多智体系统时态认知规范的模型检测算法*

吴立军[1], 苏开乐[1,2+]

[1](中山大学 计算机科学与技术系,广东 广州 510275)

[2](河南科技大学 电子信息工程学院,河南 洛阳 471003)

A Model Checking Algorithm for Temporal Logics of Knowledge in Multi-Agent Systems

WU Li-Jun[1], SU Kai-Le[1,2+]

[1](Department of Computer Science and Technology, Zhongshan University, Guangzhou 510275, China)

[2](Institute of Electronics and Information Engineering, He'nan University of Science and Technology, Luoyang 471003, China)

+ Corresponding author: Phn: +86-20-87233631, Fax: +86-20-87647600, E-mail: gdgmjsjx@163.com, http://www.zsu.edu.cn

Received 2003-09-01; Accepted 2004-01-06

Wu LJ, Su KL. A model checking algorithm for temporal logics of knowledge in multi-Agent systems. *Journal of Software*, 2004,15(7):1012~1020.

http://www.jos.org.cn/1000-9825/15/1012.htm

Abstract: Model checking has being used mainly to check if a system satisfies the specifications expressed in temporal logic and people pay little attention to the model checking problem for logics of knowledge. However, in the distributed systems community, the desirable specifications of systems and protocols have been expressed widely in logics of knowledge. In this paper, the model checking approaches for the temporal logic of knowledge are discussed. On the base of SMV (symbolic model verifier), according to the semantics of knowledge and set theory, several approaches for model checking of knowledge and common knowledge are presented. These approaches make SMV's functions extended from temporal logics to temporal logics of knowledge. They also correspond to other model checking approaches and tools where the output is the set of states.

Key words: symbolic model checking; multi-Agent system; verification of protocol; SMV; TMN cryptographic protocol

摘 要: 模型检测技术一直以来主要是检验用时态逻辑描述的规范,人们很少注意认知逻辑的模型检测问题,而在分布式系统领域,系统和协议的规范已广泛地采用知识逻辑来描述.着重研讨了时态认知逻辑的模型检测算法.在 SMV(symbolic model verifier)模型检测器的基础上,根据知识的语义和集合理论,提出了多种检验知识和公共知识的算法,从而使 SMV 的检测功能由时态逻辑扩充到时态认知逻辑.这些方法也适用于其他以状态集合作为输出的模型检测方法和工具的功能扩充.

* Supported by the National Natural Science Foundation of China under Grant No.60073056 (国家自然科学基金) ; the Natural Science Foundation of Guangdong Province of China under Grant No.001174 (广东省自然科学基金)

作者简介: 吴立军(1965－),男,湖南岳阳人,博士生,主要研究领域为智体与网络安全;苏开乐(1964－),男,教授,博士生导师,主要研究领域为智体与网络安全.

关键词: 符号模型检测;多智体系统;协议验证;SMV;TMN 密码协议
中图法分类号: TP18 **文献标识码**: A

模型检测是一种检测有限状态并发系统的自动化检测技术,它主要是检测用时态逻辑描述的规范.如SMV(symbolic model verifier)检测分支时态逻辑[1]、SPIN 检测线性时态逻辑[2,3],人们很少注意时态认知逻辑的模型检测问题.然而由时态逻辑和知识逻辑组合的时态认知逻辑已被广泛应用于分布式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比较新的重要课题.Ron van der Meyden 和 N.V. Shilov 研究了具有完全记忆的系统中知识和时间的模型检测问题,并且证明了该问题在最好情况下是 PSPACE-完全的,最坏的情况是不可判定的[4].Wiebe van der Hoek 等人提出了基于局部命题思想的算法对时态认知逻辑表述的规范进行检测,然而只限于线性时态逻辑和知识逻辑组合的时态认知逻辑 CKLn,而且他们工作的理论基础还须进一步加以研究[5].

本文给出了完全不同的模型检测时态认知逻辑方法.在 SMV 模型检测器的基础上,根据知识的语义和集合理论,进一步检测知识算子和公共知识算子表述的规范,从而使 SMV 的检测功能从分支时态逻辑扩充到时态认知逻辑.本文的各种算法都经过严格的证明,算法的复杂性都是多项式时间的.我们已将各种算法编码并与 SMV 结合,对一些协议验证了用时态认知公式表述的规范(见本文后面的实例).

我们准备今后将我们的代码加入到 SMV 中,形成一种新的工具和描述语言.这种工具不但能够检测时态逻辑表述的规范,而且能够检测时态认知逻辑表述的规范,从而更好地用于协议验证.我们也将用这种新工具验证更多的协议.

1 符号模型检测、SMV 和时态认知逻辑

符号模型算检测是以 OBDD 为工具、根据不动点理论计算出满足时态逻辑描述的规范的状态集合.

