

多智能体系统时态认知规范高效符号 模型检测的算法研究

吴立军¹⁾ 苏金树¹⁾ 苏开乐²⁾

¹⁾(国防科学技术大学计算机学院 长沙 410073)

²⁾(中山大学计算机科学与技术系 广州 510275)

摘 要 Clarke 和 McMillan 提出了利用 μ 演算和 OBDDs 符号模型检测时态逻辑的方法. 这些方法是非常有效的, 能用于验证许多具有极大状态空间的实际系统(状态个数可以超过 10^{20}). 但是, 这些方法不能检测知识逻辑. 而时态认知逻辑能更精确地描述分布式领域中系统和协议的规范. 文章首先讨论了 Kripke 结构和 μ 演算的扩展, 然后提出了利用扩展 μ 演算和 OBDDs 符号模型检测时态认知逻辑的方法.

关键词 OBDDs; μ 演算; 时态认知逻辑; 符号模型检测; 安全协议验证

中图法分类号 TP18

Symbolic Model Checking Knowledge and Time in Multi-Agent System Via Extended Mu-Calculus

WU Li-Jun¹⁾ SU Jin-Shu¹⁾ SU Kai-Le²⁾

¹⁾(School of Computer Science, National University of Defense Technology, Changsha 410073)

²⁾(Department of Computer Science and Technology, Sun Yat-Sen University, Guangzhou 510275)

Abstract Clarke and McMillan presented symbolic approaches to model check temporal logics via μ -calculus and OBDDs. These approaches are very efficient and can be applied to verify many practical systems with extremely large state spaces in excess of 1020 states. However, these approaches cannot model check knowledge logics. But temporal logics of knowledge can describe more accurately the desirable specification of systems and protocols in distributed systems. In this paper, the symbolic approaches for model checking the temporal logic of knowledge via extended μ -calculus and OBDDs are discussed mainly. First the Kripke structure and μ -calculus are extended. Then the symbolic approaches for model checking temporal logics of knowledge via extended μ -calculus and OBDDs are presented.

Keywords OBDDs; μ -calculus; temporal logics of knowledge; symbolic model checking; protocol verification

1 引 言

模型检测自 19 世纪 80 年代提出以来, 已经变

成了最成功的自动验证技术之一. 它被广泛地应用于硬件测试和通信协议及控制系统的有效性证明等诸多领域. 模型检测一直以来主要用来检验系统是否满足用时态逻辑描述的规范. 人们很少注意知识

逻辑的模型检测的问题. 然而在分布式系统中, 时态认知逻辑能更准确地描述系统和协议的规范, 因而被广泛地用于系统和协议的规范描述. 因此模型检测时态认知逻辑是一个新的重要的研究领域.

van der Meyden 和 Shilov 讨论了具有完全记忆的系统中模型检测知识和时间的问题^[1]. van der Hoek 等提出了基于命题逻辑的时态认知逻辑模型检测算法^[2]. 但它只是局限于由 LTL 和知识算子组成的逻辑, 而且他们的工作基础还需进一步研究. 苏开乐等在时态认知逻辑方面也做了许多工作^[3-4]. 吴立军等讨论并提出了基于 SMV 的时态认知逻辑模型检测方法, 但没有解决极大状态空间系统的模型检测问题^[5].

mu 演算是一种功能强大的语言, 它通过使用最大最小固定点算子来描述转移系统的性质. 许多时态和程序逻辑能转换成 mu 演算. 而 mu 演算存在高效的模型检测算法. 因此研究者们对 mu 演算在计算机辅助验证领域的应用产生了极大的兴趣.

OBDDs(Ordered Binary decision Diagrams)是布尔公式的经典表达式. 实际上它们比传统的 CNF 和 DNF 范式更加紧凑. 因此 OBDDs 被广泛应用于计算机辅助设计和验证中, 包括组合逻辑的符号验证等. 通过它我们可以验证许多具有极大状态空间的实际系统. 而这些系统的验证是状态枚举法无法实现的.

Clarke 和 McMillan 等提出了通过 mu 演算和 OBDDs 符号模型检测时态逻辑的方法^[6-7]. 这些方法具有极高的效率, 能用来验证许多具有极大状态空间的实际系统(状态个数超过 10^{20}). 然而, 这些方法不能检测认知逻辑.

本文根据以上方法和知识的语义, 利用扩展的 mu-演算和 OBDDs, 给出了高效符号模型检测时态认知逻辑的方法(由 CTL(Computation Tree Logic)和知识算子组合而成的时态认知逻辑). 这些方法能够对具有极大状态空间的系统进行安全属性的验证. 作为例子, 我们给出了 SSL2.0 的安全属性的验证.

2 Kripke 结构的扩展和 mu 演算的扩展

2.1 Kripke 结构的扩展

假设 Kripke 结构 $M = (S, T, L)$, 其中 S 是状态集合, T 是转移关系的集合, L 是映射 $L: S \rightarrow 2^{AP}$

