时态认知逻辑 CTL * K 的符号化模型检查算法

陈 彬 王智学
(解放军理工大学指挥自动化学院 南京 210007)

摘 要 时序认知逻辑是由时序逻辑和认知逻辑组合而成的逻辑,主要应用于多主体系统的规范定义。大多数时序认知逻辑是基于CTL的,表达能力有限。并且已知的一些模型检查算法存在内存不足和状态爆炸等问题。讨论了基于CTL*的时态认知逻辑CTL*K的语法、语义和模型,它能够在表达力很强的时态逻辑CTL*基础上描述智能体的知识、目标等意向特征。并给出了CTL*K的模型检查算法,其核心思想就是将CTL*K公式的检查问题转化为CTL*公式的模型检查问题,可以使检查的系统规模得以大幅度提高。并且将算法编码后容易集成到NuSMV模型检查器。

关键词 符号模型检测,多主体系统,时态认知逻辑
中图法分类号 TP311 **文献标识码** A

Symbolic Model Checking Algorithm for Temporal Epistemic Logic CTL * K

CHEN Bin WANG Zhi-xue
(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

Abstract Temporal epistemic logics have been gradually used in specification of multiple agents system, which are composed by temporal logics and epistemic logics. Most of temporal epistemic logics are based on CTL, which have a limited expressivity. And some model checking techniques existing for them have problems such as memory-shortage and state-explosion. A temporal epistemic logic CTL * K based on CTL * was proposed. Through the definition of syntax and semantics, CTL * K had a strong expressivity and could describe agents' epistemic properties such as belief and goal. To check CTL * K, a symbolic model checking algorithm for CTL * K was offered, which translated a CTL * K formula into a common CTL * formula and could be easily encoded into NuSMV model checker. The experiment showed that the algorithm could obviously enlarge the size of system to be checked.

Keywords Symbolic model checking, Multiple agent system, Temporal epistemic logic

模型检查(model checking)技术主要检查用时态逻辑描述的规范,如SMV检查分支时态逻辑[1]、SPIN检查线性时态逻辑[2,3],人们很少注意时态认知逻辑的模型检查问题。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多智能体系统(Multiple Agents System, MAS)性质的规范描述,因此模型检查时态认知逻辑是一个比较新的重要课题。在基于LTL(linear temporal logic)时态认知逻辑方面,Wiebe van der Hoek等人通过引入局部命题将CKLn模型检查问题转化为LTL模型检查问题,然后利用现有的LTL模型检查方法和工具来完成时态认知逻辑CKLn的模型检查[4]。而在基于CTL(Computation Tree Logic)时态认知逻辑方面,吴立军等人根据知识语义和集合理论,利用SMV模型检查器检查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CKCn的模型检查[5]。

这些方法解决了两类认知逻辑描述的规范的模型检查问题,但还存在问题。首先这些方法一般都需要构建整个状态空间,这样对大状态空间系统来说就可能存在内存不足等问题。吴立军等人在原有方法上做了改进,提出了"On the Fly"模型检查时态认知规范的算法,该算法只需产生系统的部分甚至小部分状态空间就能找到一个反例,无需生成整个搜索空间且算法的时间复杂性是多项式时间的。但这也只能在一定程度上减轻状态爆炸问题[6]。另外,Wiebe van der Hoek等人的工作的理论基础还需进一步加以研究[4];而吴立军等人提出的时态认知逻辑模型检查方法是基于状态空间的,不是基于OBDDs的;但SMV工具采用符号化模型检查算法检查系统是否满足用分支时态逻辑描述的规范,是基于OBDDs的,通过这两种截然不同的方法的集成来解决CKCn的模型检查问题既不自然,也是困难的。

本文主要讨论了基于CTL*(Computation Tree Logic)的时态认知逻辑。这里将其称为CTL*K。对于描述状态转换系统的性质来说,时态逻辑CTL*是一种比CTL和LTL表达力更强的规范描述逻辑[7]。它用时态算子描述沿计算路径的状态变化,用路径算子解释时间分支性质。很多系统性质可以用CTL或LTL表达,但有些性质只能用CTL*表达。如$E(FXp) \vee EG(EXp)$用CTL和LTL均不能表达,因此一个功能全面的模型检查算法应能检查CTL*公式。另外,本

到稿日期:2008-06-16 本文受863高技术计划基金项目(2007AA01Z126),国防预研基金项目(9140A06020206JB8101)资助。
陈 彬(1979-),男,讲师,主要研究方向为需求工程等,E-mail:chenbinmsn@msn.com;王智学(1962-),男,教授,主要研究方向为需求工程等。

文在 CTL＊的基础上，主要讨论了知识算子和目标算子的语法和语义，这两种算子对于描述智能体的意向特征是非常重要的。

对于 CTL＊K 的模型检查，本文设计的算法的核心思想就是将 CTL＊K 公式的检查问题转化为 CTL＊公式的模型检查问题。另外，为了改善状态空间爆炸问题，本文采用了更好的符号化方法[1]，不需要显式地构建状态空间，可以使检查的系统规模得以大幅度提高。并且将算法编码后，容易集成到 NuSMV 模型检查器，从而使 NuSMV 的检查功能从分支时态逻辑扩充到时态认知逻辑 CTL＊K。目前我们正在着手进行工具的开发，以便于更好地用于 MAS 性质的验证。