SMV 是在 1987 年秋天由卡内基·梅隆大学在读博士生 McMillan 研发的模型检测系统[6].它的主要思想就是采用符号模型算法检验系统是否满足用分支时态逻辑描述的规范.用此技术,人们发现了许多用 IEEE Futurebus+标准(IEEE Standard 896,1-19991)描述的 Futurebus+Cache 一致性协议中未发现过的以及潜在的错误,检验的基本步骤如下:

(1) 将系统形式化为 $M=(S,R,L)$.

(2) 给出系统要满足的规范,即 CTL(computation tree logic)时态逻辑公式.

(3) 用 SMV 检验系统是否满足规范.如果满足就输出 true,否则就输出 false 和反例.加入某个参数后也可以输出满足规范φ的状态集合 $T,T=\{s \mid s\in S,M,s \mid =\varphi\}$.

SMV 具有比较好的检测效率,但是它只能检测用分支时态逻辑 CTL 表示的规范φ,而对于形如 $K_i\varphi$,$K_jK_i\varphi$和 $C_G\varphi$的规范就无能为力了.我们在 SMV 输出的状态集合 T 的基础上进一步检测知识和公共知识,从而实现了规范 $K_i\varphi$,$K_jK_i\varphi$和 $C_G\varphi$的模型检测.当然,今后我们会将我们的代码加入到 SMV 中形成一种新的工具,这种工具不但能够检测时态逻辑表述的规范,而且能够检测时态认知逻辑表述的规范,从而更好地用于协议验证.

时态认知逻辑是由时态逻辑和认知逻辑组合而成的逻辑,在基于 LTL(linear temporal logic)时态认知逻辑方面已有部分研究成果.比如,Wiebe van der Hoek 等人检测的是由线性时态逻辑 LTL 和认知逻辑组合的逻辑(Halpern and Vardi's logic CKLn[7]),其方法是通过引入局部命题将 CKLn 模型检测问题转化为 LTL 模型检测问题,然后利用现有的 LTL 模型检测方法和工具来完成时态认知逻辑 CKLn 的模型检测.CKLn 公式集 wff 定义如下[5]:

$\langle wff\rangle$::=true
| p /* primitive propositions */
| $\neg\langle wff\rangle$ /* negation */
| $\langle wff\rangle\vee\langle wff\rangle$ /* disjunction */

| ○⟨*wff*⟩ /* next */
| ⟨*wff*⟩ U ⟨*wff*⟩ /* until */
| K_i⟨*wff*⟩ /* agent *i* knows */
| C_G⟨*wff*⟩ /* it is common knowledge in *G* that */

本文研究的主要是由分支时态逻辑 CTL 和认知逻辑组合的逻辑 CKC*n*,在利用 SMV 模型检测器的基础上,根据知识语义和集合理论,进一步检测知识算子和公共知识算子表述的规范,从而完成时态认知逻辑 CKC*n* 的模型检测.

CKC*n* 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¬(not),∨(or),∧(and),→(implies),↔(if and only if),路径量词有量词 A(for all computation paths)和量词 E(for some computation path),时态算子有算子 X(next time)、算子 F(eventually or in the future)、算子 G(always or globally)、算子 U(until)、算子 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[6],因此 CTL 公式 *CTLf* 可定义如下:

⟨*CTLf*⟩::=true
| *p* /* *p* 为原子命题 */
| ¬⟨*CTLf*⟩
| ⟨*CTLf*⟩∨⟨*CTLf*⟩
| EX⟨*CTLf*⟩
| EG⟨*CTLf*⟩
| E[⟨*CTLf*⟩U⟨*CTLf*⟩]

而 CKC*n* 是由分支时态逻辑 CTL 加上知识算子 K_i 和公共知识算子 C_G 组成的(这里 *G* 是智体集合).因此 CKC*n* 公式 *CKCnf* 可定义如下:

⟨*CKCnf*⟩::=true
| *p* /* *p* 为原子命题 */
| ¬⟨*CKCnf*⟩
| ⟨*CKCnf*⟩∨⟨*CKCnf*⟩
| EX⟨*CKCnf*⟩
| EG⟨*CKCnf*⟩
| E[⟨*CKCnf*⟩U⟨*CKCnf*⟩]
| K_i⟨*CKCnf*⟩
| C_G⟨*CKCnf*⟩

由于 SMV 已经考虑了用 EX,EG,EU 表述的公式的模型检测,因此我们以下主要考虑 $K_i\varphi$和 $C_G\varphi$ (其中φ是 CTL 公式)的模型检测问题.