(AP 是命题集合), 每一状态对应在该状态下为真的原子命题集合, 假设有 n 个智能体 $\text{agent } i (i=1, 2, \dots, n)$, 且 S 的任意状态 $s = (s_1, s_2, \dots, s_n)$, 其中 s_i 是 $\text{agent } i$ 的局部状态(这里环境也看做一个智能体), s 是系统的全局状态, 定义 trace 是一个有限状态序列 $u_1 u_2 \dots u_m$ (使得对所有 $0 < i < m$ 成立 $u_i R u_{i+1}, u_i \in S$), 一个 run 就是一个状态的无限序列 $r: N \rightarrow S$, 其中 r 的每一个有限前缀都是一个 trace, 设 $s = (s_1, s_2, \dots, s_n), t = (t_1, t_2, \dots, t_n)$, 如果 $s_i = t_i$, 那么就说对 $\text{agent } i, s$ 和 t 是不能分辨的(indistinguishable)^[8], 并记作 $s \sim_i t$, 令 $R_i = \{(s, t) | s \in S, t \in S, s \sim_i t\} (i=1, 2, \dots, n)$, 显然 $R_i (i=1, 2, \dots, n)$ 是与智能体 $\text{agent } i$ 有关的 S 上的等价关系. 因此 Kripke 结构 M 可扩充为 $(S, T, L, R_1, R_2, \dots, R_n)$ (即 Kripke 结构由 (S, T, L) 扩展为具有我们定义的等价关系的 Kripke 结构 $(S, T, L, R_1, R_2, \dots, R_n)$).

2.2 mu 演算的扩展

本文对 Kozen 的 mu 演算^[9]进行扩展并应用它对时态认知逻辑进行验证. 设系统的 Kripke 结构为 $M = (S, T, L, R_1, R_2, \dots, R_n), VAR = \{Q_1, Q_2, \dots\}$ 是一个关系变量集合, 对于每个关系变量 $Q \in VAR$ 指定 S 的一个子集与之对应. 扩展的 mu 演算公式可以构造如下:

如果 $p \in AP$, 那么 p 是一个公式.

如果 f 是一个关系变量, 那么 f 是一个公式.

如果 f 和 g 是公式, 那么 $\neg f, f \wedge g$ 和 $f \vee g$ 是公式.

如果 f 是一个公式, 且 $a \in T$, 那么 $[a]f$ 和 $\langle a \rangle f$ 是公式.

如果 f 是一个公式, 那么 $[R_i]f$ 是一个公式 ($i=1, 2, \dots, n$).

如果 $Q \in VAR$ 且 f 是一个公式, 那么 $\mu Q.f$ 和 $\nu Q.f$ 是公式, 这里 f 是关于 Q 单调的.

公式 $\langle a \rangle f$ 的直观意义是“构造到 f 为真的状态的一个 a -转移是可能的”. 类似的, 公式 $[a]f$ 的直观意义是“在通过 a -转移达到的所有状态, f 都为真”. 公式 $[R_i]f (i=1, 2, \dots, n)$ 的直观意义是“ f 在通过关系 R_i 的所有状态都为真”. μ 和 ν 分别表示最小和最大固定点. 没有状态的集合用 False 表示, 所有状态组成的集合用 True 表示.

形式上, 一个公式 f 被解释为使 f 真的所有状态组成的集合, 记为 $[f]_M$, 这里 M 是一个转移系统, $e: VAR \rightarrow 2^S$ 是一个环境. 我们用 $e[Q \leftarrow W]$ 表示一个新环境, 这个环境除了 $e[Q \leftarrow W](Q) = W$ 外与

e 完全相同. 因此集合 $[f]_{Me}$ 能被递归定义如下:

$$[p]_{Me} = \{s \mid p \in L(s)\}.$$

$$[Q]_{Me} = e(Q).$$

$$[\neg f]_{Me} = s \setminus [f]_{Me}.$$

$$[f \wedge g]_{Me} = [f]_{Me} \cap [g]_{Me}.$$

$$[f \vee g]_{Me} = [f]_{Me} \cup [g]_{Me}.$$

$$[\langle a \rangle f]_{Me} = \{s \mid \exists t[(s, t) \in T \text{ and } t \in [f]_{Me}]\}.$$

$$[[a]f]_{Me} = \{s \mid \forall t[(s, t) \in T \text{ implies } t \in [f]_{Me}]\}.$$

$$[[R_i]f]_{Me} = \{s \mid \forall t[(s, t) \in R_i \text{ implies } t \in [f]_{Me}]\}.$$

$[\mu Q.f]_{Me}$ 是谓词转移器 $\tau: 2^S \rightarrow 2^S$ ($\tau(w) = [f]_{Me}[Q \leftarrow W]$) 的最小固定点.

$[\nu Q.f]_{Me}$ 是谓词转移器 $\tau: 2^S \rightarrow 2^S$ ($\tau(w) = [f]_{Me}[Q \leftarrow W]$) 的最大固定点.

3 时态认知逻辑与 OBDDs

3.1 时态认知逻辑

本文研究由 CTL 和认知逻辑组成的 CKKn 逻辑^[5]. CKKn 中的公式由命题算子、路径量词、时态算子和知识算子组成, 其中命题算子有 \neg (not)、 \vee (or)、 \wedge (and)、 \rightarrow (implies)、 \leftrightarrow (if and only if), 路径量词有量词 A (for all computation paths) 和量词 E (for some computation path), 时态算子有 X (next time)、 F (eventually or in the future)、 G (always or globally)、 U (until)、 R (release), 知识算子主要有 K (know)、 C_G (common knowledge of G). CTL 有 10 种基本算子: AX 和 EX , AF 和 EF , AG 和 EG , AU 和 EU , AR 和 ER . 其中每种算子都能用 EX , EG , EU 表述^[10].

