
3. 适应性[97]：求解框架应能防止包括运行于公共云上的求解器在内的第三方获取真实的求解结果。
4. 开销：框架不应该引入太多的开销。

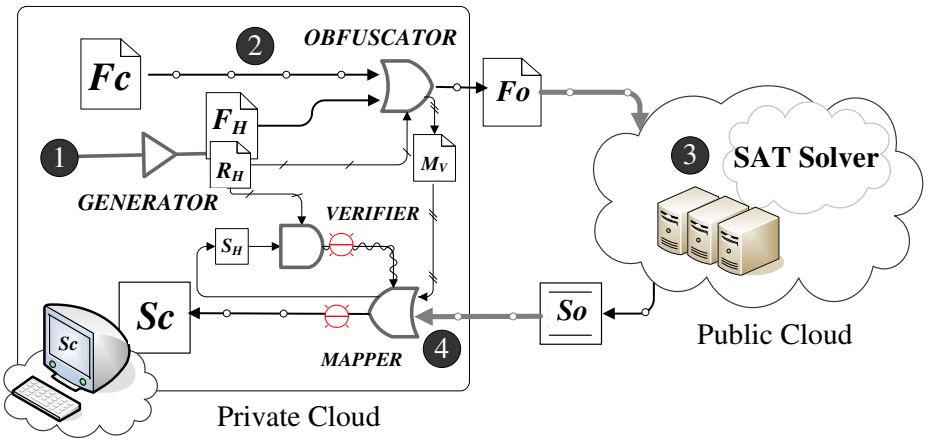


图 7.1　输入输出隐私保护的 SAT 求解框架

根据上述目标, 本章给出了一个输入输出隐私保护的 SAT 求解框架，该框架基于 CNF 公式混淆算法。为了混淆 CNF 公式，根据后续章节介绍的 SSH 规则和 CSA 策略, 在 CNF 公式的子句中加入新的文字，并在公式中加入新的子句。新加入的文字和一部分新的子句来自于另外一个 CNF 公式，称这个公式为 Husks 公式。SSH 规则和 CSA 策略保证原始的 CNF 公式可以和 Husk 公式无缝地混合在一起，以达到2) 和 3) 的要求。SSH 规则和 CSA 策略将在*7.3.3.4)* 和 *7.3.3.5)* 小节中介绍。

定义 7.1 (Husks **公式**): Husks 公式是包含一个以上解的可满足公式，并且解中变量的赋值是非特异的，也就是不是全 T 或全 F。

框架的详细实现在图 7.1中给出。在框架下，SAT 问题的求解包含了 4 步。

步骤 1，GENERATOR 产生一个 Husks 公式 F_H 和它的一个解 R_H。

步骤 2，OBFUSCATOR 混淆原始 CNF 公式 F_C，并产生一个新的 CNF 公式 F_O。

步骤 3，F_O 由位于公共云上的 SAT 求解器求解，并返回结果 S_O。

步骤 4，MAPPER 和 VERIFIER 从 S_O 中抽取出 S_C，并确认 S_C 是 F_C 的解。

步骤 3在公共云中运行，其余的步骤在可信的私有云上运行。GENERATOR，OBFUSCATOR，MAPPER 和 VERIFIER 算法将在 7.3.2，7.3.3 和7.3.4小节中分别介绍。

7.3.2　Husks **公式的产生**

产生可满足 CNF 公式的算法有多种[157, 158]。本文中，定义7.1中指出的 Husks 公式采用质因数的方法构造[158]。产生 Husks 公式的 GENERATOR 的实现在算法7.1中描述。**首先**, 给定两个质数 $p_A \neq p_B$(第 3行)，将 $p_A * p_B$ 赋值给乘法器 M 的输出，并且限制 $I_1 \neq 1$ and $I_2 \neq 1$ (第4行)。其中，I_1 和 I_2 是 M 的输入。

算法 7.1 GENERATOR

1: input : NULL;
2: output : Husks CNF F_H and Husks result R_H;
3: Generating prime numbers p_A and p_B ;
4: $\Phi = M(I_1 \neq 1, I_2 \neq 1, O = p_A * p_B)$;
5: $F_H = Tseitin(\Phi)$;
6: $R_H = p_A \mid p_B$;

第二, 将乘法器 M 编码为 CNF 公式 $Tseitin(M)$(第5行)。

为了满足 $Tseitin(M)$, M 的两个输入一定是 $\{I_1 = p_A, I_2 = p_B\}$ 或 $\{I_1 = p_B, I_2 = p_A\}$, 这就使得 $p_A|p_B$ 或 $p_B|p_A$ 为 $Tseitin(M)$ 的两个解。我们取其中一个为 R_H。

构造具有两个以上部分赋值解 (见定义6.2) 的 Husk 公式也可采用类似的方法。这里给出一个简单构造方法：以两个质数 P_B 和 P_C 构成的合数替换原有的一个输入 P_B 作为新输入，由构造过程可知，使 CNF 满足的输入可以是 ($I_1 = P_A$,$I_2 = P_BP_C$)、($I_1 = P_BP_C$,$I_2 = P_A$)、($I_1 = P_AP_B$,$I_2 = P_C$)、($I_1 = P_C$,$I_2 = P_AP_B$)、($I_1 = P_AP_C$,$I_2 = P_B$)、($I_1 = P_B$,$I_2 = P_AP_C$)，这六种组合就构成了该公式的六个解，不妨任取其中四个解，取解中赋值相同的变量集合作为 R_H；取另外两个解中赋值相同的变量集合作为 S_H；我们将得到两个部分赋值解，且两个部分赋值解所包含的解个数比例为 2：1。使用类似的方法，通过选择合适的合数作为输入并且选择适当的解组合为部分赋值解，我们可以构造解空间不同的 Husks 公式。

7.3.3 解空间上估计的混淆

7.3.3.1 CNF 公式中的结构

在设计可以防止电路结构信息泄露的混淆算法之前，我们首先开讨论如何从 CNF 公式中获取电路结构信息。文献[11, 12] 给出了从 CNF 公式中获取电路结构信息的算法细节，在介绍这些算法之前，首先了解算法中用到的概念。

定义 7.2 (CNF **标记**): 门 g 的 CNF 标记就是它的 Tseitin 编码 $Tseitin(g)$。CNF 标记中的每个子句称为门的特征子句。包含门中所有变量的特征子句称为关键子句。对应于门输出的变量称为输出变量。

公式 (1.2) 中的 AND2 门，$\neg e \vee c$ 是它的一个特征子句，$e \vee \neg c \vee \neg d$ 是它的关键子句。e 是输出变量。

文献[11] 指出，具有相同特征函数的门在同一种规则下都将被编码为相同的 CNF 标记。

算法 7.2 OBFUSCATOR

```
input : The original CNF F_C, Husks CNF F_H, Husks result R_H
output : The obfuscated CNF F_O, variable mapping M
mark(F_C);
for c ∈ F_C do
    if c ∈ Key Clause Set then
        lit =get literal ∈ R_H;
        c = c ∪ ¬lit;
        nc = generate_new_clause(c, lit);
        F_C = F_C ∪ nc;
    end if
end for
for c ∈ F_C do
    averagelen = (σ_{c'∈F_C} |c'|) / |F_C| ;
    while |c| < averagelen do
        lit =get literal ∈ R_H ;
        while ¬lit ∈ c do
            lit=get literal ∈ R_H ;
        end while
        c = c ∪ ¬lit;
    end while
    M =remap all variable in F_C ∪ F_H ;
    F_O =reorder all clause in F_C ∪ F_H ;
end for
```

基于上述概念, Roy 等人.[11] 首先将 CNF 公式转化超图 G, 而后在图中匹配 CNF 标记，通过同构子图的方式来恢复出 CNF 公式携带的门信息，最后，创建出最大无关集来表示恢复出来的电路信息。

Fu 等人[12] 给出了一种改进方法，基于关键子句和 CNF 标记的模式匹配检测出所有门，并构建最大匹配门的子集。潜在的攻击者可以利用这些知识恢复出电路结构。因此，CNF 标记和关键子句是需要保护的重要信息。

7.3.3.2 SAT 问题解的类型

根据文献[160]，对外包计算问题的而言，解的类型可以分为显式稀有事件和隐式稀有事件。显式稀有事件 (Obvious Rare Events (ORE))：对某一些计算，判断解是真实解的标准是显而易见的，即给定 x 和 f(x)，计算者很容易确定 x 是否为 f(x) 的解。SAT 问题正是这样的一类问题，给定一个 CNF 公式的解，我们可以在线性时间内判断出它是否是真实的解。实际上，任何 NP 复杂性问题均存在有效的验证算法。隐式稀有事件 (Camouflaged Rare Events (CRE))：对某一些计算，仅

算法 7.3 mark and **generate_new_clause**

```
1: mark;
2: input : CNF formula S;
3: output : marked S ;
4: for (C ∈ S) & (|C| ≡ 3) do
5:     for l ∈ C do
6:         for (C1 ∈ S) & (¬l ∈ C1) & (|C1| ≡ 2) do
7:             for l1 ∈ C1 do
8:                 if (¬l1 ∈ C) & (l1 ≠ l) then
9:                     match + + ;
10:                end if
11:            end for
12:        end for
13:    end for
14: end for
15: if match ≡ 2 then
16:    mark l as output literal ;
17:    mark C as Key Clause;
18: end if
19: generate_new_clause;
20: input : key clause C in AND2, Husk literal lit;
21: output : new clause C1;
22: olit=Getting output literal from C ;
23: C1 = lit ∪ ¬olit ;
```

给出 x 和 f(x)，并不能判断出 x 是不是稀有事件，例如寻找单向哈希值 y 的逆，为了找到产生 y=h(x) 的 x，其中 h 是单向函数，每个参与者都得到一个 y 值。由于提交给计算参与者的 y 值不一定是有效值，即使参与者找到相应 x 值，依然无法确定是否得到正确值。

为了防止奇货可居的计算参与者，隐藏解的真实标准是非常必要的，显然相对于显式稀有事件，隐式稀有事件的隐藏更加容易一些。因为显式稀有事件的标准是众所周知的，而隐式稀有事件的标准可以由任务的派发者确定。因此，将 SAT 问题由显式稀有事件转化为隐式稀有事件是防范奇货可居者的重要手段。

7.3.3.3 输入输出隐私保护策略

为了防止 CNF 公式以及解的信息泄露，给出了一个隐私保护的策略，该策略基于下面的事实和期望：

事实 1: 改变公式中的 CNF 标记和关键子句可以使基于模式匹配或子图同构的电路结构恢复技术失效。

事实 2: 混淆后的解空间不应该被缩小，否则会误导真实应用，例如验证等。

期望 1: 鉴于事实 2, 解空间应该被扩大，以便于误导公共云上包括 SAT 求解器在内的第三方。

期望 2: 可以从混淆后的解中较快的恢复出原公式的解。

本文给出的隐私保护策略，称为 OBFUSCATOR，会根据电路结构感知（CSA）策略和解空间保持（SSH) 规则，将 Husks 公式 F_H 嵌入到原始公式 F_C 中，产生一个新的 CNF 公式 F_O。通过 CSA 策略,OBFUSCATOR 改变子句的文字集合以及公式中的子句集合，来防止公式中的电路结构被恢复。通过 SSH 规则，混淆后的解空间是原公式解空间的上估计，以便于在公共云上的 SAT 求解器都无法获取真实的解。

我们将在*7.3.3.4*) 和 *7.3.3.5*) 分别介绍 SSH 规则和 CSA 策略。

7.3.3.4 解空间保持（SSH）规则

让我们来考虑一个有趣的问题，我们将 SAT 问题编码为 CNF 公式外包到云中，由 SAT 求解器求解；并且不希望 SAT 求解器知道实际的 SAT 问题以及它的解。一个简单地方法就是将真实的 CNF 公式和一个可满足公式混合在一起，并将混合后的 CNF 公式外包的云上，真实公式的解将会混杂在混合后的解中。但是，基于分区[156] 的方法可以将两个不相关的公式分离开来，从而得到真实的 CNF 公式。

让我们来考虑一个改进的方法: 任意公式 F_C，和一个可满足公式 F_H 及其一个解$\mathbf{R_H}$, F_C 和 F_H 没有公共变量, 也即：$V_{F_C} \cap V_{F_H} = \phi$。我们将 F_C 和 F_H 无缝混合，以便于隐藏 F_C。同时，保持 F_C 中的所有解都包含在新的公式中。