2 基于 SMV 的多智体时态认知逻辑模型检测方法

假设 Kripke 结构 $M=(S,R,L)$,其中 *S* 是状态集合,*R* 是转移关系的集合,*L* 是映射 $L:S\to 2^{AP}$(*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* 个智体 agent *i*(i=1,2,…,*n*),*S* 的任意状态 $s=(s_1,s_2,\ldots,s_n)$,其中 s_i 是 agent *i* 的局部状态(这里环境也看做一个智体),*s* 是系统的全局状态,定义 trace 是一个有限状态序列 $u_1u_2\ldots u_m$(使得对所有 $0<i<m$ 成立 $u_iRu_{i+1},u_i\in S$)[5],一个 run 就是一个状态的无限序列 *r*: $N\to S$,其中 *r* 的每一个有限前缀都是一个 trace.设 $s=(s_1,s_2,\ldots,s_n)$,$t=(t_1,t_2,\ldots,t_n)$,如果 $s_i=t_i$,那么就说对 agent *i*,*s* 和 *t* 是不能分辨的(indistinguishable)[8],并记作 $s\sim_i t$,令 $K_i=\{(s,t)|s\in S,t\in S,s\sim_i t\}$($i$=1,2,…,*n*),显然 K_i(i=1,2,…,*n*)是与智体 agent *i* 有关的 *S* 上的等价关系.因此 Kripke 结构 *M* 可扩充为$(S,R,L,K_1,\ldots,K_n)$.为了方便起见,仍然用 *M* 表示扩充后的结构,以下如无特别声明,*M* 即指扩充后的结构.依照文献[9]中多智体系统知识和公共知识的语义,我们可以定义 *M* 中知识和公共知识的语义如下:

$(M,s)|=K_i\varphi$ iff $(M,t)|=\varphi$ for all t such that $(s,t)\in K_i$.

$(M,s)|=C_G\varphi$ iff $(M,s)|=E_G^k\varphi$, for k=1,2,….

其中 E_G^k 归纳定义如下:$E_G^1\varphi$表示"G 中每个智体知道φ",$E_G^{k+1}\varphi$表示 $E_GE_G^k\varphi$(即表示"G 中每个智体知道 $E_G^k\varphi$").

对任意状态 $s\in S$ 和 agent i,定义 agent i 在状态 s 的直达状态集 $T(s,i)=\{t|t\in S,(s,t)\in K_i\}$.对于直达状态集 $T(s,i)$,我们有下面的命题:

命题 1. 如果 $s'\in T(s,i)$,且 $K_i(i=1,\ldots,n)$是等价关系,那么 $T(s',i)=T(s,i)$.

证明:如果 $t\in T(s',i)$,那么,$(s',t)\in K_i$,又因为 $s'\in T(s,i)$,所以$(s,s')\in K_i$,由于 K_i 是等价关系,所以$(s,t)\in K_i$,故 $t\in T(s,i)$,即 $T(s',i)\subseteq T(s,i)$.同理可证 $T(s',i)\supseteq T(s,i)$.故 $T(s',i)=T(s,i)$. □

对于任意分支时态逻辑 CTL 公式φ,我们都可以用 SMV 模型检测工具来检测系统是否满足这个规范φ,并求出满足这个规范φ的状态集合,这一节主要是在此基础上探讨时态认知逻辑规范(形如 $K_i\varphi$,$K_jK_i\varphi$和 $C_G\varphi$的公式)的模型检测问题.下面的研究如无特别声明都假定 $K_i(i=1,\ldots,n)$是等价关系.

2.1 $K_i\varphi$的模型检测

设 $T_{k_i}=\{s|s\in S,(M,s)|=K_i\varphi\}$,对于 T_{k_i} 和 T 我们有如下命题:

命题 2. 对任意 CTL 公式φ与 agent i,如果 $K_i(i=1,\ldots,n)$是等价关系,那么 $T_{k_i}\subseteq T$.

证明:设 $s\in T_{k_i}$,那么$(M,s)|=K_i\varphi$,由定义可得:对所有 s'(满足$(s,s')\in K_i$),有$(M,s')|=\varphi$,因为 K_i 是等价关系,所以$(s,s)\in K_i$,故$(M,s)|=\varphi$,所以 $s\in T$.故 $T_{k_i}\subseteq T$. □

我们要检测系统 M 是否满足用时态认知逻辑表述的规范 $K_i\varphi$,就是要求 $T_{k_i}=\{s|s\in S,(M,s)|=K_i\varphi\}$.由命题 2 可知,可以在 T 中进一步求 T_{k_i},因此,$K_i\varphi$的模型检测问题可以分为两大步:

(1) 用 SMV 模型检测工具求得满足φ(用时态逻辑 CTL 公式表示的规范)的状态集 T.

(2) 在 T 中求得满足规范 $K_i\varphi$的状态集 T_{k_i}.

下面讨论如何求 T_{k_i} 的问题.

命题 3. 对任意 $s\in T$,且 $K_i(i=1,\ldots,n)$是等价关系,那么 $T(s,i)\subseteq T_{k_i}$ 的充要条件是 $T(s,i)\subseteq T$.