3.1.1 时态认知逻辑的语法

CKKn 由分支时态逻辑 CTL 加上知识算子 K_i 和公共知识算子 C_G 组成 (这里 G 是智能体集合). 因此 CKKn 公式 CKKn f 可定义如下:

$$\langle \text{CKKn}f \rangle ::= \text{true}$$

$$\mid p \mid * p \text{ 为原子命题 } * /$$

$$\mid \emptyset \langle \text{CKKn}f \rangle$$

$$\mid \langle \text{CKKn}f \rangle \vee \langle \text{CKKn}f \rangle$$

$$\mid EX \langle \text{CKKn}f \rangle$$

$$\mid EG \langle \text{CKKn}f \rangle$$

$$\mid E[\langle \text{CKKn}f \rangle \cup \langle \text{CKKn}f \rangle]$$

$$\mid K_i \langle \text{CKKn}f \rangle$$

$$\mid C_G \langle \text{CKKn}f \rangle$$

3.1.2 时态认知逻辑的语义

文献[10]定义了形如 $\neg f$, $f \wedge g$, $f \vee g$, EXf ,

EGf 和 $E[f \cup g]$ 的公式的语义. 我们这里进一步定义知识和公共知识的语义. 假设多智能体系统的 Kripke 结构是 $M = (S, T, L, R_1, R_2, \dots, R_n)$. 根据文献[11], 我们能定义 M 中知识和公共知识的语义如下.

$$(M, s) \models K_i \varphi \text{ iff } (M, t) \models \varphi \text{ for all } t \text{ such that } (s, t) \in R_i.$$

$$(M, s) \models C_G \varphi \text{ iff } (M, s) \models E_G^k \varphi, \text{ for } k = 1, 2, \dots.$$

其中 E_G^k 归纳定义如下: $E_G^1 \varphi$ 表示“ G 中每个智能体知道 φ ”, $E_G^{k+1} \varphi$ 表示 $E_G E_G^k \varphi$ (即表示“ G 中每个智能体知道 $E_G^k \varphi$ ”). 实际上, $C_G \varphi$ 表示“ φ 是 G 中所有智能体的公共知识”, 其语义也可用语言递归表述如下: G 中每个智能体知道 φ , 而且 G 中每个智能体知道每个智能体知道 φ , 而且 G 中每个智能体知道每个智能体知道每个智能体知道 φ ,

3.2 OBDDs

OBDDs 是布尔公式的经典描述形式^[9]. 实质上, 他们常常比传统的布尔公式描述形式 (如 conjunctive normal form 和 disjunctive normal form) 更加紧凑. 一个 OBDD 类似如一个二元决策树, 只是它的结构是有向非循环图而不是树. OBDD 有一个总的变量序, 以便从根到叶对图进行遍历.

Bryant 证明了对一个给定的布尔函数和变量顺序, 存在唯一一个 OBDD 与之对应. 给定一个布尔函数, 其 OBDD 的大小极大地依赖于变量顺序. Bryant 也描述了关于 OBDD 运算的高效算法, 例如: 给定公式 f 和 g 的 OBDD, 如何计算 $\neg f$ 和 $f \vee g$ 的 OBDD 问题. Bryant 也讨论了计算形如 $\exists v[f]$ (这里 v 是布尔变量, f 是公式) 和 $f|_{a=0} \vee f|_{a=1}$ 的 OBDD 问题. 公式 f 中的变量替代 (记作 $f\langle v \leftarrow w \rangle$) 可以用量词来实现, 即 $f\langle v \leftarrow w \rangle = \exists v[(v \leftrightarrow w) \wedge f]$.

4 通过扩展 mu 演算符号模型检测时态认知逻辑

我们将通过两步来讨论如何实现模型检测. 首先讨论如何将 CKKn 转化成扩展的 mu 演算, 然后讨论如何通过扩展的 mu 演算和 OBDDs 模型检测时态认知逻辑.

4.1 将 CKKn 转化成扩展 mu 演算

在这一节, 我们主要研究如何将 CKKn 转化成 mu 演算.

命题 1. EGf 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的

最大固定点^[10].

证明. 设状态 $s \in [EGf]_{Me}$. 因此存在一条从状态 s 开始的路径, 在这条路径上的所有状态, f 都为真. 因此 $s \in [f]_{Me}$. 假设 t 是该路径上的第二个状态. 那么显然有 $t \in [EGf]_{Me}$. 因此 $s \in [EXEGf]_{Me}$ 且 $s \in [f \wedge EXEGf]_{Me}$. 从而有 $[EGf]_{Me} \subseteq [f \wedge EXEGf]_{Me}$. 同理可证 $[f \wedge EXEGf]_{Me} \subseteq [EGf]_{Me}$. 因此 $EGf = f \wedge EXEGf$. 从而 EGf 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的固定点. 以下证明 EGf 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的最大固定点. 设 g 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的固定点且 $s \in [g]_{Me}$. 显然有 $s \in [EXg]_{Me}$ 且 $s \in [f]_{Me}$. 由以上证明可知如果 g 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的固定点且 $s \in [EXg]_{Me}$, 那么 $s \in [EGg]_{Me}$; 如果 g 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的固定点且 $s \in [EGg]_{Me}$, 那么 $s \in [EGf]_{Me}$. 从而 $[g]_{Me} \subseteq [EGf]_{Me}$. 因此 EGf 是谓词转移 $\tau(Q) = f \wedge EXQ$ 的最大固定点. 证毕.

命题 2. $E(f \cup g)$ 是谓词转移 $\tau(Q) = g \vee (f \wedge EXQ)$ 的最大固定点^[10].

命题 2 的证明类似于命题 1.

下面我们讨论如何将 $K_i f$ 和 $C_G f$ 转化成扩展 mu 演算公式.

设 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge K_i Q, i = 1, 2, \dots, n\} = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$.