为实现 F_C 和 F_H 无缝混合, 直觉的方法是将 F_H 中的变量加入到 F_C 的子句中, 并且使用 F_C 和 F_H 中的变量产生新的子句。由于 CNF 特性, 对任何 CNF 公式 F_C, 在子句中加入新的文字可能会扩展解空间，而加入包含 F_C 中变量的子句则会缩减解空间。那我们如何在此情况下保证 F_C 所有的解仍旧保留在新的公式中? 在给出具体答案之前，首先澄清下面的概念。

定义 7.3 (**解包含关系** $S_C \subseteq S_O$): CNF 式 F_C 和 F_O 具有 n_{F_C} 公共变量 $x_1, ..., x_{n_{F_C}}$ 并且有 $|V_{F_C}| \equiv n_{F_C}, |V_{F_O}| \equiv n_{F_O}, n_{F_O} \geqslant n_{F_C} > 0$。$S_C$ 和 S_O 分别是 F_C 和 F_O 的解, 在 S_C 和 S_O 中，n_{F_C} 个公共变量的赋值是相同的, 也就是 $S_C = \{x_1 = B_1, ..., x_{n_{F_C}} = B_{n_{F_C}} | B_i \in \{T, F\},\ 1 \leqslant i \leqslant n_{F_C}\}$, $S_O = \{x_1 = B_1, ..., x_{n_{F_C}} = B_{n_{F_C}}, ..., x_{n_{F_O}} = B_{n_{F_O}} | B_i \in \{T, F\},\ 1 \leqslant i \leqslant n_{F_O}\}$。我们称解 S_C 包含于 S_O, 记为 $S_C \subseteq S_O$。

定义 7.4 (**解空间等价** (SSE)): CNF 公式 F_C 有 n 个解 $\{S_{C_1}, ..., S_{C_n}\}$; CNF 公式 F_O 也有 n 个解 $\{S_{O_1}, ..., S_{O_n}\}$, 并且对于任意 $i \in [1, n]$, $S_{C_i} \subseteq S_{O_i}$。我们称 F_O 解空间等价于 F_C, 记为 $F_C \equiv_{SSE} F_O$。

定义 7.5 (**解空间上估计** (SSO)): CNF 公式 F_C 有 n 个解 $\{S_{C_1},...,S_{C_n}\}$; CNF 公式 F_O, 有 m 个解, $\{S_{O_1},...,S_{O_n},...,S_{O_m}\}$, 并且 $m \geqslant n$, 并且对于任意 $i \in [1,n]$, $S_{C_i} \subseteq S_{O_i}$。我们称 F_O 的解空间是 F_C 的上估计, 记为 $F_C \vdash_{sso} F_O$。

为了在新公式中保持 F_C 公式中的所有解, 仍然需要使用第5章提出的解空间保持 (SSH) 规则，使用公式 F_H 和它的解 R_H 来混淆公式 F_C，以便于保证混淆后公式具有上估计的解空间。

为叙述的完整性，我们在本节中再次给出:

解空间保持（SSH）规则:

1. **规则 1**: 对任一子句 $c \in F_C$, 从 R_H 中任取出变量, 并按照下列规则插入到子句 c: 如果在 R_H 中变量的赋值是 T, 作为负文字; 如果在 R_H 中变量的赋值是 F, 作为正文字; 用新生成的子句代替原始子句 c。

2. **规则 2**: 使用 R_H 中的文字和 F_C 中的变量创建新的子句，按照下列规则: 如果在 R_H 中文字是 T，就作为正文字; 如果在 R_H 中文字是 F，就作为负文字。

定义 7.6 (**Obf**(F_C, F_H, R_H)): 对任意公式 F_C, 和可满足公式 F_H 及其一个赋值 R_H,

$Obf(F_C, F_H, R_H)$ 是在基于 SSH 规则将 F_C 和 F_H 混合后得到的公式。

如果 F_H 是 Husks 公式并且 R_H 是它其中一个解, 就称 $Obf(F_C, F_H, R_H)$ 为 **SSO obfuscation**。

对基于 SSH 混淆，下列定理成立。

定理 7.1 (SSO Obfuscation): 对任意 CNF 公式 F_C, 以及 Husks 公式 F_H, 如果 $V_{F_C} \cap V_{F_H} = \phi$, 并且 R_H 是 F_H 的一个解。

则 $F_C \vdash_{sso} Obf(F_C, F_H, R_H)$。

定理7.1的证明将在7.4.1小节给出。

经过 SSO 混淆生成的 CNF 公式 $F_O = Obf(F_C, F_H, R_H)$，包含了 F_C 和 F_H 中的所有变量。

如果 F_H 中的变量被赋值为 R_H, 则有:

$F_O(R_H/V_{F_O}) = Obf(F_C, F_H(R_H/V_{F_H}), R_H)$

根据引理7.17.4.1小节）, $F_H(R_H/V_{F_H})$ 可以被表示为一个单一 Husk 公式，有唯一解 R_H。根据第5章中的定理推论5.2, 对任意 F_C 和混淆后公式 F_O, 则有:

1. F_C 是不可满足的当且仅当 F_O 不可满足。并且 F_C 的不可满足核可以通过从 F_O 的不可满足核中删除 F_H 中的文字获得。

2. F_C 可满足当且仅当 F_O 是可满足的。并且 F_C 的解可以通过将 F_O 的解投影到 F_C 的变量集中获得。

因此 F_O 的解可以被划分为 F_C 和 F_H 的解。F_C 的解可以从 F_O 的解中通过投影获得。

如果 F_H 中的变量被赋值为 R_H 之外的解,$(S_H \neq R_H)$, 则有:

$F_O(S_H/V_{F_O}) = Obf(F_C, F_H(S_H/V_{F_H}), R_H)$。

由于 R_H 不是 $F_H(S_H/V_{F_H})$ 的解, 混淆可能会扩展 F_C 的解, 也就是: 如果 F_O 可满足, 投影得到的 F_C 可能是假解。在解恢复阶段，我们通过在投影时限制 F_H 必须为 R_H 来排除此类情况。

综上所述，F_O 的解空间是 F_C 解空间的上估计。F_O 和 F_C 可以使用同样的 SAT solver 求解, 但是在不知晓 R_H 的情况下，F_C 无法从 F_O 获取。

从上面的分析可以看出，混淆后的解空间和 Husks 公式的解空间具有密切关系，通过使用不同 Husks 公式产生方法，我们可以定制解空间各异的 Husks，我们控制部分赋值解 R_H 和非 R_H 中解的个数，就可以调制出噪音解比例不同的混淆公式。

7.3.3.5 电路结构感知（CSA）策略

通过 SSH 规则, OBFUSCATOR 可以将在子句中添加新文字并且创建新的子句, 同时还保证了解空间不被缩减。为了防止电路结构被恢复，在哪种子句中加入文字，创建何种类型的新子句，仍然是悬而未决的问题。

由于门是电路的基本构件，并且*7.3.3.1)*小节指出 CNF 标记和关键子句是检测电路结构的关键。我们试图通过增加变量和子句来改变 CNF 的标记。为了误导潜在的攻击值，新加入得文字和子句还应该和原有的子句构成新的合法标记，以便于将原始的公式无缝隐藏。

以 AND2 为例，图7.2a) 是 AND2 门 a 的 CNF 标记。将 A 添加到关键子句 c_1 中，并且使用 A 和 a 产生子句 c_4, 将门 a 从 AND2 转变为 AND3, 加入了一个输入变量 A, 并且该变量和原始输入变量 b 和 c 不可区分。OR, NAND, NOR 门和 AND 门类似，均可以进行此类变换。

7.3.3.6 OBFUSCATOR 算法

OBUFSCATOR 算法遵循上述 SSH 规则和 CSA 策略，混淆 CNF 公式 F_C，从而防止 F_C 的携带的电路结构和它的真实解被潜在的攻击者获取。

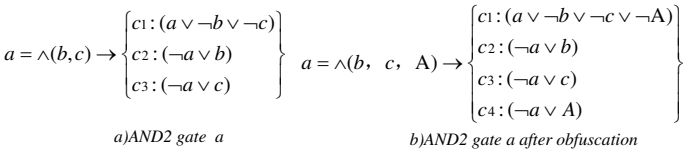


图 7.2　将 AND2 门改变为 AND3 门.

为了达到上述目标，OBFUSCATOR 首先会检测 CNF 公式中的门, 而后将这些门转变为不同标记的门. OBFUSCATOR 的详细实现在算法7.2中, 其中使用 *mark*(第 3 行) 来检测 CNF 公式中的关键子句和输出变量，并使用 *generate_new_clause*(第 8 行) 产生新的子句。由于 AND2 是最常见的门, 我们在仅仅给出 AND2 门的 **mark** 算法，同样, 我们也仅仅给出 AND2 的 **generate_new_clause** 算法，这两个算法组合起来可以将 AND2 的标记转换为 AND3 的标记. 具体的实现在算法7.3中。

算法7.2和7.3中实现的基于 SSH 和 CSA 混淆过程描述如下:

Procedure 7.3.3.6.

输入：公式 F_C, Husks 公式 F_H, 解 R_H.

输出: Formula F_O.

根据算法7.2, F_C 包含了**关键子句** (第5行) 和**非关键子句**, 相应的子句集合表示为F_{Ck} 和F_{Cn}.

步骤 1: 对关键子句 $c \in F_{Ck}$, 从 R_H 中取出文字 *lit*, 根据 SSH 规则 1 将 ¬*lit* 加入到 *c*(算法7.2的7, 19行). 生成子句的集合记为 S_3 .

步骤 2: 根据 SSH 规则 2，使用 R_H 中文字 lit 和 *c* 中的输出变量，产生新的子句 (算法 7.2的8行, 算法7.3 的23行)。新产生的子句集合记为 S_4.

步骤 3: 将 S_3, S_4, F_H, 和 F_{Cn}, 混合产生 F_O (算法7.2的7, 19, 9 22 行).

end Procedure.

7.3.4　真实解的恢复

在公共云上的求解器完成求解并给出 F_O 解 S_O，并返回给私有云中。和混淆器相对应，在私有云中使用 MAPPER 和 VERIFIER 将 F_C 的解从 S_O 中过滤出来. MAPPER 和 VERIFIER 的实现在算法7.5中.

根据定理7.1, 如果结果是 UNSAT, 那么原始的 CNF 公式也是 (第3行). 如果结果是 SAT, MAPPER(8-10行) 将解投影到 F_C 和 F_H 的变量上, 已获得 S_C 和 S_H，分别作为 F_C 和 F_H 的候选解。VERIFIER (12-17行) 检测 S_H 是否等于 R_H, 如果等于,

算法 7.5 MAPPER and VERIFIER

```
1:  input : Obfuscated result S_O, variable mapping table M, Husk result R_H;
2:  output : Result S_C;
3:  if S_O is UNSAT then
4:      return UNSAT ;
5:  end if
6:  for lit ∈ S_O do
7:      var = abs(lit);
8:      rvar = M[var].variable;
9:      if M[var].formula is F_C then
10:         S_C[rvar] = lit > 0?rvar : ¬rvar ;
11:     else
12:         Hlit = lit > 0?rvar : ¬rvar;
13:         if R_H[rvar] ≠ Hlit then
14:             alert("Get another Solution from SAT Solver");
15:             break;
16:         end if
17:     end if
18: end for
19: print " SAT solution is S_C ";
```

S_C 就是 F_C 的真实解. 否则, S_C 可能是假解, 此时, 需要从 SAT 求解器获得一个新的解 (第14行).

解的投影过程依赖于变量映射表 M，它由 OBFUSCATOR(算法7.2的21行) 创建. $M[var].variable$ (第8行) 表示了 var 的原始变量名, $M[var].formula$ (第9行) 表示 var 所属于的公式，可以是 F_C 或 F_H。

7.4 理论分析

7.4.1 正确性证明

根据定理7.1，在 SSH 规则下，原始的 CNF 公式可以和 Husks 公式无缝混合, 而不会削减解空间. 本节中，我们证明这些定理。首先给出以下的引理。

引理 7.1 (Husks Equation(HE)): 对 Husks 公式 F_H 且 $|V_{F_H}| = n$, 它的全部 m 解 $\{S_{H_l}|1 \leqslant l \leqslant m\}$, 且解 $S_{H_l} = \{y_k = B_{l_k}|B_{l_k} \in \{T, F\}, 1 \leqslant k \leqslant n\}$.