1 CTL＊K 的语法和语义

1.1 CTL＊K 语法

CTL＊[8]可以表示分支和线性性质，CTL＊公式由路径公式和时态公式组成。路径公式包含路径算子，分别是 A(对于所有的路径)和 E(对于所有的路径)。时态公式包括时态算子，用来描述沿着路径所具有的性质：X(下一状态)和 F(将来某些状态)、G(总是)和 U(直到…为止)。我们要讨论的是基于 CTL＊(Computation Tree Logic)的时态认知逻辑，这里记为 CTL＊K。CTL＊K 除了包括 CTL＊的基本语法成分外，还包括由知识算子 K_i、实现型目标算子 $A-G_i$ 和维护型目标算子 $M-G_i$ 构成的意向公式，下标 i 表示智能体 i，如 $K_i\varphi$ 表示“智能体 i 知道 φ”。CTL＊K 公式包括：

(1)状态公式的语法为：

a)如果 $p\in AP$，则 p 是一个状态公式；

b)如果 φ 和 ψ 是状态公式，则 $\neg\varphi$ 和 $\varphi\wedge\psi$ 是状态公式；

c)如果 φ 是路径公式，则 $E\varphi$ 和 $A(\varphi)$ 是状态公式；

(2)路径公式的语法为：

a)如果 φ 是一个状态公式，则 φ 也是一个路径公式；

b)如果 φ 和 ψ 是路径公式，则 $\neg\varphi,\varphi\wedge\psi,X\varphi,F\varphi,G\varphi$ 和 $\varphi U\psi$ 是路径公式；

(3)意向公式的语法为：

a)如果 φ 是一个状态公式，则 $K_i\varphi$ 是一个意向公式；

b)如果 φ 是一个状态公式，则 $A-G_i\varphi$ 是一个意向公式；

c)如果 φ 是一个状态公式，则 $M-G_i\varphi$ 是一个意向公式。

1.2 Kripke 结构扩展

CTL＊K 逻辑中公式的语义可用 Kripke 结构给出。Kripke 结构是一个元组 $M=(W,I,R,AP,\pi)$，其中：

(1)W 表示一状态(世界)集合；

(2)I 表示初始状态(世界)集合；

(3)$R\subseteq W\times W$ 表示状态间的变迁关系集合，把一个状态 $w\in W$ 和其后继状态 $w'\in W$ 联系起来；

(4)$AP=\{p,q,\cdots.\}$ 是原子命题集合；

(5)π 将每个状态 w 和对原子命题 p 的一个真值赋值联系起来，即对于每个 $w\in W$ 和原子命题 p，有 $\pi(w)(p)\in$ {true,false}；

为了表达意向算子的语义，需要对 M 进行扩展。扩展后的 Kripke 结构是一个元组：$(W,I,R,AP,\pi,K_1,\cdots,K_n)$，仍记为 M。其中：

(6)$K_1,\cdots,K_n$ 是 W 上的等价关系。K_i^M 和 π^M 用来表示 Kripke 结构 M 中的 K_i 和 π 函数。当不引起混淆时，忽略上缀 M。定义 $K_i(w)=\{w'\mid(w,w')\in K_i\}$，即 $K_i(w)$ 是智能体 i 在状态 w 中认为可能的状态集合。

1.3 CTL＊K 语义

为了表示 CTL＊K 公式的语义，首先定义情景(situation)和路径(path)的概念。

定义 1 情景是由 Kripke 结构 M 和状态 w 组成的(M,w)，或者是由 M 和路径 σ 组成的(M,σ)。通过情景，可以归纳地给出公式的语义，例如对于原子命题 $p\in AP$，$(M,w)\mid=piff\ \pi^M(w)(p)=$true。

定义 2 设 $M=(W,I,R,AP,\pi,K_1,\cdots,K_n)$ 是一个 Kripke 结构，并设 $w_0\in W$：

(1)M 的一条路径 σ 是从 w_0 出发的一条路径。如果 $\sigma=w_0$(从 w_0 出发的空路径)或者 σ 是一个(可能是无限的)状态序列，即 $\sigma=(w_0,w_1)(w_1,w_2)\cdots(w_n,w_{n+1})$，其中对任意 $i>0,(w_{i-1},w_i)\in R$。

(2)如果 σ 是 M 的一条路径，路径的长度表示为 $|\sigma|$。如果 σ 是一个无限路径，那么 $|\sigma|=\infty$。如果 $\sigma=w_0$，那么 $|\sigma|=0$。对某些 $n\geqslant0$，如果 $\sigma=(w_0,w_1)(w_1,w_2)\cdots(w_n,w_{n+1})$，$|\sigma|=n+1$。另外，用 $\sigma(i)$ 表示路径上的第 i 个状态 w_i，用 $\sigma[w]$ 表示从状态 w 出发的所有路径的集合，即 $\sigma[w]=\{\sigma'\mid\sigma'$ 是 M 的一条路径，且 $\exists j\geqslant0,\sigma'(j)=w\}$，$\sigma^i$ 表示该路径中状态 w_i 之后的路径，即 $(w_i,w_{i+1}),(w_{i+1},w_{i+2}),\cdots,(w_n,w_{n+1})$。