命题 3. 对应 $K_i f$ 的扩展 mu 演算公式是 $[R_i]f$.

这个命题能直接从 $K_i f$ 和 $[R_i]f$ 的语义获得.

以下讨论怎样将 $C_G f$ 转化成扩展 mu 演算公式.

命题 4. 如果 $[Q_1]_{Me} \subseteq [Q_2]_{Me}$, 那么 $[E_G Q_1]_{Me} \subseteq [E_G Q_2]_{Me}$.

证明. 设 $s \in [K_i Q_1]_{Me} (i = 1, 2, \dots, n)$, 那么 $(M, s) \models K_i Q_1$. 由 $K_i Q_1$ 的语义, 对所有的 $(s, t) \in R_i$, 我们有 $(M, t) \models Q_1$. 又因为 $[Q_1]_{Me} \subseteq [Q_2]_{Me}$, 所以对于所有满足 $(s, t) \in R_i$ 的 t 有 $(M, t) \models Q_2$. 由知识的定义, 我们有 $(M, s) \models K_i Q_2$. 因此 $s \in [K_i Q_2]_{Me}$. 从而 $[K_i Q_1]_{Me} \subseteq [K_i Q_2]_{Me}, i = 1, 2, \dots, n$. 因此 $[E_G Q_1]_{Me} \subseteq [E_G Q_2]_{Me}$. 证毕.

命题 5. 谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge K_i Q, i = 1, 2, \dots, n\} = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$ 是单调的.

证明. 设 $[Q_1]_{Me} \subseteq [Q_2]_{Me}$. 由命题 4, 有 $[E_G Q_1]_{Me} \subseteq [E_G Q_2]_{Me}$. 从 $\tau(Q_1) = \{s \mid s \in S, (M, s) \models f \wedge K_i Q_1, i = 1, 2, \dots, n\} = \{s \mid s \in S, (M, s) \models f \wedge$

$E_G Q_1\}$ 和 $\tau(Q_2) = \{s \mid s \in S, (M, s) \models f \wedge K_i Q_2, i = 1, 2, \dots, n\} = \{s \mid s \in S, (M, s) \models f \wedge E_G Q_2\}$ 容易看出 $\tau(Q_1) \subseteq \tau(Q_2)$. 证毕.

命题 6. 对于任意的公式 f , 我们有 $[C_G f]_{Me} \subseteq [f]_{Me}$.

证明. 设 $s \in [C_G f]_{Me}$, 那么就有 $s \in [K_i f]_{Me} (i = 1, 2, \dots, n)$. 由知识的语义, 对所有满足 $(s, t) \in R_i$ 的 t , 有 $(M, t) \models f$. 从 $R_i (i = 1, 2, \dots, n)$ 的定义知 $R_i (i = 1, 2, \dots, n)$ 是等价关系, 所以 $(s, s) \in R_i$. 从而 $(M, s) \models f$. 从而 $s \in [f]_{Me}$, 因此 $[C_G f]_{Me} \subseteq [f]_{Me}$. 证毕.

命题 7. $C_G f$ 是谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$ 的一个固定点.

证明. 由 $C_G f$ 的语义可知 $E_G(C_G f) = C_G f$. 因此 $f \wedge E_G(C_G f) = f \wedge C_G f$. 由命题 6, $[f \wedge E_G(C_G f)]_{Me} = [f \wedge C_G f]_{Me} = [f]_{Me} \cap [C_G f]_{Me} = [C_G f]_{Me}$, 即 $[f \wedge E_G(C_G f)]_{Me} = [C_G f]_{Me}$. 因此 $C_G f$ 是谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$ 的一个固定点. 证毕.

因为谓词转移 τ 是单调的且状态空间 S 是有限的, 由文献[7], 我们知道 τ 有一个最大固定点. 下面我们证明 $C_G f$ 是谓词转移 τ 的最大固定点.

命题 8. $C_G f$ 是谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\} = \{s \mid s \in S, (M, s) \models f \wedge K_i Q, i = 1, 2, \dots, n\}$ 的最大固定点.

证明. 设 Q_1 是谓词转移 τ 的一个固定点, 那么 $[Q_1]_{Me} = [f \wedge E_G Q_1]_{Me} \subseteq [f]_{Me}$ 且 $[Q_1]_{Me} = [f \wedge E_G Q_1]_{Me} \subseteq [E_G Q_1]_{Me}$. 由命题 4 我们有 $[E_G Q_1]_{Me} \subseteq [E_G f]_{Me}$. 通过反复运用命题 4, 可得对任意 k , $[E_G^k Q_1]_{Me} \subseteq [E_G^k f]_{Me}$. 再根据 $C_G Q_1$ 和 $C_G f$ 的语义, 我们有 $[C_G Q_1]_{Me} \subseteq [C_G f]_{Me}$. 同理通过对 $[Q_1]_{Me} \subseteq [E_G Q_1]_{Me}$ 反复运用命题 4, 我们有 $[Q_1]_{Me} \subseteq [E_G Q_1]_{Me} \subseteq [E_G^2 Q_1]_{Me} \dots$. 因此对任意的 k , $[Q_1]_{Me} \subseteq [E_G^k Q_1]_{Me}$, 根据 $C_G Q_1$ 的语义, 可得 $[Q_1]_{Me} \subseteq [C_G Q_1]_{Me}$. 因此 $[Q_1]_{Me} \subseteq [C_G f]_{Me}$. 由命题 7 知 $C_G f$ 是谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$ 的一个固定点. 所以 $C_G f$ 是谓词转移 $\tau(Q) = \{s \mid s \in S, (M, s) \models f \wedge E_G Q\}$ 的最大固定点. 证毕.