证明:充分性:对任意 $s\in T$,因为 $T(s,i)=\{t|t\in S,(s,t)\in K_i\}$,$T(s,i)\subseteq T$,所以对每一 $t\in T(s,i)$,有$(s,t)\in K_i$,$t\in T$,从而有$(M,t)|=\varphi$,由 $K_i\varphi$的语义知$(M,s)|=K_i\varphi$,即 $s\in T_{k_i}$.此时对任意 $s'\in T(s,i)$,有 $s'\in T$,由命题 1 知,$T(s',i)=T(s,i)$.故 $T(s',i)\subseteq T$,由上面推导知 $s'\in T_{k_i}$,故 $T(s,i)\subseteq T_{k_i}$.

必要性:因为 $T(s,i)\subseteq T_{k_i}$,所以$(M,s)|=K_i\varphi$,因而对任意 $t\in T(s,i)$,有$(M,t)|=\varphi$,即 $t\in T$,故 $T(s,i)\subseteq T$. □

因此,我们要模型检测 $K_i\varphi$,即要求 T_{k_i},只需遍历集合 T 的所有状态,每一步对于状态 s,只要 $T(s,i)\subseteq T$,就将 $T(s,i)$放到结果集 T_{k_i} 中,并且从 T 中削除 $T(s,i)$,从而削减了集合 T,否则不将 $T(s,i)$放入结果集 T_{k_i} 中,只从 T 中削除 $T(s,i)\cap T$,从而削减集合 T;然后从 T 中取出另一状态,重复以上过程,直到遍历完 T 中所有状态,T 变成空集为止,最后得到的结果集 T_{k_i} 就是我们所要求的满足规范 $K_i\varphi$的状态集合.算法描述如下所示:

规范 K_{ij} 的模型检测算法.

```
function modelcheckingKi(T)
    Q:=∅;   //∅为空集;
    for all s∈T do
        P:=evaluateT(s,i);
        if  P⊆T then
            Q:=Q∪P;
            T:=T\P;
        else
            T:=T\P;
        end if;
    end for all;
```

return(Q);
end function
function *evaluateT*(s,i)
Q:=∅; //∅为空集;
for all s' such that $(s,s')\in K_i$ do
Q:=$Q\cup\{s'\}$;
end for all;
return(Q);
end function

上述算法中 T 为利用 SMV 检测到的满足规范 φ的状态集合.

2.2 $K_jK_i\varphi$的模型检测

设 $T_{k_{ji}}=\{s|s\in S,(M,s)|=K_jK_i\varphi\}$,那么模型检测规范 $K_jK_i\varphi$的问题就是要求 $T_{k_{ji}}$ 的问题.

命题 4. 对任意 CTL 公式 φ和任意智体 agent i,agent j,如果 $K_i(i=1,\ldots,n)$是等价关系,那么 $T_{k_{ji}}\subseteq T_{k_i}$.

证明:设 $s\in T_{k_{ji}}$,那么$(M,s)|=K_jK_i\varphi$,由定义知,对所有满足$(s,s')\in K_j$ 的 s',有$(M,s')|=K_i\varphi$,因为 K_j 是等价关系,所以$(s,s)\in K_j$,由上面的证明知$(M,s)|=K_i\varphi$,所以 $s\in T_{k_i}$,故 $T_{k_{ji}}\subseteq T_{k_i}$. □

下面讨论如何求 $T_{k_{ji}}$ 的问题:

命题 5. 对任意 $s\in T_{k_i}$,且 $K_i(i=1,\ldots,n)$是等价关系,那么 $T(s,j)\subseteq T_{k_{ji}}$ 的充要条件是 $T(s,j)\subseteq T_{k_i}$.

命题 5 的证明与命题 3 的证明类似.

因此,我们要模型检测 $K_jK_i\varphi$,即求 $T_{k_{ji}}$,先求 T_{k_i},然后遍历集合 T_{k_i} 的所有状态,每一步对于状态 s,只要 $T(s,j)\subseteq T_{k_i}$,就将 $T(s,j)$放到结果集 $T_{k_{ji}}$ 中,并从 T_{k_i} 中削除 $T(s,j)$,从而削减集合 T_{k_i},否则不将 $T(s,j)$放入结果集 $T_{k_{ji}}$ 中,只从 T_{k_i} 中削除 $T(s,j)$,从而削减集合 T_{k_i},然后从 T_{k_i} 中取出另一状态,重复以上过程,直到遍历完 T_{k_i} 中所有状态,T_{k_i} 变为空集为止,此时得到的结果集 $T_{k_{ji}}$ 就是我们所要求的满足规范 $K_jK_i\varphi$的状态集合.算法描述如下:

规范 $K_jK_i\varphi$的模型检测算法.
function *modelcheckingK*$_j$*K*$_i$(T)
Q:=∅; //∅为空集;
P:=*modelcheckingK*$_i$(T);
for all $s\in P$ do
L:=*evaluateT*(s,j);
if $L\subseteq P$ then
Q:=$Q\cup L$;
P:=$P\backslash L$;
else
P:=$P\backslash L$;
end if;
end for all;
return(Q);
end function

2.3 公共知识$C_G\varphi$的模型检测

给定集合 $G\subseteq${agent 1,…,agent n},为了讨论方便,不妨设 G={agent 1,…,agent m},$C_G\varphi$表示φ是 G 中各 agent 的公共知识,定义 $T_c=\{s|s\in S,(M,s)|=C_G\varphi\}$.