对每个 S_{H_l}, 令 $F_{slH} = (\bigwedge_{1\leqslant i\leqslant n}^{B_{l_i}\equiv T} y_i) \wedge (\bigwedge_{1\leqslant j\leqslant n}^{B_{l_j}\equiv F} \neg y_j)$,

并且令 $F_{sH} = \bigvee_{1\leqslant l\leqslant m} F_{slH}$,

则有 $F_{sH} \equiv F_H$.

证明：1) 由于 $F_{sH} \equiv T$ 则必然存在

$$F_{slH} = (\bigwedge_{1 \leqslant i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n}^{B_{l_j} \equiv F} \neg y_j) \equiv T \tag{7.1}$$

则有:

$$S_{H_l} = \{y_i = T, y_j = F | B_{l_i} \equiv T, B_{l_j} \equiv F, 1 \leqslant i, j \leqslant n\} \tag{7.2}$$

令 B_i, B_j 分别替换 $y_i = T$ 中的 T 和 $y_j = F$ 中的 F, 则有：

$$S_{H_l} = \{(y_i = B_{l_i}, y_j = B_{l_j}) | B_{l_i} \equiv T, B_{l_j} \equiv F, 1 \leqslant i, j \leqslant n\} \tag{7.3}$$

因为 S_{H_l} 是 F_H 的一个解，则有 $F_H(S_H/V_{F_H}) \equiv T$。则有:

$$F_{sH} \vdash F_H \tag{7.4}$$

2) 由于 $F_H \equiv T$, 必然存在 F_H 的解，如式 (7.5) 所示, 可以使 $F_H(S_{H_1}/V_{F_H})$ 为真。

$$S_{H_1} = \{y_k = B_{1_k} | B_{1_k} \in \{T, F\}, 1 \leqslant k \leqslant n\}. \tag{7.5}$$

根据式 (7.5) 构造 F_{s1H}，则有式 (7.7):

$$F_{s1H} = (\bigwedge_{1 \leqslant i \leqslant n}^{B_{1_i} \equiv T} y_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n}^{B_{1_j} \equiv F} \neg y_j) \tag{7.6}$$

$$F_{sH} = F_{s1H} \vee (\bigvee_{2 \leqslant l \leqslant m} F_{slH}). \tag{7.7}$$

因为 $F_{s1H} \equiv T$, 并且有式 (7.6) 和 (7.7), 则有:

$$F_{sH} \equiv T \tag{7.8}$$

$$F_H \vdash F_{sH} \tag{7.9}$$

根据式 (7.4) 和 (7.9), 则有:

$$F_{sH} \equiv F_H \tag{7.10}$$

根据引理7.1，一个 Husks 公式等价于解子句的析取，其中每个解子句是该解中所有文字的合取。

引理 7.2 (OR Hold Obfuscation): 对公式 F_C 和 $F_{_R}H \vee F_{_S}H$, 及 $F_{_R}H \vee F_{_S}H$ 的一个赋值 R_H 有,

$$Obf(F_C, F_{_R}H \vee F_{_S}H, R_H) \equiv Obf(F_C, F_{_R}H, R_H) \vee Obf(F_C, F_{_S}H, R_H)$$

证明: 假设：

- F_C=$F_{Ck} \wedge F_{Cn}$, F_{Ck}= $\bigwedge_1^m(a_i \vee X_i)$.
- F_H=$F_{_R}H \vee F_{_S}H$
- R_H=$\{y_j = B_j | B_j \in \{T, F\}, 1 \leqslant j \leqslant n\}$.
- 令 $F_O = Obf(F_C, F_H, R_H)$

根据 **Procedure**7.3.3.6, 按下列 3 个步骤构造 F_O.

1. $(y_j \equiv B_j) \in R_H$, $(a_i \vee X_i) \in F_{Ck}$ 和规则 1:

   如果 $B_j \equiv T$, 则子句 $C_{ij} = (a_i \vee X_i) \wedge \neg y_j$.

   如果 $B_j \equiv F$, 则子句 $C_{ij} = (a_i \vee X_i) \wedge y_j$.

   令 $S_3 = \bigwedge_{1 \leqslant i \leqslant m}^{1 \leqslant j \leqslant n} C_{ij}$

2. $(y_j \equiv B_j) \in R_H$, $(a_i \vee X_i) \in F_{Ck}$ 以及规则 2:

   如果 $B_j \equiv T$, 则子句 $D_{ij} = \neg a_i \wedge y_j$.

   如果 $B_j \equiv F$, 则子句 $D_{ij} = \neg a_i \wedge \neg y_j$.

   令 $S_4 = \bigwedge_{1 \leqslant i \leqslant m}^{1 \leqslant j \leqslant n} D_{ij}$.

3. 令 $F_{_d}C = S_3 \wedge S_4 \wedge F_{Cn}$, then $F_O = F_H \wedge F_{_d}C$. ■

根据步骤3):

$$\begin{aligned} F_O = F_H \wedge F_{_d}C \qquad & F_H = F_{_R}H \vee F_{_S}H \models \\ F_O = (F_{_R}H \vee F_{_S}H) \wedge F_{_d}C \qquad & \models \\ F_O = (F_{_R}H \wedge F_{_d}C) \vee (F_{_S}H \wedge F_{_d}C) \qquad & \end{aligned} \tag{7.11}$$

根据 **Procedure**7.3.3.6, $F_{_d}C$ 仅和 F_C 以及 R_H 相关, 则

$$F_{_S}H \wedge F_{_d}C \equiv Obf(F_C, F_{_S}H, R_H) \tag{7.12}$$

$$F_{RH} \wedge F_{dC} \equiv Obf(F_C, F_{RH}, R_H) \tag{7.13}$$

根据式 (7.11), (7.12), (7.13), 有:

$$F_O \equiv Obf(F_C, F_{RH}, R_H) \vee Obf(F_C, F_{SH}, R_H) \tag{7.14}$$

引理 7.3 (AND Hold Obfuscation): 公式 F_C 和 $F_{RH} \wedge F_{SH}$, 并且 R_H 是 $F_{RH} \wedge F_{SH}$ 的一个赋值, 则有

$$Obf(F_C, F_{RH} \wedge F_{SH}, R_H)$$
$$\equiv Obf(F_C, F_{RH}, R_H) \wedge Obf(F_C, F_{SH}, R_H)$$

证明: 假设

- 子句 $A = a \vee X$ 和 $B = b$, 并且 $b \notin X$, 任意公式 F_{Cn}, F_{SH}
- 令 $F_{Ck} = A, F_C = F_{Ck} \wedge F_{Cn}$,
  $F_{RH} = B, F_H = F_{RH} \wedge F_{SH}$, 当 $R_H = \{b \equiv T\}$;
- 令 $F_O = Obf(F_C, F_H, R_H)$

根据 **Procedure** 7.3.3.6, 按下列 3 个步骤构造 F_O。

1. 根据 R_H 和规则 1, 构造子句 $C = A \vee \neg b$, 令子句集合 $S_3 = C$
2. 根据 R_H 和规则 2, 对于文字 $a \in A$, 构造子句 $D = \neg a \vee b$; 令子句集合 $S_4 = D$;
3. 令 $F_{dC} = S_3 \wedge S_4 \wedge F_{Cn}$.
   则有 $F_O = F_H \wedge F_{dC}$. ■

$$\begin{aligned} F_O = F_H \wedge F_{dC} \qquad & F_H = F_{RH} \wedge F_{SH} \models \\ F_O = (F_{RH} \wedge F_{SH}) \wedge F_{dC} \qquad & \models \\ F_O = (F_{RH} \wedge F_{dC}) \wedge (F_{SH} \wedge F_{dC}) \qquad & \models \end{aligned} \tag{7.15}$$

根据 **Procedure** 7.3.3.6, F_{dC} 仅仅和 F_C、R_H 相关, 则

$$F_{SH} \wedge F_{dC} \equiv Obf(F_C, F_{SH}, R_H) \tag{7.16}$$

$$F_{RH} \wedge F_{dC} \equiv Obf(F_C, F_{RH}, R_H) \tag{7.17}$$

根据式7.15)、7.16）和7.17），则有

$$F_O = Obf(F_C, F_{_RH}, R_H) \wedge Obf(F_C, F_{_SH}, R_H) \tag{7.18}$$

根据引理7.2和7.3, 对 Husks 公式 F_H, 公式的与和非关系在混淆后依然保持。

接下来我们来讨论基于 Husks 公式的解空间上估计的 SSO 混淆。

Theorem 7.1 解空间上估计（SSO) Obfuscation

对于 CNF 公式 F_C, Husks 公式 F_H, 如果

1. $V_{F_C} \cap V_{F_H} = \phi$, R_H 是 F_H 的 m 个解之一.

2. $F_O = Obf(F_C, F_H, R_H)$.

则 $F_C \vdash_{sso} F_O$.

证明：假设 F_H 的一个解 R_H=$\{y_i = B_{R_k} | B_{R_k} \in \{T, F\}, 1 \leqslant k \leqslant n\}$, 它其余的 $m-1$ 个解 $\{S_{H_l} | 1 \leqslant l \leqslant m-1\}$. $S_{H_l} = \{y_k = B_{l_k} | B_{l_k} \in \{T, F\},\ 1 \leqslant k \leqslant n\}$.

根据 R_H 和 S_H，我们定义 $F_{_RH}$ 和 $F_{_SH}$:

$$F_{_RH} = (\bigwedge_{1 \leqslant i \leqslant n}^{B_{R_i} \equiv T} y_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n}^{B_{R_j} \equiv F} \neg y_j) \tag{7.19}$$

$$F_{_SH} = \bigvee_{1 \leqslant l \leqslant m-1} ((\bigwedge_{1 \leqslant i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n}^{B_{l_j} \equiv F} \neg y_j)) \tag{7.20}$$

根据引理7.1, 有:

$$F_{_RH} \vee F_{_SH} \equiv F_H \tag{7.21}$$

$F_O = Obf(F_C, F_H, R_H)$ 以及式 (7.21), 有:

$$F_O \equiv Obf(F_C, F_{_RH} \vee F_{_SH}, R_H) \tag{7.22}$$

根据引理7.2和式 (7.22), 有:

$$F_O = Obf(F_C, F_{_RH}, R_H) \vee Obf(F_C, F_{_SH}, R_H) \tag{7.23}$$

根据定理5.1, 有:

$$Obf(F_C, F_{_R}H, R_H) \equiv F_C \wedge F_{_R}H \tag{7.24}$$

根据式 (7.23) 和 (7.24), 有:

$$F_O \equiv (F_C \wedge F_{_R}H) \vee Obf(F_C, F_{_S}H, R_H) \tag{7.25}$$

根据式 (7.25), 有:

$$F_C \wedge F_{_R}H \vdash_{sso} F_O \tag{7.26}$$

因为 $F_{_R}H$ 可满足, $V_{F_C} \cap V_{F_H} = \phi$, 根据式 (7.26):

$$F_C \vdash_{sso} F_O \tag{7.27}$$

由定理的证明可见，混淆后得到公式 F_O 的解包含两部分，一部分为 $F_C \wedge F_{_R}H$，另外一部分为 $Obf(F_C, F_{_S}H, R_H)$。其中 $F_C \wedge F_{_R}H$，由于 $F_{_R}H$ 有唯一解，因此这部分的解取决于 F_C。下面，我们讨论 $Obf(F_C, F_{_S}H, R_H)$ 解的情况。

为便于说明，我们令 $F_{_F}O = Obf(F_C, F_{_S}H, R_H)$。同样令

$$F_{slH} = (\bigwedge_{1 \leqslant i \leqslant n}^{B_{l_i} \equiv T} y_i) \wedge (\bigwedge_{1 \leqslant j \leqslant n}^{B_{l_j} \equiv F} \neg y_j)) \tag{7.28}$$

则由假设可知

$$F_{_S}H = \bigvee_{1 \leqslant l \leqslant m-1} F_{slH} \tag{7.29}$$

根据引理7.2和式子7.29,

$$F_{_F}O = \bigvee_{1 \leqslant l \leqslant m-1} (Obf(F_C, F_{slH}, R_H)) \tag{7.30}$$