由以上 8 个命题, 我们可以得到 CKKn 转化为 mu 演算的算法如图 1 所示.

```

Function CKKtomu( $f$ : CKKn formula): extended mu-formula;
Case
  F is an atomic proposition  $p$ 
    Return  $p$ ;
  F is of the form  $\neg f_1$ 
    Return  $\neg \text{CKKtomu}(f_1)$ ;
  F is of the form  $f_1 \wedge f_2$ 
    Return  $\text{CKKtomu}(f_1) \wedge \text{CKKtomu}(f_2)$ ;
  F is of the form  $f_1 \vee f_2$ 
    Return  $\text{CKKtomu}(f_1) \vee \text{CKKtomu}(f_2)$ ;
  F is of the form  $\text{EX}f$ 
    Return  $\langle a \rangle \text{CKKtomu}(f)$ ;
  F is of the form  $\text{EG}f$ 
    Return  $\nu Q.(\text{CKKtomu}(f) \wedge \langle a \rangle Q)$ ;
  F is of the form  $E(f \cup g)$ 
    Return  $\mu Q.(\text{CKKtomu}(g) \vee (\text{CKKtomu}(f) \wedge \langle a \rangle Q))$ ;
  F is of the form  $K_i f (i=1, 2, \dots, n)$ 
    Return  $[R_i] \text{CKKtomu}(f)$ ;
  F is of the form  $C_g f$ 
    Return  $\nu Q.(\text{CKKtomu}(f) \wedge (\bigwedge_{i=1}^n [R_i] Q))$ ;
End case
End function

```

图 1 CKKn 转化为 mu 演算的算法

4.2 通过 mu 演算和 OBDDs 符号模型检测时态认知逻辑

模型检测是判定在一个给定模型 M 中给定公式 f 是否为真的过程. 在本节, 我们将讨论如何通

过 mu 演算和 OBDDs 对时态认知逻辑进行高效模型检测. 首先我们讨论如何将扩展的 Kripke 结构 $M=(S, T, L, R_1, R_2, \dots, R_n)$ 编码成 OBDDs. 这里的编码类似于文献[10]中 5.2 节提出的 Kripke 结构的编码. 域 S 编码成 n 个布尔变量 x_1, x_2, \dots, x_n 的值的集合, 即 S 编码成长度为 n 的布尔向量空间. 有时候, 布尔变量 x_1, x_2, \dots, x_n 用向量符号 \mathbf{x} 来表示. 给定一个解释, 我们按以下方法来产生闭扩展 mu 演算公式的 OBDDs:

(1) 每个原子命题 p 有一个 OBDD 与之相关, 记作 $\text{OBDD}_p(\mathbf{x})$. 一个布尔向量 $\mathbf{y} \in \{0, 1\}^n$ 满足 OBDD_p 的充分必要条件是 $\mathbf{y} \in L(p)$.

(2) 转移关系 T 有一个 OBDD 与之相关, 记作 $\text{OBDD}_T(\mathbf{x}, \mathbf{x}')$. 一个布尔向量 $(\mathbf{y}, \mathbf{z}) \in \{0, 1\}^{2n}$ 满足 OBDD_T 的充分必要条件是 $(\mathbf{y}, \mathbf{z}) \in T$.

(3) 每一个等价关系 $R_i (i=1, 2, \dots, n)$ 有一个 OBDD 与之相关, 记作 $\text{OBDD}_{R_i}(\mathbf{x}, \mathbf{x}')$. 一个布尔向量 $(\mathbf{y}, \mathbf{z}) \in \{0, 1\}^{2n}$ 满足 OBDD_{R_i} 的充分必要条件是 $(\mathbf{y}, \mathbf{z}) \in R_i$.

假设 f 是一个扩展 mu 演算公式, Q_1, Q_2, \dots ,

```

Function OBDDF( $f$ : extended mu-formula, assoc: association list): OBDD;
Case
  F is an atomic proposition  $p$ 
    Return  $\text{OBDD}_p$ ;
  F is an relation variable  $Q$ 
    Return  $\text{assoc}[Q]$ ;
  F is of the form  $\neg f_1$ 
    Return  $\text{OBDD-NEGATE}(\text{OBDDF}(f_1))$ ;
  F is of the form  $f_1 \wedge f_2$ 
    Return  $\text{OBDD-AND}(\text{OBDDF}(f_1), \text{OBDDF}(f_2))$ ;
  F is of the form  $f_1 \vee f_2$ 
    Return  $\text{OBDD-OR}(\text{OBDDF}(f_1), \text{OBDDF}(f_2))$ ;
  F is of the form  $\langle a \rangle f$ 
    Return  $\exists \mathbf{x}' \text{OBDD-AND}(\text{OBDD}_T(\mathbf{x}, \mathbf{x}'), \text{OBDDF}(f, \text{assoc})(\mathbf{x}'))$ ; (here,  $\text{OBDDF}(f, \text{assoc})(\mathbf{x}')$  is the OBDD in which each Boolean variable  $x_i$  is replaced by its primed person  $x'_i$ )
  F is of the form  $[a]f$ 
    Return  $\text{OBDDF}(\neg \langle a \rangle \neg f, \text{assoc})$ ;
  F is of the form  $[R_i]f (i=1, 2, \dots, n)$ 
    Return  $\forall \mathbf{x}' (\text{OBDD-AND}(\text{OBDD}_{R_i}(\mathbf{x}, \mathbf{x}'), \text{OBDDF}(f, \text{assoc})(\mathbf{x}')))$ ; (here,  $\text{OBDDF}(f, \text{assoc})(\mathbf{x}')$  is the OBDD in which each Boolean variable  $x_i$  is replaced by its primed person  $x'_i$ )
  F is of the form  $\mu Q.f$ 
    Return  $\text{fixedpoint}(f, \text{assoc}, \text{False-OBDD})$ ;
  F is of the form  $\nu Q.f$ 
    Return  $\text{fixedpoint}(f, \text{assoc}, \text{True-OBDD})$ ;
End case
End function

Function fixedpoint( $f$ : extended mu-formula, assoc: association list, Obdd:OBDD)
  Resultobdd := Obdd;
  Oldobdd :=  $\text{OBDDF}(f, \text{assoc}(Q \leftarrow \text{Resultobdd}))$ ;
  While (Resultobdd  $\neq$  Oldobdd)
    Resultobdd := Oldobdd;
    Oldobdd :=  $\text{OBDDF}(f, \text{assoc}(Q \leftarrow \text{Resultobdd}))$ ;
  End while;
  Return (resultobdd);
End function