命题 6. 对任意的 CTL 公式 φ和{agent 1,…,agent n}的子集 G,如果 $K_i(i=1,\ldots,n)$是等价关系,那么 $T_c\subseteq T$.

证明:假定 $s\in T_c$,那么$(M,s)|=C_G\varphi$,从而对每一自然数 k,有$(M,s)|=E_G^k\varphi$,(k=1,2,…),特别地,当 k=1 时,$(M,s)|=K_i\varphi$ (i=1,2,…,m),所以 $s\in T_{k_i}$,由命题 2 知,$T_{k_i}\subseteq T$,故 $T_c\subseteq T$. □

设 $T_k=\{s|s\in S,(M,s)|=E_G^k\varphi\}$($k$=1,2,…).

命题 7. 对任意 agent 子集 G={agent 1,…,agent m}和 CTL 公式φ,如果 K_i(i=1,…,m)是等价关系,那么 $T_{k+1}\subseteq T_k$ (k=1,2,…).

证明:设 $s\in T_{k+1}$,那么$(M,s)|=E_G^{k+1}\varphi$,即对任意 i (i=1,2,…,m),$(M,s)|=K_iE_G^k\varphi$,所以对任意 $t\in T(s,i)$(即$(s,t)\in K_i$),有$(M,t)|=E_G^k\varphi$,因为 K_i 是等价关系,所以 $s\in T(s,i)$(即$(s,s)\in K_i$).所以$(M,s)|=E_G^k\varphi$,即 $s\in T_k$,故 $T_{k+1}\subseteq T_k$(k=1,2,…). □

下面的命题证明了在已知 T_k 的情况下如何求 T_{k+1}.

命题 8. 设 K_i 是等价关系,$s\in T_k$,那么 $s\in T_{k+1}$ 的充要条件是 $T(s,i)\subseteq T_k$(对所有 i=1,2,…,m).

证明:充分性:因为 $T(s,i)\subseteq T_k$ (i=1,2,…,m),所以对任意 $t\in T(s,i)$(即$(s,t)\in K_i$),有$(M,t)|=E_G^k\varphi$,由知识算子的语义知$(M,s)|=K_iE_G^k\varphi$ (i=1,2,…,m),所以$(M,s)|=E_G^{k+1}\varphi$,即 $s\in T_{k+1}$.

必要性:假设存在某个 $i\in\{1,2,\ldots,m\}$,使得 $T(s,i)\subseteq T_k$ 不成立,那么存在 $t_0\in T(s,i)$(即$(s,t_0)\in K_i$),但 $t_0\notin T_k$,即$(M,t_0)|=E_G^k\varphi$不成立,由知识算子的语义知$(M,s)|=K_iE_G^k\varphi$不成立,从而$(M,s)|=E_G^{k+1}\varphi$不成立,即 $s\notin T_{k+1}$. □

命题 9. 存在 m_0($1\le m_0\le|S|$)(这里$|S|$表示状态集合 S 中的状态个数),使得对任意 $j\ge m_0$,有 $T_j=T_{m_0}$.

证明:设$|S|=M$,因为$\{T_K\}$是递减的状态集合序列,所以 $T_1,\ldots,T_M,T_{M+1}$ 中必存在某个 m_0 使得 $T_{m_0+1}=T_{m_0}$,否则,假设 $T_1,\ldots,T_M,T_{M+1}$ 中任意两个集合都不等,因为 T_1 元素最多为 M,故 T_M 的元素个数最多为 0,T_{M+1} 的元素个数就也为 0 了,即 $T_M=T_{M+1}$,故存在某个 m_0,使得

$$T_{m_0+1}=T_{m_0} \tag{1}$$

现在证明对任意的整数 k,如果 $T_{k+1}=T_k$,那么 $T_{k+2}=T_{k+1}$.证明如下:设 $s\in T_{k+1}$,那么对任意 $t\in T(s,i)$(i=1,2,…,m),有$(M,t)|=E_G^k\varphi$,即 $t\in T_k$,又因为 $T_{k+1}=T_k$,所以 $t\in T_{k+1}$,从而$(M,t)|=E_G^{k+1}\varphi$,所以$(M,s)|=K_iE_G^{k+1}\varphi$ (i=1,2,…,m),从而$(M,s)|=E_G^{k+2}\varphi$,所以 $s\in T_{k+2}$,故 $T_{k+1}\subseteq T_{k+2}$,由命题 7 知,$T_{k+1}\supseteq T_{k+2}$,故 $T_{k+1}=T_{k+2}$.由 k 的任意性及式(1)易得:对任意 $j\ge m_0$,有 $T_j=T_{m_0}$. □

命题 10. 存在正整数 $m_0\le|S|$使得 $T_c=T_{m_0}$.