让我们通过 $Obf(F_C, F_{slH}, R_H)$ 的构造过程来考察其解的情况，为便于说明，令 $F_{_{sl}}O = Obf(F_C, F_{slH}, R_H)$。根据引理7.3和式子7.28，有

$$F_{_{sl}}O = \bigwedge_{1 \leqslant i \leqslant n}^{B_{l_i} \equiv T} (Obf(F_C, y_i, R_H) \wedge \bigwedge_{1 \leqslant j \leqslant n}^{B_{l_j} \equiv F} (Obf(F_C, \neg y_j, R_H) \tag{7.31}$$

由于 F_{slH} 的解和 R_H 为同一公式的两个不同的解，那么必然具有赋值不同的变量。假设 R_H 中 y_i 赋值和 $B = \neg y_j$ 中的不同，除去 y_j 之外的无关变量，将 R_H 简化为 $y_i = T$，并将 y_j 记为 b，公式 $Obf(F_C, B = \neg y_j, R_H)$ 可表示为：$Obf(F_C, B = \neg b, b = T)$。为不失一般性，对 F_C，任取其中一个子句 A，可给出如下的条件：

1. 子句 $A = a \vee X$, 任意公式 F_{Cn} 和子句 $B = \neg b$, 并且 $b \notin X$;

2. 令 $F_{Ck} = A, F_C = F_{Ck} \wedge F_{Cn}, F_H = B$, 并且 $R_H = \{b \equiv T\}$;

3. 令 $F_O = Obf(F_C, F_H, R_H)$

根据 **Procedure** 5.4.1.1, 按照以下 3 个步骤构造 F_O。

Step1 根据 R_H 和规则 1, 构造子句 $C = A \vee \neg b$；令子句集合 $S_3 = C$

Step2 根据 R_H 和规则 2, 对于文字 $a \in A$, 构造子句 $D = \neg a \vee b$; 并且令子句集合 $S_4 = D$;

Step3 令 $F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn}$.

$$
\begin{aligned}
F_O = F_H \wedge S_3 \wedge S_4 \wedge F_{Cn} \quad & F_H = B \quad \models \\
F_O = B \wedge S_3 \wedge S_4 \wedge F_{Cn} \quad & S_3 = C \quad \models \\
F_O = B \wedge C \wedge S_4 \wedge F_{Cn} \quad & S_4 = D \quad \models \\
F_O = B \wedge C \wedge D \wedge F_{Cn} \quad & B = \neg b \quad \models \\
F_O = \neg b \wedge C \wedge D \wedge F_{Cn} \quad & C = A \vee \neg b \quad \models \\
F_O = \neg b \wedge (A \vee \neg b) \wedge D \wedge F_{Cn} \quad & \quad \models \\
F_O = \neg b \wedge D \wedge F_{Cn} \quad & D = \neg a \vee b \quad \models \\
F_O = \neg b \wedge (\neg a \vee b) \wedge F_{Cn} \quad & \quad \models \\
F_O = F_{Cn} \wedge \neg a \wedge \neg b &
\end{aligned}
\tag{7.32}
$$

同样，对于$Obf(F_C, B = b, b = F)$，按照上面的构造过程，可以推出 $F_O = F_{Cn} \wedge a \wedge b$。由于混淆导致部分关键子句 F_{Ck} 被消除，这无疑会增大解空间，导致伪解产生；另一方面，生成了一些新的子句，这可能会对解产生新的约束，而会将原本的解消除。由此可见，混淆后解和 F_C 不再等价。

综上所述，经过 SSO 混淆，在保持原解空间的基础上，还可能会产生部分伪解，这就使得混淆后的解空间是原解空间的上估计，伪解起到为原始公式加入噪音解的效果。

7.4.2 有效性分析

7.4.2.1 输入数据隐藏

通过增加冗余的文字和子句, OBFUSCATOR 可以将 CNF 公式中的标记改变为另一合法标记. 在混淆之后，原始的 CNF 公式就被转化为混有噪音电路的另一个公式。由于混淆后的的 CNF 公式被外包作为 SAT 求解器的输入, 原始 CNF 公式中的电路结构就并不再直接暴露给潜在的攻击者.

图7.3a) 和7.3b) 给出了两个 AND2 门 a 和 eCNF 标记, 混淆后的标记显示在图7.3c) 和7.3d) 中.

有三种改变

1. 关键子句 c_1 和 c_5 的长度由 3 变为 4，基于关键子句模式匹配的电路结构探测算法[12] 将不再有效;

2. a 的特征子句 c_1-c_3) 以及 e 的特征子句 c_5-c_7 变为了不同的形式, 并且有新的子句加入了公式如 c_4 和 c_8, 基于标记子图同构的电路结构检测算法[11] 将不再有效;

3. 通过加入合适的新文字以及构造新的子句, 门 a 的 CNF 标记从 AND2 变为了 AND3, 如图7.3a) 和7.3c)。Husk 文字 A, 成为了新生成的 AND3 门的一个输入, 并且与原始 AND2 门的输入 b 和 c 不可区分。这也使得区分混淆后 AND2 和真实 AND3 变得不再可能.

7.4.2.2 输出数据隐藏

根据7.1, 在 SSO 混淆之后解空间为原解空间的上估计. 这也就意味着，SAT 求解器都无法确切知道真实的解。首先，他们无法区分原始公式的变量和 Husk 公式的变量，而 Husk 公式的变量的赋值对验证毫无意义。其次，他们无法确认一个可满足解也意味着原始 SAT 问题也是可满足，因为混淆会引入假解。通过解空间上估计, 我们把一个 Rare Events 转变为了一个 Camouflaged Rare Events, 这也是文献[8] 曾经期待的事情。

$$a=\wedge(b,c)\rightarrow\begin{Bmatrix}c1:(a\vee\neg b\vee\neg c)\\c2:(\neg a\vee b)\\c3:(\neg a\vee c)\end{Bmatrix}$$

a)AND2 gate a

$$a=\wedge(b,\ c,\ \mathrm{A})\rightarrow\begin{Bmatrix}c1:(a\vee\neg b\vee\neg c\vee\neg\mathrm{A})\\c2:(\neg a\vee b\vee\mathrm{B})\\c3:(\neg a\vee c\vee\mathrm{C})\\c4:(\neg a\vee\mathrm{A}\vee\mathrm{D})\end{Bmatrix}$$

c)AND2 gate a after obfuscation

$$e=\wedge(f,g)\rightarrow\begin{Bmatrix}c5:(e\vee\neg f\vee\neg g)\\c6:(\neg e\vee f)\\c7:(\neg e\vee g)\end{Bmatrix}$$

b)AND2 gate e

$$e=\wedge(f,\ g,\ \neg E)\rightarrow\begin{Bmatrix}c5:(e\vee\neg f\vee\neg g\vee\mathrm{E})\\c6:(\neg e\vee g\vee\mathrm{F})\\c7:(\neg e\vee f\vee\mathrm{G})\\c8:(\neg e\vee\neg\mathrm{E}\vee\mathrm{H})\end{Bmatrix}$$

d)AND2 gate e after obfuscation

图 7.3　混淆前后门 a 和门 e 的 CNF 标记

7.4.3 算法复杂性分析

7.4.3.1 混淆算法的复杂性

算法7.2实现了混淆，其中主程序仅仅包含一层循环，但是其中一个子程序**mark**(算法7.3) 包含了 4 层循环, 由于两层内循环的上界为子句长度，因此混淆算法复杂性为 $O(n^2)$。

7.4.3.2 解恢复算法的复杂性

解恢复在算法7.5中实现, 由于仅仅包含一层循环, 算法复杂度为 $O(n)$. 根据定理7.1, 来自于 SAT 求解器的解可能包含假解，因此为获得正确解，算法7.5可能会运行不只一次. 由于算法7.5为线性复杂度, 带给整个求解过程的开销较小.

7.5 实验评估

7.5.1 实验设计

本文给出的算法由 C 语言实现. 实验用机器的配置为 Intel Core(TM) i7-3667U CPU @ 2.00GHz, 8GB RAM.

将 ISCAS89 测试集中的部分电路展开 100 次并编码为 CNF 公式, 产生的 Husks 公式包含了节点数和子句数为 $vn = 675/cn = 2309$, 并且将原始公式中的 2 输入门转换为 3 输入门. 使用 MiniSat 作为求解器.

7.5.2 实验结果分析

表7.1给出了实验结果，表中各个参数的意义如下所示。

vn/cn:CNF 公式中的变量数和子句数.

Marked Gate: 混淆过程中改变的门数.

Solve Times: 混淆前后的 SAT 求解时间.

Obfuscation Times: 混淆时间.

Map Time: 解恢复时间.

根据算法7.2, 混淆时间取决于改变的门数, 解恢复时间取决于 CNF 公式的尺寸, 实验表明了这一事实.

就异构加速比（Asymmetric Speedup）[79] 而言,60% 的电路, 值超过了 260%, 这表明了外包复杂 SAT 求解函数的必要性. 某些尺寸较小的电路 B，异构加速比小于 1. 特别是对于电路 s3384, 由于混淆花费了大量的时间，转换了 139860 个门, 使得异构加速比仅为 5.22%.

$$Asymmetric\ Speedup = \frac{Solving\ Time}{Obfuscation\ Times + Map\ Time} \tag{7.33}$$

表 7.1　不同类型电路的 CNF 公式的运行时间

circuit	vn	cn	Marked Gates	Solve Time(s)			Obfuscation Time(s)	Map Time(s)	Asymmetric Speedup
				Before obfuscation	After obfuscation	Solve Overhead			
s1196a	13970	50011	4397	0.056003 s	0.068004 s	21.43%	0.076004 s	0.004000 s	***70.00%***
s1196b	13970	50011	4397	0.048003 s	0.060003 s	25.00%	0.064004 s	0.012000 s	***63.16%***
s1196	13970	50011	4397	0.048003 s	0.080005 s	66.67%	0.072004 s	0.000000 s	***66.67%***
s13207	34423	117739	6221	0.280017 s	0.316019 s	12.86%	0.284017 s	0.016001 s	***93.33%***
s3384	60503	210546	*139860*	5.04031 s	12.4368 s	*146.75%*	*96.450027s*	0.132008 s	***5.22%***
s15850	181772	668093	41471	16.361 s	46.0069 s	*181.20%*	1.392087 s	0.048003 s	1136.11%
s3271	57976	219807	14597	5.97237 s	6.4044 s	7.23%	1.396087 s	0.020001 s	421.75%
s3330	49635	159813	9679	7.9405 s	7.28045 s	**-8.31%**	0.264016 s	0.016001 s	2835.72%
s38417	492876	1939718	9563	652.685 s	3538.12 s	442.09%	0.412025 s	0.012000 s	153926.07%
s38584	593381	2227470	66491	496.847 s	1092.3 s	119.85%	9.756609 s	0.120007 s	5030.54%
s4863	85963	320633	*128896*	352.746 s	352.606 s	**-0.04%**	*133.788361s*	0.140008 s	263.38%
s5378	59462	209372	14096	11.9407 s	8.73655 s	**-26.83%**	0.428026 s	0.020001 s	2665.17%
s6669	115303	408306	10868	1.31608 s	1.6081 s	22.19%	0.296018 s	0.024001 s	411.25%
s9234	87484	337020	15392	244.627 s	292.058 s	19.39%	0.728045 s	0.032002 s	32185.77%
s35932	584534	2135163	23418	282.17 s	714.573 s	*153.24%*	0.676042 s	0.024001 s	40307.52%

实验也显示出 SAT 求解的开销, 不同的电路具有不同的表现。60 % 以上的电路, 开销小于 30%; 而 40% 电路, 开销超过了 100%.