```

图 2 通过扩展 mu 演算和 OBDDs 模式检测时态认知逻辑的算法

Q_k 是 f 的自由关系变量, $\text{assoc}[Q_i]$ 给出对应于与关系变量 Q_i 相关的状态集合的 OBDD. 通过增加一个关系变量 Q 和关联一个与之对应的 OBDD B_Q , $\text{assoc}(Q \leftarrow B_Q)$ 产生一个新的关联. 现在我们描述通过扩展 mu 演算和 OBDDs 模型检测时态认知逻辑的算法. 算法主要由一个函数 $\text{OBDDF}()$ 组成, $\text{OBDDF}()$ 有两个参数 f (扩展 mu 演算公式) 和 assoc (关联列表), 返回一个与 f 的语义对应的 OBDD (在系统 M 中满足 f 的状态集合的 OBDD). 函数 $\text{OBDDF}()$ 中的第 1、第 2、第 6 和第 7 种情形, 从 OBDD 和扩展 mu 演算公式的语义定义可以直接获得. 第 8 种情形, 从扩展 mu 演算公式 $[R_i]f (i=1, 2, \dots, n)$ 和知识的语义定义可以直接获得. 子函数 $\text{OBDD-NEGATE}()$, $\text{OBDD-AND}()$ 和 $\text{OBDD-OR}()$, 已在文献[10]中由 Bryant 进行了描述. False-OBDD 表示对应 False 的 OBDD, True-OBDD 表示对应 True 的 OBDD. 子函数 $\text{fixedpoint}()$ 与文献[7]中的类似. 算法描述如图 2 所示.

现在考虑算法的复杂性. 假设系统的 Kripke 结构为 $M=(S, T, L, R_1, R_2, \dots, R_n)$.

命题 9. 通过扩展 mu 演算和 OBDDs 模式检测时态认知逻辑的算法的时间复杂性是 $O(n^k)$, 这里 $n=|S|$, k 是被检测公式中固定点算子的最大嵌套深度. 因此当公式中固定点算子的最大嵌套深度一定时, 算法的时间复杂性相对于系统状态数是多项式时间的.

命题 9 的证明由文献[10]加以扩展后很容易获得.

本文在文献[7]和[10]的方法的基础上, 扩展了 Kripke 结构和 mu 演算, 并通过扩展 mu 演算和 OBDDs, 提出了模型检测时态认知逻辑的方法, 使得文献[7]和[10]的方法从高效模型检测时态逻辑扩展到模型检测时态认知逻辑. 从本文第 5 节和文献[7]和[10]不难看出, 算法在实际中是非常有效的, 能够用于验证具有极大状态空间的系统(如状态数超过 10^{20}).

5 实 例

我们用 SSL 2.0 (Secure Sockets Layer) 握手协议作为实例来进行分析.

5.1 SSL 2.0 握手协议

SSL 2.0 协议包括 SSL 2.0 记录协议和 SSL 2.0 握手协议. 本节主要使用我们的方法模型检测

SSL 2.0 握手协议. 该协议可以描述如下^[11]:

```

 $C \rightarrow S: C, SuiteC, N_C;$ 
 $S \rightarrow C: SuiteS, N_S, \text{sign}_{CA}\{S, K_S^+\};$ 
 $C \rightarrow S: \{SecretC\}_{K_S^+};$ 
 $C \rightarrow S: \{N_S\}_{\text{Master}(SecretC)};$ 
 $S \rightarrow C: \{N_C\}_{\text{Master}(SecretC)};$ 
 $S \rightarrow C: \{SessionId\}_{\text{Master}(SecretC)}.$ 

```

这里 C 是客户端, S 是服务器, $SuiteC$ 和 $SuiteS$ 分别是 C 和 S 的加密爱好, N_C 和 N_S 分别是由 C 和 S 发布的具有新鲜性的随机数. K_S^+ 是 S 的公钥, $SecretC$ 是由 C 产生的随机秘密, $SessionId$ 是特定会话的标识符, $\text{Master}(SecretC)$ 是从 $SecretC$ 产生的主秘密(密钥). $\text{sign}_{CA}\{S, K_S^+\}$ 是 S 的数字签名. 协议运行的目的是在 C 和 S 之间建立一个密钥. 这个密钥用于今后秘密通信.