证明:显然 $T_c=\cap^{\infty}_{k=1}T_k$,由命题 9 知,$T_c=\cap^{m_0}_{k=1}T_k$,由命题 7 知,$\{T_k\}$是单调递减状态集合序列,所以 $T_c=T_{m_0}$.□

由以上命题,我们容易得出求 T_c 的算法:我们要求满足规范 $C_G\varphi$的状态集合,即要求 T_c,先求 T_1,T_2,如果 $T_1=T_2$,那么由命题 9 和命题 10 知:T_1 就是满足规范 $C_G\varphi$的状态集合 T_c,否则再求 T_3,如果 $T_3=T_2$,那么 T_2 就是满足规范 $C_G\varphi$的状态集合 T_c,否则再求 T_4,如此重复上述过程,由命题 10 知:最多$|S|$步即可求出 T_c.

上面每一步中求 T_{k+1} 的算法如下:

由命题 8 知:只需遍历集合 T_k,对每一状态 $s\in T_k$,如果 $T(s,1)\subseteq T_k,T(s,2)\subseteq T_k,\ldots,T(s,m)\subseteq T_k$,那么就将 s 放入结果集 T_{k+1} 中,并且从 T_k 中削除 s,否则不将 s 放入结果集 T_{k+1} 中,只从 T_k 中削除 s,然后从 T_k 中取出另一状态,重复上述过程,直到遍历完 T_k 中所有状态,T_k 变成空集为止,最后我们得到结果集 T_{k+1}.算法描述如下:

规范 $C_G\varphi$的模型检测算法.

```
function modelcheckingC_G(T)
    Q:=T;
    Q′:=evaluateK(Q);
    while (Q≠Q′) do
        Q:=Q′;
        Q′:=evaluateK(Q′);
    end while;
    return(Q);
end function
function evaluateK(Q)     //由 T_k 计算 T_{k+1},Q 为 T_k(k=1,2,…);
    Q′:=∅;       //∅为空集;
```

```
        for all s∈Q do
            L:=∪^m_{i=1}T(s,i);
            if L⊆Q then
                Q′:=Q′∪{s};
                Q:=Q\{s};
            else
                Q:=Q\{s};
            end if;
        end for all;
        return(Q′);
    end function
```

3 算法的复杂性分析

这里只考虑我们的算法的复杂性.设 Kripke 结构 $M=(S,R,L)$,相应的等价关系为 $K_1,K_2,\ldots,K_n$.

命题 11. 在规范 $K_i\varphi$的模型检测中,算法的时间复杂性不超过 $2|S|^2$,因而是多项式时间的(对于 S 中状态个数$|S|$).

证明:对每一个 s,求 $T(s,i)$最多需$|S|$步,在对 T 的遍历过程中,每一步判定 $T(s,i)\subseteq T$,需$|T(s,i)|\times|S|$步,设第 j 步的 $T(s,i)=S(j)$,且假设经过 k 步后 T 被削减为空集,算法结束,显然 $T=\cup_{j=1}^{k}S(j)$且对任意 $i,j(1\le i,j\le k)$有 $S(i)\cap S(j)=\varnothing$(这里∅表示空集).因此遍历完 T 总共最多需要$\sum^{k}_{j=1}(|S|+|S(j)|\times|S|)\le k|S|+|S|^2\le 2|S|^2$ 步,因此算法的复杂性是多项式时间的(相对于 S 中状态个数$|S|$). □

命题 12. 在规范 $K_jK_i\varphi$的模型检测中,算法的时间复杂性不超过 $4|S|^2$,因而是多项式时间的(对于 S 中状态个数$|S|$).

证明过程类似于命题 11 的证明过程.

命题 13. 在公共知识 $C_G\varphi$的模型检测中,算法的时间复杂性不超过 $2m\times|S|^3$,因而是多项式时间的(对于 S 中状态个数$|S|$).