这些事实提醒我们两件事情：首先, 混淆时间取决于被改变的门数, 需要研究更加精巧的混淆算法以改变较少的门的情况下仍然可以迷惑攻击者. 第二, 由于混淆引入的 SAT 求解开销，因电路而异，在设计混淆算法时，需要考虑修改后结构对求解时间的影响。

7.6　结论

本文给出了电路结构感知且解空间加噪的 CNF 混淆算法，可防止在 SAT 问题外包计算时，CNF 公式中的电路结构以及解被窃取。理论分析和实验表明，算法可以有效的改变结构，同时还不会缩减 CNF 公式的解空间。

第八章　结束语

本章对全文进行总结，并对进一步研究工作进行展望。

8.1　工作总结

SAT 求解计算外包中的隐私保护是一项有趣而富有挑战的工作。本文针对软硬件设计和验证中的 SAT 求解为研究对象，从问题具体特征着手，系统深入地研究了 SAT 求解过程中输入数据和输出数据的保护问题。具体而言，本文主要对以下几个重要问题进行了深入研究。

1. 开放环境下基于加噪 CNF 混淆的 SAT 求解框架。CNF 公式混淆是保证开放环境下的 SAT 求解隐私性的重要手段。现有的方法基于双射或多射群加密，通过分段坡度编码对 CNF 公式进行混淆，从而隐藏 CNF 公式中携带的结构信息，但是这种方法改变了原有 CNF 公式的外部表示，需要设计新的求解算法，并且在目前仅可使用全空间遍历的方法进行求解，无法利用已有成熟的 SAT 求解算法，因此极大降低了其实用性。本文通过对 CNF 公式自身逻辑特点的分析，提出了基于加噪思路的混淆算法，通过在原始 CNF 公式中无缝的混入噪音公式，在隐藏原有的结构信息的同时，保持原有的 CNF 数据形式和解空间，从而复用目前已有的求解算法；在算法的实用性和隐私保护上取得了良好的折衷。本文从理论上证明了算法的正确性并通过大量的仿真实验验证了算法的性能。

2、解空间投影等价的 CNF 混淆算法假设攻击者已经确切知道公式中夹杂了噪音变量和子句，因而试图从分析解的角度出发，先还原 CNF 公式，而后再获取其中携带的结构信息。通过引入具有簇形解的噪音 CNF 公式，使噪音变量的取值不再唯一，消除攻击者利用 ALLSAT 还原 CNF 公式的可能性，本文从理论上证明了算法的正确性。

3. 解空间加噪的 CNF 混淆算法。由于 SAT 求解时，输出数据也包含了敏感信息，在研究了隐藏结构信息的 CNF 混淆算法之后，本文进一步研究了隐藏 CNF 解的混淆方法。针对输出信息保护，本文提出了解空间上估计的 CNF 混淆算法，通过扩展噪音公式解空间，使得混淆后 SAT 问题的解空间为原始解空间的上估计，通过引入噪音解实现对原始解的隐藏。本文从理论上充分证明了算法的正确性，并通过大量的仿真实验验证了方法的有效性和性能。

4. CNF 混淆算法的有效性评价。混淆算法保证混淆后的 CNF 公式可用已有的求解器求解，并且可以用较小的开销恢复出原始的解，但是除此之外，在开放环境下为保证 SAT 计算外包的顺利实施，混淆算法仍然需要满足其他的特性。结合程序混淆的有效性评价标准，本文抽象出 CNF 公式混淆的有效性评价标准，通

过对混淆策略的细分，针对两种混淆策略进行定量分析和定性评价，为设计出更好的混淆策略提供了依据，

8.2 研究展望

本文深入研究了 SAT 求解计算外包的隐私保护问题，在软硬件验证和设计领域 SAT 问题的隐私保护方法上取得了一定的研究成果，但由于开放环境下 SAT 问题计算外包涉及的因素多，该领域还存在许多问题需要进一步的研究。在本文研究的基础上，需要进一步研究的课题包括：

第一，轻量级的结构感知混淆策略设计。本文设计的结构感知混淆算法虽然对隐藏原有结构信息具有很好的效果，但算法中检测所有常用结构，这种检测方法在保证完备性的情况下，需要付出较大的检测和混淆开销。因此，如何进一步地深入挖掘和利用 CNF 结构信息，设计仅针对部分关键 CNF 结构进行修改仍然具有良好混淆效果的算法，是一项值得研究且具有一定挑战性的工作。

第二，求解性能敏感的混淆策略设计。通常 SAT 问题的复杂度和其问题结构具有一定的关联，本文提出的基于加噪的混淆算法，主要面向软硬件设计和验证领域 SAT 问题的隐私保护需求，更多关注结构信息隐藏而未考虑结构改变带来复杂度变化，对某些通用类型的 SAT 问题，有可能出现混淆后 SAT 问题求解时间跃变的情况。因此，研究各种类型 SAT 问题结构和求解复杂度之间的关系，避免因混淆引发的结构改变导致求解时间跃变的情况发生，是未来改进混淆算法使其更具通用性，值得研究的问题。

第三，解空间加噪的混淆算法需要从多个伪装解中筛查出真实解，因此需要出现多次交互开销。结合程序混淆技术，对位于云端的标准求解器进行功能改造，从而使得真实解过滤过程可以通过一次交互来实现，并通过程序混淆技术防范攻击者对其进行分析，也是一个及其具有实用价值的研究方向。

致　谢

值此成文之际，谨向在我攻读博士期间给予我指导、关心、支持和帮助的老师、领导、同学和亲人们致以衷心的感谢！

首先，衷心感谢我的导师贾焰老师为我提供宝贵的学习机会！在课题选择和问题解决过程中，以敏锐的学术洞察力和深厚的科研经验，高屋建瓴地为我论证把关。您在百忙之中仍抽出时间对我的课题进行指导，及时为我解决困难和提供帮助。没有您的指导和帮助，我的研究工作将不可能顺利完成。贾老师对我的言传身教将使我终身受益，您严谨的治学作风、高深的学术造诣、忘我的工作态度和勇攀高峰的精神将永远影响和激励着我。

衷心感谢学院廖湘科老师，工程中心吴庆波老师、戴华东老师、李佩江政委和孔金珠老师为我提供宽松的科研环境，有效保证了我博士课题的研究时间，使我的研究工作能够顺利进行。

衷心感谢我的师姐韩伟红老师！在整个博士课题研究过程中，韩师姐始终给予我热情的指点与帮助，使我的研究工作能够顺利进行。韩师姐严谨的工作态度、细致入微的方法指导，无不使我深感敬佩并始终将师姐作为我学习的榜样。您在生活中给与了我无微不至的关怀，您的热情大方、细心体贴和乐观开朗时刻温暖和鼓励着我。

衷心感谢颜跃进、杨沙洲、刘晓健、汪黎等同事，你们在工程实践和课题研究过程中都提出了良好的建议。感谢唐晓东、陈松政、魏立峰、何连跃、阳国贵、任怡、董攀、邵立松、马俊、易晓东、高珑、谭郁松、王小川、张卫华等同事，大家勤奋投入的工作态度和超强的科研与工程能力一直都是我学习的目标，与大家并肩奋斗的日子使我得到了良好的锻炼，对今后遇到的任何困难都能够从容面对。

衷心感谢曾经一起战斗过的同事丁滟，你始终如一的勤奋给与我很多的激励。感谢李姗姗、刘晓东、林斌、鲁晓佩、郑思等廖们的师妹师弟们，你们朝气蓬勃、思维活跃，参加你们的讨论会，使我开阔了视野。感谢博士生队的李小芳、周静、阳柳、王晓斌等同学，尽管接触的机会并不太多，但是每次有困难的时候总能得到同学们的援手，非常感谢。

感谢学院、学员大队、学员队各级领导对我的教育、关心和帮助，你们的辛勤工作为我们创造了良好的学习和生活环境。

特别感谢我的家人。感谢沈胜宇，与你的每一次交流，都是一场大考，让我得以从固有思维中解放出来，对科学研究有了进一步的认识，从追求一个个小目

标开始，逐渐品尝到科研中蕴含的各种滋味，并达到最终的目标。感谢我的儿子，你的笑容是我最大的安慰，让我在失落中重新拾起勇气；你的懂事和成长给与我力量，激励着我，无论道路如何坎坷我都会坚持走下去。

特别感谢含辛茹苦将我抚育成人并始终宽容待我，随时为我提供无私帮助的父母亲和我的姐姐，对你们的感激之情无法用语言来表达，在我多年的求学和工作之路上，你们始终给与我全身心的支持和关爱。我无法在生活上给与你们照顾，唯有在将来的工作中刻苦努力、做出更大的成绩，以报答你们的养育之恩。祝愿你们永远健康幸福!

参考文献

[1] Davis M, Putnam H. A Computing Procedure for Quantification Theory [J/OL]. J. ACM. 1960, 7 (3): 201–215. http://doi.acm.org/10.1145/321033.321034.

[2] Hachtel G D, Somenzi F. Logic synthesis and verification algorithms [M/OL]. Springer, 2006. http://dx.doi.org/10.1007/0-387-31005-3.

[3] Clarke E M, Grumberg O, Jha S, et al. Counterexample-Guided Abstraction Refinement [C/OL]. In Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings. 2000: 154–169. http://dx.doi.org/10.1007/10722167_15.

[4] Soos M, Nohl K, Castelluccia C. Extending SAT Solvers to Cryptographic Problems [C/OL] // Kullmann O. In Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings. 2009: 244–257. http://dx.doi.org/10.1007/978-3-642-02777-2_24.

[5] Hyvärinen A E J, Junttila T A, Niemelä I. Grid-Based SAT Solving with Iterative Partitioning and Clause Learning [C/OL] // Lee J H. In Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings. 2011: 385–399. http://dx.doi.org/10.1007/978-3-642-23786-7_30.

[6] Paralleling OpenSMT Towards Cloud Computing.http://www.inf.usi.ch/urop-Tsitovich-2-127208.pdf.

[7] Formal in the Cloud OneSpin: New Spin on Cloud Computing.http://www.eejournal.com/archives/articles/20130627-onespin/?printView=true.

[8] Du W, Goodrich M T. Searching for High-Value Rare Events with Uncheatable Grid Computing [C/OL] // Ioannidis J, Keromytis A D, Yung M. In Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings. 2005: 122–137. http://dx.doi.org/10.1007/11496137_9.

[9] Balduzzi M, Zaddach J, Balzarotti D, et al. A security analysis of amazon's elastic compute cloud service [C/OL] // Ossowski S, Lecca P. In Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012. 2012: 1427–1434. http://doi.acm.org/10.1145/2245276.2232005.

[10] Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds [C/OL] // Al-Shaer E, Jha S, Keromytis A D. In Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009. 2009: 199–212. http://doi.acm.org/10.1145/1653662.1653687.

[11] Roy J A, Markov I L. Restoring Circuit Structure from SAT Instances. 2004.

[12] Fu Z, Malik S. Extracting Logic Circuit Structure from Conjunctive Normal Form Descriptions [C/OL]. In 20th International Conference on VLSI Design (VLSI Design 2007), Sixth International Conference on Embedded Systems (ICES 2007), 6-10 January 2007, Bangalore, India. 2007: 37–42. http://doi.ieeecomputersociety.org/10.1109/VLSID.2007.81.

[13] Brun Y, Medvidovic N. Keeping Data Private while Computing in the Cloud [C/OL] // Chang R. In 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29, 2012. 2012: 285–294. http://dx.doi.org/10.1109/CLOUD.2012.126.

[14] Chrabakh W, Wolski R. The GridSAT portal: a Grid Web-based portal for solving satisfiability problems using the national cyberinfrastructure [J/OL]. Concurrency and Computation: Practice and Experience. 2007, 19 (6): 795–808. http://dx.doi.org/10.1002/cpe.1079.

[15] Sahai A, Waters B. Fuzzy Identity Based Encryption [J/OL]. IACR Cryptology ePrint Archive. 2004, 2004: 86. http://eprint.iacr.org/2004/086.

[16] Goyal V, Pandey O, Sahai A, et al. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data [J/OL]. IACR Cryptology ePrint Archive. 2006, 2006: 309. http://eprint.iacr.org/2006/309.

[17] Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data [C/OL] // Juels A, Wright R N, di Vimercati S D C. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006. 2006: 89–98. http://doi.acm.org/10.1145/1180405.1180418.

[18] Curtmola R, Garay J A, Kamara S, et al. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions [J/OL]. IACR Cryptology ePrint Archive. 2006, 2006: 210. http://eprint.iacr.org/2006/210.

[19] Curtmola R, Garay J A, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions [C/OL] // Juels A, Wright R N, di Vimercati S D C. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006. 2006: 79–88. http://doi.acm.org/10.1145/1180405.1180417.

[20] Curtmola R, Garay J A, Kamara S, et al. Searchable symmetric encryption: Improved definitions and efficient constructions [J/OL]. Journal of Computer Security. 2011, 19 (5): 895–934. http://dx.doi.org/10.3233/JCS-2011-0426.