5.2 系统模型

我们假设有一个攻击者 I . 那么客户端 C 能观察到的可能信息包括:

三方 C, S, I 的名字—— C, S, I ;

三方 C, S, I 的加密爱好—— $SuiteC, SuiteS, SuiteI$;

三方 C, S, I 的 Nonces (具有新鲜性的随机数)—— N_C, N_S, N_I ;

三方 C, S, I 的随机秘密—— $SecretC, SecretS, SecretI$;

三方 C, S, I 的主秘密—— $\text{Master}(SecretC), \text{Master}(SecretS), \text{Master}(SecretI)$;

三方 C, S, I 的公钥—— K_C^+, K_S^+, K_I^+ ;

三方 C, S, I 的私钥—— K_C^-, K_S^-, K_I^- ;

三方 C, S, I 的数字签名—— $\text{sign}_{CA}\{C, K_C^+\}, \text{sign}_{CA}\{S, K_S^+\}, \text{sign}_{CA}\{I, K_I^+\}$;

特定会话的标识符—— $SessionId$;

我们由此可知客户端 C 有 25 种可能信息. 因此在客户端 C 的可能状态集中有 2^{25} 个可能的局部状态.

同理, 服务器 S 有 25 种可能信息. 因此在服务器 S 的可能状态集中有 2^{25} 个可能的局部状态. 攻击者 I 有 25 种可能信息. 因此在攻击者 I 的可能状态集中有 2^{25} 个可能的局部状态.

因此系统的全局状态集 ($S_A \times S_B \times S_I$) 包含 $2^{25} \times 2^{25} \times 2^{25}$ 个可能状态, 即 2^{75} 个状态. 每个全局状态能被表示为 $s = (s_c, s_s, s_i)$. 当然实际状态数要少得多.

现在我们部分地描述状态转移关系 R . 在客户

端 C 发送信息 $\{C, SuiteC, N_C\}$ 之前, C 的局部状态是 $s_{C1} = \{K_C^-\}$, 服务器 S 的局部状态是 $s_{S1} = \{K_S^-\}$, 攻击者 I 的局部状态是 $s_{I1} = \{K_I^-\}$, 因此系统的全局状态是 (s_{A1}, s_{B1}, s_{I1}) ; 在客户端 C 发送信息 $\{C, SuiteC, N_C\}$ 之后, C 的局部状态是 $\{K_C^-, C, SuiteC, N_C\}$, 服务器 S 的局部状态是 $s_{S2} = \{K_S^-, C, SuiteC, N_C\}$, 攻击者 I 的局部状态是 $s_{I2} = \{K_I^-, C, SuiteC, N_C\}$, 因此系统的全局状态是 (s_{A2}, s_{B2}, s_{I2}) ; 因此系统的全局状态由 (s_{A1}, s_{B1}, s_{I1}) 转移到 (s_{A2}, s_{B2}, s_{I2}) .

5.3 系统规范

在文献[11]的基础上, 我们用时态认知逻辑来更加准确地描述 SSL2.0 协议应该满足的安全属性.

(1) 存在一条运行路径, 在此路径上, “客户端 C 和服务器 S 之间的共享秘密是 $SecretC$ ” 终究是客户端 C 和服务器 S 之间的公共知识.

(2) 对于所有的运行路径, 攻击者 I 总是不知道“客户端 C 和服务器 S 之间的共享秘密是 $SecretC$ ”.

(3) 由各方选择的用于加密和握手信息认证的密码算法是被 C 和 S 支持的算法中最强的, 我们可以通过以下方式模型化它: 要求由 S 存储且用做 C 的密码爱好的 $SuiteC$ 与实际由 C 发送的密码爱好完全一致, 反过来也如此. 因此“ $SuiteC$ 是 C 的密码爱好”和“ $SuiteS$ 是 S 的密码爱好”终究是客户端 C 和服务器 S 之间的公共知识.

我们用 f 表示“客户端 C 和服务器 S 之间的共享秘密是 $SecretC$ ”. 用 $C_G f$ (这里 $G = \{\text{客户端 } C, \text{服务器 } S\}$) 表示“ f 是客户端 C 和服务器 S 之间的公共知识”, 用 g 表示“ $SuiteC$ 是 C 的密码爱好”, 用 h 表示“ $SuiteS$ 是 S 的密码爱好”, 用 $C_G g$ 表示“ g 是客户端 C 和服务器 S 之间的公共知识”, 用 $C_G h$ 表示“ h 是客户端 C 和服务器 S 之间的公共知识”. 那么第 1、第 3 个性质能表示为 $EF(C_G(f \wedge g \wedge h))$, 第 2 个性质能表示为 $AG \neg K_I f$. 因此系统规范能表示为 $EF(C_G(f \wedge g \wedge h)) \wedge AG \neg K_I f$.

5.4 符号模型检测系统规范

$EF(C_G(f \wedge g \wedge h))$ 可以被转化成扩展 μ 演算公式 $\mu Z. (\nu Q. (f \wedge g \wedge h \wedge [R_C]Q \wedge [R_S]Q) \vee \langle a \rangle Z)$, $AG \neg K_I f$ 可以被转化成扩展 μ 演算公式 $\neg \mu Z. ([R_I]f) \vee \langle a \rangle Z$. 因此系统规范 $EF(C_G(f \wedge g \wedge h)) \wedge AG \neg K_I f$ 能被转化为扩展 μ 演算公式 $\mu Z. (\nu Q. (f \wedge g \wedge h \wedge [R_C]Q \wedge [R_S]Q) \vee \langle a \rangle Z) \wedge (\neg \mu Z. ([R_I]f) \vee \langle a \rangle Z)$. 根据上面提出的方法, 我们验证了系统(SSL2.0 协议)不满足规范 $EF(C_G(f \wedge g \wedge h))$,