证明:假定 G 中智体个数为 m,由于求 T_c 的算法如下:先计算 T_1 和 T_2,如果 $T_2=T_1$,那么 T_1 就是要求的 T_c,否则计算 T_3,如果 $T_3=T_2$,那么 T_2 就是要求的 T_c,如此下去,由命题 9 和命题 10 可知,最多 m_0 步就有 $T_{m_0+1}=T_{m_0}$,因而 T_{m_0} 就是要求的 T_c.下面计算已知 T_k 求 T_{k+1} 的时间,计算过程与命题 11 类似,不同的是每步要判断 m 个式子 $T(s,i)\subseteq T_k$ $(i=1,\ldots,m)$,因此已知 T_k 求 T_{k+1} 的最多步数为 $2m|S|^2$,从而求 T_c 的总步数最多为 $m_0\times 2m|S|^2\le 2m|S|^3$ 步(这里 m_0 与命题 9 和命题 10 中的 m_0 相同),因此算法的时间复杂性不超过 $2m|S|^3$,是多项式时间的(对于 S 中状态个数$|S|$). □

由于模型检测 CTL 规范的算法复杂性是线性时间的,我们的算法复杂性是多项式时间的,所以整个规范 $K_i\varphi$, $K_jK_i\varphi$及 $C_G\varphi$的模型检测算法复杂性是多项式时间的(对于 S 中状态个数$|S|$).

4 实 例

我们以 TMN 密码协议作为实例来进行分析[9].

4.1 TMN密码协议

TMN 密码协议是应用于移动通信系统的密码分配协议,原始协议如下:

$A\rightarrow S:B,\{N_a\}_{K_s}$

$S\rightarrow B:A$

$B\rightarrow S:A,\{N_b\}_{K_s}$

$S\rightarrow A:B,\{N_a\}_{N_a}$

其中 A 为初始者,B 为响应者,S 为可信第三方,K_s 为 S 的公钥,N_a,N_b 是 A 和 B 发布的具有新鲜性的随机数,N_a 也

是 A 的公钥,N_b 是 B 的公钥也作为 A,B 间秘密通信的会话密钥,协议运行的目的是在 A 和 B 之间建立一个会话密钥 N_b(即 K_{ab}),这个密钥在他们今后秘密通信时使用.

4.2 系统模型

我们假设有一个攻击者 I.在整个协议的运行过程中,A 具有的信息有 A_0(初始信息),$\{B,\{N_a\}_{K_s}\}$,$\{B,\{N\}_{N_a}\}$,其中 N 可以是 N_b 也可以是 $N_{I(B)}$(当 I 冒充 B 的身份时).因此,根据 A 能观察到的信息的不同,A 的局部状态集 S_A 有 3 种状态:

S_{A1} 对应初始信息$\{A_0\}$ (协议运行之前).

S_{A2} 对应观察到信息$\{\{A_0\},\{B,\{N_a\}_{K_s}\}\}$ (协议运行第 1 步之后).

S_{A3} 对应观察到信息$\{\{A_0\},\{B,N_a,\{N_a\}_{K_s}\},\{B,\{N\}_{N_a}\}\}$ (协议运行第 4 步之后).

同理,B 的局部状态集 S_B 有 3 种状态:

S_{B1} 对应初始信息$\{B_0\}$.

S_{B2} 对应观察到信息$\{\{B_0\},\{A\}\}$.

S_{B3} 对应观察到信息$\{\{B_0\},\{A\},\{A,N_b,\{N_b\}_{K_s}\}\}$.

由于 I 可以知道正在通信的任何信息,所以 I 的局部状态集 S_I 有 5 种状态:

S_{I1} 对应初始信息$\{I_0\}$.

S_{I2} 对应观察到信息$\{\{I_0\},\{B,\{N_a\}_{K_s}\}\}$.

S_{I3} 对应观察到信息$\{\{A\},\{I_0\},\{B,\{N_a\}_{K_s}\}\}$.

S_{I4} 对应观察到信息$\{\{A\},\{I_0\},\{B,\{N_a\}_{K_s}\},\{B,\{N_b\}_{K_s}\}\}$.

S_{I5} 对应观察到信息$\{\{B,\{N\}_{N_a}\},\{B,\{N_b\}_{K_s}\},\{A\},\{I_0\},\{B,\{N_a\}_{K_s}\}\}$.

因为 S 是可信服务器,所以在不考虑服务器 S 的前提下,全局状态集 $S=S_A\times S_B\times S_I$ 最多有 3×3×5=45 个状态,每个全局状态可表示为 $s=(s_a,s_b,s_I)$.有些不可能状态可以删除.

对于状态转移关系 R,部分描述如下:在 A 发送信息$\{B,\{N_a\}_{K_s}\}$前,智体 A 的局部状态为 s_{A1},智体 B 的局部状态为 s_{B1},I 的局部状态为 s_{I1},故系统全局状态为(s_{A1},s_{B1},s_{I1});当 A 发送信息$\{B,\{N_a\}_{K_s}\}$后,智体 A 的局部状态变为 s_{A2},智体 B 的局部状态仍为 s_{B1},I 的局部状态变为 s_{I2},故系统全局状态由(s_{A1},s_{B1},s_{I1})转移到(s_{A2},s_{B1},s_{I2});同样,当 B 接收信息$\{A\}$后,系统全局状态由(s_{A2},s_{B1},s_{I2})转移到(s_{A2},s_{B2},s_{I3}).