[21] Gentry C. Fully homomorphic encryption using ideal lattices [C/OL] // Mitzenmacher M. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. 2009: 169–178. http://doi.acm.org/10.1145/1536414.1536440.

[22] Sorensson N. MiniSat-SAT Algorithms and Applications.

[23] Tseitin G. On the complexity of derivations in propositional calculus [J]. 1983.

[24] Zhang L, Malik S. Conflict driven learning in a quantified Boolean Satisfiability solver [C/OL] // Pileggi L T, Kuehlmann A. In Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, ICCAD 2002, San Jose, California, USA, November 10-14, 2002. 2002: 442–449. http://doi.acm.org/10.1145/774572.774637.

[25] Eén N, Sörensson N. An extensible sat-solver [M] // Enrico Giunchiglia A T. Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003Vol.2919. Berlin Heidelberg: Springer-Verlag, 2003: 2003: 502–518.

[26] Zhang L, Madigan C F, Moskewicz M W, et al. Efficient Conflict Driven Learning in Boolean Satisfiability Solver [C]. In Proceedings of the 2001 International Conference on Computer-Aided Design, ICCAD 2001. 2001: 279–285.

[27] McMillan K L. Applying sat methods in unbounded symbolic model checking [M] // Ed Brinksma K G L. International Conference on Computer Aided Verification, CAV 2002Vol.2404. Berlin Heidelberg: Springer-Verlag, 2002: 2002: 250–264.

[28] Ravi K, Somenzi F. Minimal assignments for bounded model checking [M] // Kurt Jensen A P. Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004Vol.2988. Berlin Heidelberg: Springer-Verlag, 2004: 2004: 31–45.

[29] Chauhan P, Clarke E M, Kroening D. A sat-based algorithm for reparameterization in symbolic simulation [C]. In Proceedings of the 41th Design Automation Conference, DAC 2004. 2004: 524–529.

[30] Shen S, Qin Y, Li S. Minimizing counterexample with unit core extraction and incremental sat [M] // Cousot R. Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005Vol.3385. Berlin Heidelberg: Springer-Verlag, 2005: 2005: 298–312.

[31] Jin H, Somenzi: F. Prime clauses for fast enumeration of satisfying assignments to boolean circuits [C/OL]. In Proceedings of the 42th Design Automation Conference, DAC 2005. 2005: 750–753. http://dx.doi.org/10.1109/DAC.2005.193911.

[32] Jin H, Han H, Somenzi F. Efficient conflict analysis for finding all satisfying assignments of a boolean circuit [M] // Nicolas Halbwachs L D Z. Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005Vol.3440. Berlin Heidelberg: Springer-Verlag, 2005: 2005: 287–300.

[33] Grumberg O, Schuster A, Yadgar A. Memory efficient all-solutions sat solver and its application for reachability analysis [M] // Alan J Hu A K M. International Conference on Formal Methods in Computer-Aided Design,FMCAD 2011Vol.3312. Berlin Heidelberg: Springer-Verlag, 2004: 2004: 275–289.

[34] Ganai M K, Gupta A, Ashar P. Efficient sat-based unbounded symbolic model checking using circuit cofactoring [C/OL]. In Proceedings of the 2004 International Conference on Computer-Aided Design, ICCAD 2004. 2004: 510–517. http://dx.doi.org/10.1109/ICCAD.2004.1382631.

[35] Jie-Hong Roland Jiang W-L H, Hsuan-Po Lin. Interpolating functions from large Boolean relations [C]. In Proceedings of 2009 International Conference on Computer-Aided Design. 2009: 779–784.

[36] Chockler H, Ivrii A, Matsliah A. Computing Interpolants without Proofs [M] // Armin Biere T E J V, Amir Nahir. 8th International Haifa Verification Conference, HVC 2012Vol.7857. Berlin Heidelberg: Springer-Verlag, 2012: 2012: 72–85.

[37] Craig W. Linear reasoning: A new form of the herbrand-gentzen theorem [J]. The Journal of Symbolic Logic. 1957, 22 (3): 250–268.

[38] McMillan K L. Interpolation and sat-based model checking [M] // Warren A Hunt Jr F S. Computer Aided Verification, 15th International Conference, CAV 2003Vol.2725. Berlin Heidelberg: Springer-Verlag, 2003: 2003: 1–13.

[39] McMillan K L. An interpolating theorem prover [J/OL]. Theor. Comput. Sci. 2005, 345 (1): 101–121. http://dx.doi.org/10.1016/j.tcs.2005.07.003.

[40] Abdulla P A, Bjesse P, Eén N. Symbolic Reachability Analysis Based on SAT-Solvers [C/OL] // Graf S, Schwartzbach M I. In Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings. 2000: 411–425. http://dx.doi.org/10.1007/3-540-46419-0_28.

[41] Brummayer R, Biere A. Local two-level And-Inverter Graph minimization without blowup [C]. In In Proceedings of the 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS’06. 2006.

[42] Bryant R E. Graph-Based Algorithms for Boolean Function Manipulation [J/OL]. IEEE Trans. Computers. 1986, 35 (8): 677–691. http://doi.ieeecomputersociety.org/10.1109/TC.1986.1676819.

[43] euro1. http://voices.washingtonpost.com/securityfix/2007/10/database_theft_leads_to_target.html.

[44] euro1. http://www.wired.com/2010/09/google-spy/.

[45] euro1. http://www.gartner.com/smarterwithgartner/assessing-security-in-the-cloud/.

[46] euro1. https://cloudsecurityalliance.org/events/cloud-security-alliance-summit-at-rsa-2011/.

[47] euro1. http://www10.edacafe.com/blogs/thebrekertrekker/2015/07/15/guest-post-rain-or-shine-for-the-eda-cloud/.

[48] Foster I T, Zhao Y, Raicu I, et al. Cloud Computing and Grid Computing 360-Degree Compared [J/OL]. CoRR. 2009, abs/0901.0131. http://arxiv.org/abs/0901.0131.

[49] Brakerski Z, Rothblum G N. Black-box obfuscation for d-CNFs [C/OL] // Naor M. In Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014. 2014: 235–250. http://doi.acm.org/10.1145/2554797.2554820.

[50] Rivest R L, Shamir A, Adleman L M. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems [J/OL]. Commun. ACM. 1978, 21 (2): 120–126. http://doi.acm.org/10.1145/359340.359342.

[51] Gamal T E. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms [C/OL] // Blakley G R, Chaum D. In Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. 1984: 10–18. http://dx.doi.org/10.1007/3-540-39568-7_2.

[52] Goldwasser S, Micali S. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information [C/OL] // Lewis H R, Simons B B, Burkhard W A, et al. In Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA. 1982: 365–377. http://doi.acm.org/10.1145/800070.802212.

[53] Paillier P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes [C/OL] // Stern J. In Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. 1999: 223–238. http://dx.doi.org/10.1007/3-540-48910-X_16.

[54] Implementing Gentry's fully-homomorphic encryption scheme. [J/OL]. 2011: 129–148. http://dx.doi.org/110.1007/978-3-642-20465-4_9.

[55] Improved key generation for Gentry’ s fully homomorphic encryption scheme. [J/OL]. 2011: 10–22. http://dx.doi.org/110.1007/978-3-642-25516-8_2.

[56] Stehlé D, Steinfeld R. Faster Fully Homomorphic Encryption [C/OL] // Abe M. In Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. 2010: 377–394. http://dx.doi.org/10.1007/978-3-642-17373-8_22.

[57] Smart N P, Vercauteren F. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes [C/OL] // Nguyen P Q, Pointcheval D. In Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings. 2010: 420–443. http://dx.doi.org/10.1007/978-3-642-13013-7_25.

[58] Gentry C. Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness [C/OL] // Rabin T. In Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. 2010: 116–137. http://dx.doi.org/10.1007/978-3-642-14623-7_7.

[59] López-Alt A, Tromer E, Vaikuntanathan V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption [C/OL] // Karloff H J, Pitassi T. In Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012. 2012: 1219–1234. http://doi.acm.org/10.1145/2213977.2214086.

[60] Bos J W, Lauter K E, Loftus J, et al. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme [C/OL] // Stam M. In Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings. 2013: 45–64. http://dx.doi.org/10.1007/978-3-642-45239-0_4.

[61] Brakerski Z. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP [C/OL] // Safavi-Naini R, Canetti R. In Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. 2012: 868–886. http://dx.doi.org/10.1007/978-3-642-32009-5_50.

[62] van Dijk M, Gentry C, Halevi S, et al. Fully Homomorphic Encryption over the Integers [C/OL] // Gilbert H. In Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. 2010: 24–43. http://dx.doi.org/10.1007/978-3-642-13190-5_2.

[63] Coron J, Mandal A, Naccache D, et al. Fully Homomorphic Encryption over the Integers with Shorter Public Keys [C/OL] // Rogaway P. In Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. 2011: 487–504. http://dx.doi.org/10.1007/978-3-642-22792-9_28.

[64] Coron J, Naccache D, Tibouchi M. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers [C/OL] // Pointcheval D, Johansson T. In Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. 2012: 446–464. http://dx.doi.org/10.1007/978-3-642-29011-4_27.

[65] Chen Y, Nguyen P Q. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers [C/OL] // Pointcheval D, Johansson T. In Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. 2012: 502–519. http://dx.doi.org/10.1007/978-3-642-29011-4_30.

[66] Cheon J H, Coron J, Kim J, et al. Batch Fully Homomorphic Encryption over the Integers [C/OL] // Johansson T, Nguyen P Q. In Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. 2013: 315–335. http://dx.doi.org/10.1007/978-3-642-38348-9_20.

[67] Gu C. Attack on Fully Homomorphic Encryption over the Integers [J/OL]. CoRR. 2012, abs/1202.3321. http://arxiv.org/abs/1202.3321.

[68] Cheon J H, Kim J, Lee M S, et al. CRT-based fully homomorphic encryption over the integers [J/OL]. Inf. Sci. 2015, 310: 149–162. http://dx.doi.org/10.1016/j.ins.2015.03.019.

[69] Regev O. The Learning with Errors Problem (Invited Survey) [C/OL]. In Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010. 2010: 191–204. http://doi.ieeecomputersociety.org/10.1109/CCC.2010.26.

[70] Gentry C, Halevi S, Smart N P. Fully Homomorphic Encryption with Polylog Overhead [C/OL] // Pointcheval D, Johansson T. In Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. 2012: 465–482. http://dx.doi.org/10.1007/978-3-642-29011-4_28.

[71] Lyubashevsky V, Peikert C, Regev O. On Ideal Lattices and Learning with Errors over Rings [C/OL] // Gilbert H. In Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. 2010: 1–23. http://dx.doi.org/10.1007/978-3-642-13190-5_1.

[72] Gentry C, Halevi S, Smart N P. Better Bootstrapping in Fully Homomorphic Encryption [C/OL] // Fischlin M, Buchmann J A, Manulis M. In Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings. 2012: 1–16. http://dx.doi.org/10.1007/978-3-642-30057-8_1.

[73] Brakerski Z, Gentry C, Halevi S. Packed Ciphertexts in LWE-Based Homomorphic Encryption [C/OL] // Kurosawa K, Hanaoka G. In Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings. 2013: 1–13. http://dx.doi.org/10.1007/978-3-642-36362-7_1.

[74] Brakerski Z, Vaikuntanathan V. Efficient Fully Homomorphic Encryption from (Standard) LWE [J/OL]. SIAM J. Comput. 2014, 43 (2): 831–871. http://dx.doi.org/10.1137/120868669.

[75] Brakerski Z, Vaikuntanathan V. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages [C/OL] // Rogaway P. In Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. 2011: 505–524. http://dx.doi.org/10.1007/978-3-642-22792-9_29.

[76] Peikert C, Vaikuntanathan V, Waters B. A Framework for Efficient and Composable Oblivious Transfer [C/OL] // Wagner D. In Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. 2008: 554–571. http://dx.doi.org/10.1007/978-3-540-85174-5_31.

[77] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) Fully Homomorphic Encryption without Bootstrapping [J/OL]. TOCT. 2014, 6 (3): 13:1–13:36. http://doi.acm.org/10.1145/2633600.