并且找到了下面一个关于密码爱好的攻击:

$$\begin{aligned} C &\rightarrow S(I): C, SuiteC, N_C; \\ I &\rightarrow S: C, SuiteI, N_C; \\ S &\rightarrow C(I): SuiteS, N_S, \text{sign}_{CA}\{S, K_S^+\}; \\ I &\rightarrow C: SuiteI, N_S, \text{sign}_{CA}\{S, K_S^+\}; \\ C &\rightarrow S: \{SecretC\}_{K_S^+}; \\ C &\rightarrow S: \{N_S\}_{\text{Master}(SecretC)}; \\ S &\rightarrow C: \{N_C\}_{\text{Master}(SecretC)}; \\ S &\rightarrow C: \{SessionId\}_{\text{Master}(SecretC)}. \end{aligned}$$

攻击者 I 通过修改 C 的密码爱好, 强迫 C 和 S 选用比通常较弱的加密算法或签名算法. 这可能使得攻击者 I 更容易对 C 的秘密交换信息进行解密.

6 结束语

本文在文献[7, 9-10]的基础上, 根据知识和公共知识的语义, 通过使用扩展 μ 演算和 OBDDs, 提出了一种高效的模型检测时态认知逻辑的方法. 首先, 我们扩展 Kripke 结构和 μ 演算, 然后研究怎样将知识和公共知识转化成扩展 μ 演算公式, 然后通过扩展 μ 演算和 OBDDs, 模型检测时态认知逻辑. 最后我们用 SSL2.0 协议作为实例讨论我们方法在安全协议上的优点:

(1) 能用符号模型检测有知识和公共知识算子的规范.

(2) 因为使用了扩展 μ 演算公式和 OBDDs, 算法如文献[7]的算法一样高效.

目前我们正在编写相应的软件, 以便实现完全的自动化验证.

参 考 文 献

- [1] van der Meyden R, Shilov N V. Model checking knowledge and time in systems with perfect recall (extended abstract)// Goos C C ed. Proceedings of the Foundations of Software Technology and Theoretical Computer Science(LNCS 1738). Berlin: Springer-Verlag, 1999: 432-445
- [2] van der Hoek W, Wooldridge M. Model checking knowledge and time//Stefan Leue C C ed. Proceedings of the 9th International SPIN Workshop on Model Checking of Software. Grenoble, France, 2002: 1-16
- [3] van der Meyden, Su K L. Symbolic model checking the knowledge of the dining cryptographers//Proceedings of the 17th IEEE Security Foundation Workshop. Asilomar, CA, USA, 2004: 280-291

[4]

Su K L. Model checking temporal logics of knowledge in distributed systems//Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04). San Jose, CA, USA, 2004: 200-207

[5]

Wu Li-Jun, Su Kai-Le. A model checking algorithm for temporal logics of knowledge in multi-agent systems. Journal of Software, 2004, 15(7): 1012-1020(in Chinese)
(吴立军,苏开乐.多智能体系统时态认知规范的模型检测算法.软件学报,2004,15(7):1012-1020)

[6]

Clarke E M, Grumberg O, Peled D A. Model Checking. 1st Edition. Cambridge: MIT Press, 1999

[7]

Burch J R, Clarke E M, McMillan K L. Symbolic model checking: 10^{20} states and beyond. Information and Computation, 1998, 98(2): 142-170

[8]

Fagin R, Halpern J Y, Moses Y, Vardi M Y. Reasoning about Knowledge. 1st Edition. Cambridge: MIT Press, 1995: 111-120

[9]

Kozen D. Results on the prepositional mu-calculus. Theoretical Computer Science, 1983, 27: 333-354

[10]

Bryant. Graph-based algorithms for boolean function manipulation. IEEE Transaction on Computers, 1986, 35(8): 687-691

[11]

Mitchell J C, Shmatikov V, Stern U. Finite-state analysis of SSL 3.0//Proceedings of the 7th USENIX Security Symposium. San Antonio TX, USA, 1998: 201-215



WU Li-Jun, born in 1965, Ph. D. , associate professor. His main research interests include formal methods and artificial intelligence.

SU Jin-Shu, born in 1962, professor, Ph. D supervisor. His main research interest is network security.

SU Kai-Le, born in 1964, professor, Ph. D supervisor. His main research interests include formal methods and artificial intelligence.

Background

Model checking has been used mainly to check if a system satisfies the specifications expressed in temporal logic. People pay little attention to the problem of model checking logics of knowledge. However, in the distributed systems, the desirable specifications of systems and protocols have been expressed widely in the temporal logics of knowledge. Temporal logics of knowledge can express more accurately the security properties of systems and protocols. Therefore, model checking temporal logics of knowledge is a new important research domain. But at present, people have not solved the state-explosion problem in domain of model checking temporal logics of knowledge. The paper, by extending the mu-calculus of Kozen and OBDDs, presents a efficient algorithm of symbolic model checking temporal logics of knowledge in multi-agent systems. The approaches can handle se-

curity verification of systems with extremely large state spaces an so solves state-explosion problem.

The research is supported by Supported by the National Natural Science Foundation of China under grant Nos. 90604006, 60496327 and German Research Foundation under grant No. 446 CHV113/240/0-1. The mission of these projects is to research the model checking of knowledge logics and its application in network security. The work of this paper is encouraged by the background and considered as a significant part of the mission. The research group has written a series of high-quality papers in the domain published by some good journals and internal conferences. The work of this paper is to solve security verification of systems with extremely large state spaces.