4.3 系统规范

通过 TMN 密码协议的运行,智体 A 与 B 要达到建立安全会话密钥的目的,以便进行秘密通信,要达到该目的,协议必须满足两个安全性质(即条件):(1) 智体 A 与 B 在协议运行后必须相互知道他们之间的会话密钥是 K_{ab},即"A 和 B 之间的会话密钥是 K_{ab}"是 A 和 B 之间的公共知识;(2) 攻击者 I 不知道 A 和 B 之间的会话密钥是 K_{ab},用 φ 表示"A 和 B 之间的会话密钥是 K_{ab}",$K_A\varphi$ 表示"A 知道 φ",$K_B\varphi$ 表示"B 知道 φ",$K_I\varphi$ 表示"I 知道 φ",$C_G\varphi$ 表示 φ 是 A 和 B 之间的公共知识,其中 G={agent A, agent B},则规范的形式化表示为 $C_G\varphi\wedge\neg K_I\varphi$.

4.4 系统规范的检验

要检验规范 $C_G\varphi\wedge\neg K_I\varphi$,可以分别检验 $C_G\varphi$和$\neg K_I\varphi$.在检验 $C_G\varphi$时,先用 SMV 检验 φ,然后用我们的算法进一步检验公共知识 $C_G\varphi$;在检验$\neg K_I\varphi$时,先用 SMV 检验 φ,然后用我们的算法检验知识$\neg K_I\varphi$.根据上述思路,用 SMV 语言编写 TMN 密码协议的描述作为 SMV 的输入,然后运行 SMV 和我们的程序,检测到系统(即 TMN 密码协议)是不满足规范 $C_G\varphi\wedge\neg K_I\varphi$的.

5 结束语

本文在 SMV 模型检测器的基础上,根据知识的语义和集合理论,提出了多种多项式时间算法,从而使 SMV 的检测功能从分支时态逻辑扩充到时态认知逻辑,与 SMV 比较,其主要的功能改进和优点是:

(1) 与 SMV 结合能检测知识算子表示的规范;

(2) 与 SMV 结合能检测多层知识表示的规范;

(3) 与 SMV 结合能检测公共知识表示的规范;

(4) 算法的时间复杂性是多项式时间的.

目前,模型检测器比较多,例如 FDR,SMV,MUR 和 NRL 等,它们主要用来检测系统是否满足用 LTL 或 CTL 时态公式表示的规范,据我们所知,还很少有工具能检测用时态认知逻辑表示的规范.

在这些检测工具中,许多工具的输出是满足时态逻辑规范的状态集合.我们研究的方法可以与这些检测工具结合,进一步检测时态认知逻辑表述的规范,从而使它们的功能扩充到时态认知逻辑的检测,更好地用于协议验证.

我们准备今后将我们的代码加入到 SMV 中,形成一种新的工具和描述语言.这种工具不但能够检测时态逻辑表述的规范,而且能够检测时态认知逻辑表述的规范,从而更好地用于协议验证.我们也将用这种新工具验证更多的协议.

致谢 审稿专家和编辑部对本文的工作给予了很大的支持和很多建议,在此深表谢意.

References:

[1] Mcmillan KL. Symbolic model checking: 10^{20} states and beyond. Information and Computation, 1992,98(2):142~170.

[2] Holzmann G. Design and Validation of Computer Protocols. Prentice Hall, 1990.

[3] Holzmann G. The model checker spin. IEEE Trans. on Software Engineering, 1997,23(5):279~295.

[4] van der Meyden R, Shilov NV. Model checking knowledge and time in systems with perfect recall (extended abstract). In: Goos CC, ed. Foundations of Software Technology and Theoretical Computer Science (LNCS 1738). Berlin: Springer-Verlag, 1999. 432~445.

[5] van der Hoek W, Wooldridge M. Model checking knowledge and time. In: Stefan Leue CC, ed. Proc. of the 9th Int'l Spin Workshop on Model Checking of Software. Berlin: Springer-Verlag, 2002. 1~16.

[6] Clarke EM, Grumberg O, Peled DA. Model Checking. Cambridge: MIT Press, 1999. 1~33.

[7] Halpern JY, Vardi MY. The complexity of reasoning about knowledge and time: Extended abstract. Journal of Computer and System Sciences, 1989,38(1):195~237.

[8] Fagin R, Halpern JY, Moses Y, Vardi MY. Reasoning about Knowledge. Cambridge: MIT Press, 1995. 111~120.

[9] Zhang YQ, Wang CL, Feng DG. SMV analysis of TMN cryptographic protocol. Journal of Computer Reseach and Development, 2003,40(2):258~262 (in Chinese with English abstract).

附中文参考文献:

[9] 张玉清,王春玲,冯登国.TMN 密码协议的 SMV 分析.计算机研究与发展,2003,40(2):258~262.