[78] Atallah M J, Pantazopoulos K N, Rice J R, et al. Secure outsourcing of scientific computations [J/OL]. Advances in Computers. 2001, 54: 215–272. http://dx.doi.org/10.1016/S0065-2458(01)80019-X.

[79] Wang C, Ren K, Wang J. Secure and practical outsourcing of linear programming in cloud computing [C/OL]. In INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China. 2011: 820–828. http://dx.doi.org/10.1109/INFCOM.2011.5935305.

[80] Wang C, Ren K, Wang J, et al. Harnessing the Cloud for Securely Outsourcing Large-Scale Systems of Linear Equations [J/OL]. IEEE Trans. Parallel Distrib. Syst. 2013, 24 (6): 1172–1181. http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.206.

[81] Qin Y, Shen S, Kong J, et al. Cloud-Oriented SAT Solver Based on Obfuscating CNF Formula [C/OL] // Han W, Huang Z, Hu C, et al. In Web Technologies and Applications - APWeb 2014 Workshops, SNA, NIS, and IoTS, Changsha, China, September 5, 2014. Proceedings. 2014: 188–199. http://dx.doi.org/10.1007/978-3-319-11119-3_18.

[82] Qin Y, Shen S, Jia Y. Structure-Aware CNF Obfuscation for Privacy-Preserving SAT Solving [C]. In Proceedings of the 12th ACM/IEEE International Conference on Formal Methods and Models for System Design. 2014: 84–93.

[83] Atallah M J, Li J. Secure outsourcing of sequence comparisons [J/OL]. Int. J. Inf. Sec. 2005, 4 (4): 277–287. http://dx.doi.org/10.1007/s10207-005-0070-3.

[84] Atallah M J, Frikken K B. Securely outsourcing linear algebra computations [C/OL] // Feng D, Basin D A, Liu P. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010. 2010: 48–59. http://doi.acm.org/10.1145/1755688.1755695.

[85] Shamir A. How to Share a Secret [J/OL]. Commun. ACM. 1979, 22 (11): 612–613. http://doi.acm.org/10.1145/359168.359176.

[86] Du W, Jia J, Mangal M, et al. Uncheatable Grid Computing [C/OL]. In 24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan. 2004: 4–11. http://dx.doi.org/10.1109/ICDCS.2004.1281562.

[87] Golle P, Mironov I. Uncheatable Distributed Computations [C/OL] // Naccache D. In Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. 2001: 425–440. http://dx.doi.org/10.1007/3-540-45353-9_31.

[88] Szajda D, Lawson B G, Owen J. Hardening Functions for Large Scale Distributed Computations [C/OL]. In 2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA. 2003: 216–224. http://doi.ieeecomputersociety.org/10.1109/SECPRI.2003.1199338.

[89] Blanton M, Zhang Y, Frikken K B. Secure and Verifiable Outsourcing of Large-Scale Biometric Computations [C/OL]. In PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011. 2011: 1185–1191. http://doi.ieeecomputersociety.org/10.1109/PASSAT/SocialCom.2011.13.

[90] Chang R. 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29, 2012 [C/OL]. IEEE, 2012.

[91] Wang Y, Wei J, Srivatsa M. Result Integrity Check for MapReduce Computation on Hybrid Clouds [C]. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on. June 2013: 847–854.

[92] Wang Y, Wei J, Srivatsa M, et al. IntegrityMR: Integrity assurance framework for big data analytics and management applications [C/OL] // Hu X, Lin T Y, Raghavan V, et al. In Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA. 2013: 33–40. http://dx.doi.org/10.1109/BigData.2013.6691780.

[93] Gennaro R, Gentry C, Parno B. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers [C/OL] // Rabin T. In Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. 2010: 465–482. http://dx.doi.org/10.1007/978-3-642-14623-7_25.

[94] C Gentry. A fully homomorphic encryption scheme. [PhD thesis], Stanford University, 2009.

[95] Yao A C. Protocols for Secure Computations (Extended Abstract) [C/OL]. In 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. 1982: 160–164. http://doi.ieeecomputersociety.org/10.1109/SFCS.1982.38.

[96] Barak B, Goldreich O, Impagliazzo R, et al. On the (im)possibility of obfuscating programs [J/OL]. J. ACM. 2012, 59 (2): 6. http://doi.acm.org/10.1145/2160158.2160159.

[97] XS Zhang F H, Zuo W. Theory and Practice of Program Obfuscation [C]. In Convergence and Hybrid Information Technologies. 2010: 426.

[98] Barak B, Goldreich O, Impagliazzo R, et al. On the (Im)possibility of Obfuscating Programs [C/OL] // Kilian J. In Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. 2001: 1–18. http://dx.doi.org/10.1007/3-540-44647-8_1.

[99] Lynn B, Prabhakaran M, Sahai A. Positive Results and Techniques for Obfuscation [C/OL] // Cachin C, Camenisch J. In Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. 2004: 20–39. http://dx.doi.org/10.1007/978-3-540-24676-3_2.

[100] Goldwasser S, Rothblum G N. On Best-Possible Obfuscation [C/OL] // Vadhan S P. In Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings. 2007: 194–213. http://dx.doi.org/10.1007/978-3-540-70936-7_11.

[101] Goldwasser S, Rothblum G N. On Best-Possible Obfuscation [J/OL]. J. Cryptology. 2014, 27 (3): 480–505. http://dx.doi.org/10.1007/s00145-013-9151-z.

[102] Garg S, Gentry C, Halevi S, et al. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits [C/OL]. In 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA. 2013: 40–49. http://dx.doi.org/10.1109/FOCS.2013.13.

[103] Brakerski Z, Rothblum G N. Obfuscating Conjunctions [C/OL] // Canetti R, Garay J A. In Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. 2013: 416–434. http://dx.doi.org/10.1007/978-3-642-40084-1_24.

[104] Batchelder M, Hendren L J. Obfuscating Java: The Most Pain for the Least Gain [C/OL] // Krishnamurthi S, Odersky M. In Compiler Construction, 16th International Conference, CC 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 26-30, 2007, Proceedings. 2007: 96–110. http://dx.doi.org/10.1007/978-3-540-71229-9_7.

[105] Collberg C, Thomborson C, Low D. A Taxonomy of Obfuscating Transformations. Tech. Report, #148, University of Auckland.

[106] Lofstrom K, Daasch W, Taylor D. IC identification circuit using device mismatch [C]. In Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International. Feb 2000: 372–373.

[107] Oliveira A L. Robust Techniques for Watermarking Sequential Circuit Designs [C/OL]. In DAC. 1999: 837–842. http://doi.acm.org/10.1145/309847.310082.

[108] Koushanfar F, Qu G, Potkonjak M. Intellectual Property Metering [C/OL] // Moskowitz I S. In Information Hiding, 4th International Workshop, IHW 2001, Pittsburgh, PA, USA, April 25-27, 2001, Proceedings. 2001: 81–95. http://dx.doi.org/10.1007/3-540-45496-9_7.

[109] Su Y, Holleman J, Otis B P. A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations [C/OL]. In 2007 IEEE International Solid-State Circuits Conference, ISSCC 2007, Digest of Technical Papers, San Francisco, CA, USA, February 11-15, 2007. 2007: 406–611. http://dx.doi.org/10.1109/ISSCC.2007.373466.

[110] Suh G E, Devadas S. Physical Unclonable Functions for Device Authentication and Secret Key Generation [C/OL]. In Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, 2007. 2007: 9–14. http://doi.acm.org/10.1145/1278480.1278484.

[111] Koushanfar F. Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management [J/OL]. IEEE Transactions on Information Forensics and Security. 2012, 7 (1): 51–63. http://dx.doi.org/10.1109/TIFS.2011.2163307.

[112] Koushanfar F. Integrated circuits metering for piracy protection and digital rights management: an overview [C/OL] // Atienza D, Xie Y, Ayala J L, et al. In Proceedings of the 21st ACM Great Lakes Symposium on VLSI 2010, Lausanne, Switzerland, May 2-6, 2011. 2011: 449–454. http://doi.acm.org/10.1145/1973009.1973110.

[113] Li L, Zhou H. Structural transformation for best-possible obfuscation of sequential circuits [C/OL]. In 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013. 2013: 55–60. http://dx.doi.org/10.1109/HST.2013.6581566.

[114] Roy J A, Koushanfar F, Markov I L. EPIC: Ending Piracy of Integrated Circuits [C/OL]. In Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008. 2008: 1069–1074. http://dx.doi.org/10.1109/DATE.2008.4484823.

[115] Roy J A, Koushanfar F, Markov I L. Protecting bus-based hardware IP by secret sharing [C/OL] // Fix L. In Proceedings of the 45th Design Automation Conference, DAC 2008, Anaheim, CA, USA, June 8-13, 2008. 2008: 846–851. http://doi.acm.org/10.1145/1391469.1391684.

[116] Huang J, Lach J. IC Activation and User Authentication for Security-Sensitive Systems [C/OL] // Tehranipoor M, Plusquellic J. In IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2008, Anaheim, CA, USA, June 9, 2008. Proceedings. 2008: 76–80. http://dx.doi.org/10.1109/HST.2008.4559056.

[117] Oliveira A L. Techniques for the creation of digital watermarks in sequential-circuit designs [J/OL]. IEEE Trans. on CAD of Integrated Circuits and Systems. 2001, 20 (9): 1101–1117. http://doi.ieeecomputersociety.org/10.1109/43.945306.

[118] Koushanfar F, Alkabani Y. Provably Secure Obfuscation of Diverse Watermarks for Sequential Circuits [C/OL] // Plusquellic J, Mai K. In HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA. 2010: 42–47. http://dx.doi.org/10.1109/HST.2010.5513115.

[119] Yuan L, Qu G. Information Hiding in Finite State Machine [C/OL] // Fridrich J J. In Information Hiding, 6th International Workshop, IH 2004, Toronto, Canada, May 23-25, 2004, Revised Selected Papers. 2004: 340–354. http://dx.doi.org/10.1007/978-3-540-30114-1_24.

[120] Hopcroft J E, Motwani R, Ullman J D. Introduction to automata theory, languages, and computation - international edition (2. ed) [M]. Addison-Wesley, 2003.

[121] Clarke E M, Grumberg O, Jha S, et al. Counterexample-Guided Abstraction Refinement [C/OL]. In Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings. 2000: 154–169. http://dx.doi.org/10.1007/10722167_15.

[122] Yuan J, Albin K, Aziz A, et al. Constraint synthesis for environment modeling in functional verification [C/OL]. In Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003. 2003: 296–299. http://doi.acm.org/10.1145/775832.775909.

[123] Amla N, McMillan K L. A Hybrid of Counterexample-Based and Proof-Based Abstraction [C/OL]. In Formal Methods in Computer-Aided Design, 5th International Conference, FMCAD 2004, Austin, Texas, USA, November 15-17, 2004, Proceedings. 2004: 260–274. http://dx.doi.org/10.1007/978-3-540-30494-4_19.

[124] DesignCompiler. http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx.

[125] cudd. http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html.

[126] Shen S, Zhang J, Qin Y, et al. Synthesizing complementary circuits automatically [C/OL]. In Proceedings of the 2009 International Conference on Computer-Aided Design. San Jose, CA, USA, 2009: 381–388. http://dx.doi.org/10.1145/1687399.1687472.

[127] Shen S, Qin Y, Wang K, et al. Synthesizing Complementary Circuits Automatically [J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2010, 29 (8): 29:1191–29:1202. http://doi.acm.org/10.1109/TCAD.2010.2049152.

[128] Shen S, Qin Y, Zhang J, et al. A halting algorithm to determine the existence of decoder [C/OL] // Bloem R, Sharygina N. In Proceedings of 10th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2010, Lugano, Switzerland, October 20-23. 2010: 91–99. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5770937.

[129] Shen S, Qin Y, Xiao L, et al. A Halting Algorithm to Determine the Existence of the Decoder [J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2011, 30 (10): 30:1556–30:1563. http://doi.acm.org/10.1109/TCAD.2011.2159792.

[130] Shen S, Qin Y, Zhang J. Inferring assertion for complementary synthesis [C/OL]. In Proceedings of the 2011 International Conference on Computer-Aided Design. San Jose, CA, USA, 2011: 404–411. http://dx.doi.org/10.1109/ICCAD.2011.6105361.

[131] Shen S, Qin Y, Wang K, et al. Inferring Assertion for Complementary Synthesis [J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2012, 31 (8): 31:1288–31:1292. http://doi.acm.org/10.1109/TCAD.2012.2190735.

[132] Liu H-Y, Chou Y-C, Lin C-H, et al. Towards completely automatic decoder synthesis [C/OL]. In Proceedings of the 2011 International Conference on Computer-Aided Design, ICCAD 2011. San Jose, CA, USA, 2011: 389–395. http://dx.doi.org/10.1109/ICCAD.2011.6105359.

[133] Liu H-Y, Chou Y-C, Lin C-H, et al. Automatic Decoder Synthesis: Methods and Case Studies [J/OL]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2012, 31 (9): 31:1319–31:1331. http://doi.acm.org/10.1109/TCAD.2012.2191288.

[134] Tu K-H, Jiang J-H R. Synthesis of feedback decoders for initialized encoders [C/OL]. In Proceedings of the 50th Annual Design Automation Conference, DAC 2013. Austin, TX, USA, 2013: 1–6. http://dx.doi.org/10.1145/2463209.2488794.

[135] Abts D, Kim J. High Performance Datacenter Networks [M/OL]. 1st ed. Morgan and Claypool, 2011: 7–9. http://dx.doi.org/10.2200/S00341ED1V01Y201103CAC014.

[136] IEEE. IEEE Standard for Ethernet SECTION FOURTH. 2012. http://standards.ieee.org/getieee802/download/802.3-2012_section4.pdf.

[137] Jackson M, Budruk R, Winkles J, et al. PCI Express Technology 3.0 [M]. Mindshare Press, 2012.

[138] Biere A, Cimatti A, Clarke E M, et al. Symbolic Model Checking without BDDs [C/OL]. In Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings. 1999: 193–207. http://dx.doi.org/10.1007/3-540-49059-0_14.

[139] Kroening D, Ouaknine J, Strichman O, et al. Linear Completeness Thresholds for Bounded Model Checking [C/OL]. In Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. 2011: 557–572. http://dx.doi.org/10.1007/978-3-642-22110-1_44.

[140] Walker R C, Dugan R. 64b/66b low-overhead coding proposal for serial links [R]. 2000.

[141] Minsky Y, Madhavapeddy A, Hickey J. Real World OCaml [M]. O'Reilly Media, 2013.

[142] ABC:A system for sequential synthesis and verification. 2008. http://www.eecs.berkeley.edu/alanmi/abc/.

[143] Widmer A X, Franaszek P A. A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code [J/OL]. IBM Journal of Research and Development. 1983, 27 (5): 440–451. http://dx.doi.org/10.1147/rd.275.0440.

[144] Shen S, Qin Y, Li S. Localizing Errors in Counterexample with Iteratively Witness Searching [C/OL] // Wang F. In Automated Technology for Verification and Analysis: Second International Conference, ATVA 2004, Taipei, Taiwan, ROC, October 31-November 3, 2004. Proceedings. 2004: 456–469. http://dx.doi.org/10.1007/978-3-540-30476-0_37.

[145] Shen S, Qin Y, Li S. Minimizing Counterexample with Unit Core Extraction and Incremental SAT [C/OL] // Cousot R. In Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005, Paris, France, January 17-19, 2005, Proceedings. 2005: 298–312. http://dx.doi.org/10.1007/978-3-540-30579-8_20.

[146] Shen S, Qin Y, Li S. A Faster Counterexample Minimization Algorithm Based on Refutation Analysis [C/OL]. In 2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany. 2005: 672–677. http://doi.ieeecomputersociety.org/10.1109/DATE.2005.14.

[147] Shen S, Qin Y, Li S. Minimizing Counterexample of ACTL Property [C/OL] // Borrione D, Paul W J. In Correct Hardware Design and Verification Methods, 13th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2005, Saarbrücken, Germany, October 3-6, 2005, Proceedings. 2005: 393–397. http://dx.doi.org/10.1007/11560548_39.

[148] Shen S, Qin Y, Li S. A fast counterexample minimization approach with refutation analysis and incremental SAT [C/OL] // Tang T. In Proceedings of the 2005 Conference on Asia South Pacific Design Automation, ASP-DAC 2005, Shanghai, China, January 18-21, 2005. 2005: 451–454. http://doi.acm.org/10.1145/1120725.1120910.

[149] Shen S, Zhang J, Qin Y, et al. Synthesizing complementary circuits automatically [C/OL] // Roychowdhury J S. In 2009 International Conference on Computer-Aided Design, ICCAD 2009, San Jose, CA, USA, November 2-5, 2009. 2009: 381–388. http://doi.acm.org/10.1145/1687399.1687472.

[150] Shen S, Qin Y, Wang K, et al. Synthesizing Complementary Circuits Automatically [J/OL]. IEEE Trans. on CAD of Integrated Circuits and Systems. 2010, 29 (8): 1191–1202. http://dx.doi.org/10.1109/TCAD.2010.2049152.

[151] Shen S, Qin Y, Zhang J. Inferring assertion for complementary synthesis [C/OL] // Phillips J R, Hu A J, Graeb H. In 2011 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2011, San Jose, California, USA, November 7-10, 2011. 2011: 404–411. http://dx.doi.org/10.1109/ICCAD.2011.6105361.

[152] Shen S, Qin Y, Xiao L, et al. A Halting Algorithm to Determine the Existence of the Decoder [J/OL]. IEEE Trans. on CAD of Integrated Circuits and Systems. 2011, 30 (10): 1556–1563. http://dx.doi.org/10.1109/TCAD.2011.2159792.

[153] Shen S, Qin Y, Wang K, et al. Inferring Assertion for Complementary Synthesis [J/OL]. IEEE Trans. on CAD of Integrated Circuits and Systems. 2012, 31 (8): 1288–1292. http://dx.doi.org/10.1109/TCAD.2012.2190735.

[154] Li C M. Integrating Equivalency Reasoning into Davis-Putnam Procedure [C/OL] // Kautz H A, Porter B W. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. 2000: 291–296. http://www.aaai.org/Library/AAAI/2000/aaai00-045.php.

[155] Ostrowski R, Grégoire É, Mazure B, et al. Recovering and Exploiting Structural Knowledge from CNF Formulas [C/OL] // Hentenryck P V. In Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings. 2002: 185–199. http://link.springer.de/link/service/series/0558/bibs/2470/24700185.htm.

[156] Karypis G, Aggarwal R, Kumar V, et al. Multilevel Hypergraph Partitioning: Application in VLSI Domain [C/OL]. In DAC. 1997: 526–529. http://doi.acm.org/10.1145/266021.266273.

[157] Achlioptas D, Gomes C P, Kautz H A, et al. Generating Satisfiable Problem Instances [C/OL] // Kautz H A, Porter B W. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. 2000: 256–261. http://www.aaai.org/Library/AAAI/2000/aaai00-039.php.

[158] Jarvisalo M. Equivalence checking hardware multiplier designs.

[159] Craig W. Linear Reasoning. A New Form of the Herbrand-Gentzen Theorem [J/OL]. J. Symb. Log. 1957, 22 (3): 250–268. http://dx.doi.org/10.2307/2963593.

[160] Du W, Goodrich M T. Searching for High-Value Rare Events with Uncheatable Grid Computing [C/OL] // Ioannidis J, Keromytis A D, Yung M. In Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings. 2005: 122–137. http://dx.doi.org/10.1007/11496137_9.

[161] Kautz H A, Porter B W. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA [C/OL]. AAAI Press / The MIT Press, 2000.

[162] Rabin T. Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings [C/OL]. Springer, 2010.

[163] Ioannidis J, Keromytis A D, Yung M. Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings [C]. 2005.

[164] Pointcheval D, Johansson T. Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings [C/OL]. Springer, 2012.

[165] Gilbert H. Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings [C/OL]. Springer, 2010.

[166] Rogaway P. Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings [C/OL]. Springer, 2011.

[167] Juels A, Wright R N, di Vimercati S D C. Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006 [C]. ACM, 2006.

作者在学期间取得的学术成果

发表的学术论文

[1] Qin Ying, Shen Shengyu, Wu Qinbo, Dai Huadong, Jia Yan. Complementary Synthesis for Encoder with Flow Control Mechanism. ACM Transaction on Design Automation of Electronic Systems(SCI 检索，CCF B 类期刊)

[2] Qin Ying, Shen Shengyu, Jia Yan. Structure-Aware CNF Obfuscation for Privacy-Preserving SAT Solving. The Proceedings of 12th ACM/IEEE International Conference on Formal Methods and Models for System Design (Memocode 2014): 84-93.（EI 检索，20145100330266）

[3] Qin Ying, Shen Shengyu, Kong Jinzhu, Dai Huadong. Cloud-oriented SAT Solver Based on Obfuscating CNF Formula. Web Technologies and Applications. The Proceedings of APweb 2014 Workshops, SNA, NIS, and IoTS: 188-199.（EI 检索，20143518106899）

[4] Qin Ying, Shen Shengyu, Jia Yan. Structure-Aware CNF Obfuscation for Privacy-Preserving SAT Solving. IEEE Trans. on CAD of Integrated Circuits and Systems.（已投稿，SCI 检索）

[5] 秦莹，沈胜宇，贾焰. 解空间投影等价的 CNF 混淆. 计算机研究与发展（已投稿，EI 检索）

[6] 秦莹，沈胜宇，贾焰. 解空间可调制的 CNF 混淆. 软件学报（已投稿，EI 检索）

[7] 秦莹，戴华东，颜跃进. 单一内核操作系统驱动程序缺陷研究. 计算机科学 2011 第 38 卷第 4 期

[8] Shen Shengyu, Qin Ying, Wang Kefei, Pang Zhengbin, Zhang Jianmin, Li Sikun: Inferring Assertion for Complementary Synthesis. IEEE Trans. on CAD of Integrated Circuits and Systems 31(8): 1288-1292 (2012)（SCI 检索，WOS：000306595100012）

[9] Shen ShengYu, Qin Ying, Xiao Liquan, Wang Kefei, Zhang Jianmin , Li Sikun: A Halting Algorithm to Determine the Existence of the Decoder. IEEE Trans. on CAD of Integrated Circuits and Systems 30(10): 1556-1563 (2011) （SCI 检索，WOS：000295099800011）

[10] ShengYu Shen, Qin Ying, Wang Kefei, Xiao Liquan, Zhang Jianmin, Li Sikun: Synthesizing Complementary Circuits Automatically. IEEE Trans. on CAD of Integrated Circuits and Systems 29(8): 1191-1202 (2010) （SCI 检索，WOS：000282543700004）

[11] ShengYu Shen, Qin Ying , Zhang Jianmin: Inferring assertion for complementary synthesis. ICCAD 2011: 404-411 （EI 检索，20120314690308）

[12] ShengYu Shen, Qin Ying , Zhang Jianmin , Li Sikun : A halting algorithm to determine the existence of decoder. FMCAD 2010: 91-99（EI 检索，20112414063648）

申请专利

[1] 秦莹，吴庆波，戴华东，孔金珠，杨沙洲、沈胜宇、谭郁松. SAT 问题求解外包中的 CNF 公式数据保护方法 (专利号：201410292502.6)

授权专利

[1] 秦莹，戴华东，吴庆波，刘晓建，孔金珠，颜跃进，董攀. 防止内存泄露和内存多次释放的内核模块内存管理方法 (专利号：201110047800.5)

[2] 秦莹，刘晓建，戴华东，孔金珠，颜跃进. 面向硬件不可恢复内存故障的内核代码软容错方法 (专利号：201110341733.8)

[3] 戴华东，吴庆波，颜跃进，朱浩，孔金珠，秦莹. 面向固态硬盘文件系统的数据页缓存方法 (专利号：201110110264.9)

[4] 董攀，易晓东，吴庆波，戴华东，颜跃进，孔金珠，刘晓建，秦莹. 一种 sun4v 架构下的虚拟机自动启动控制方法 (专利号：201210043152.0)

参与主要科研项目

[1] 国家自然科学基金“面向通讯应用的自动对偶综合研究”(项目编号：61070132）

[2] 国家高技术研究和发展计划“智能云服务与管理平台核心软件及系统”(项目编号：2013AA01A212）

[3] 国家核高基重大专项“军用服务器操作系统”(项目编号：2009ZX01040-001）

[4] 国家核高基重大专项“军用增强型操作系统”(项目编号：2011ZX01040-001)