CTL＊K 公式的语义定义如下：

(1)$M,w\vdash$ true 总是为真；

(2)$M,w\vdash piff\pi(w)(p)=$true；

(3)$M,w\vdash\neg\varphi$ iff $M,w|\neq\varphi$；

(4)$M,w\vdash\varphi\wedge\psi$ iff $M,w\vdash\varphi$ 且 $M,w\vdash\psi$；

(5)$M,w\vdash\varphi\wedge\psi$ iff $M,w\vdash\varphi$ 且 $M,w\vdash\psi$；

(6)$M,w\vdash A\varphi$ iff $\forall\sigma\in\sigma(w)$，如果 $\sigma(j)=w$，则 $M,\sigma^j\vdash\varphi$；

(7)$M,w\vdash E\varphi$ iff $\exists\sigma\in\sigma(w)$，如果 $\sigma(j)=w$，则 $M,\sigma^j\vdash\varphi$；

(8)$M,\sigma\vdash p$ iff $M,\sigma(0)\vdash p$；

(9)$M,\sigma\vdash\neg\varphi$ iff $M,\sigma(0)|\neq\varphi$；

(10)$M,\sigma\vdash\varphi\wedge\psi$ iff $M,\sigma\vdash\varphi$ 且 $M,\sigma\vdash\varphi$；

(11)$M,\sigma\vdash X\varphi$ iff $M,\sigma(1)\vdash\varphi$；

(12)$M,\sigma\vdash F\varphi$ iff $\exists i\geqslant0:M,\sigma(i)\vdash\varphi$；

(13)$M,\sigma\vdash G\varphi$ iff $\forall i\geqslant0:M,\sigma(i)\vdash\varphi$；

(14)$M,\sigma\vdash\varphi U\psi$ iff $\exists i\geqslant0:M,\sigma(i)\vdash\varphi$ 且 $\forall j<i:M,\sigma(j)\vdash\varphi$；

扩展后的 Kripke 结构 M 可以表示意向算子。知识算子、实现型目标和维护型目标算子的语义分别为：

(15)$M,w\vdash K_i\varphi$ iff $\forall w'\in K_i(w),M,w'\vdash\varphi$；

(16)$M,w\vdash A-G_i\varphi$ iff $M,w\vdash K_i(\neg\varphi)\wedge AF(K_i\varphi)$；

(17)$M,w\vdash M-G_i\varphi$ iff $M,w\vdash K_i(\neg\varphi)\wedge AF(K_i(G\varphi))$。

给定一个 Kripke 结构 M 和一个用来表示性质的状态公式 φ，模型检查问题就是计算出 M 中那些满足公式 φ 的所有状态，即集合$\{\sigma(i)\mid\sigma$ 是 M 中的路径并且 $M,\sigma(i)\mid=\varphi\}$。

2 基于知识结构的 Kripke 结构

以上在基于状态空间的 Kripke 结构上给出了 CTL＊K 的语法和语义。为了能够对意向公式进行符号化模型检查，

下面首先给出基于知识结构的符号化 Kripke 结构概念。

2.1 知识结构

定义(带有 n 个智能体) 知识结构 $\Gamma=(V,\Theta,\tau(V,V'),O_1,\cdots,O_n)$。其中,$V=\{v_1,\cdots,v_n\}$ 为系统变量的集合,每个变量是布尔值的;Θ 是 V 上的布尔公式集合,每个布尔公式刻画了 Γ 的初始状态,因此 Θ 构成了 Γ 的初始条件集合。一个状态被定义为初始的,当且仅当该状态满足 Θ 中的某个公式;令 V' 为 V 的后继版本,即 $V'=\{v'_1,\cdots,v'_n\}$,τ 是 $V\cup V'$ 一个布尔公式,称为变迁关系;对每个 i,$O_i\subseteq V$,表示智能体 i 的可观察变量集合。

设 θ 是 Θ 中所有布尔公式的合取,即 $\theta=\varphi_1\wedge\cdots\wedge\varphi_n$,其中对 $\forall i,1\leqslant i\leqslant n,\varphi_i\in\Theta$,并且 $\{\varphi_1\}\cup\{\varphi_2\}\cdots\cup\{\varphi_n\}=\Theta$。满足 Θ 的 V(或 V 的一个子集)的任一真值赋值可看作是知识结构的一个状态。在这种解释下,Θ 中每个布尔公式都表示一个状态集合,即满足该公式的真值赋值集合。如有两个系统变量 a 和 b,则公式 $a\vee b$ 表示所有使 a 或 b 为真的状态,即相当于 $\{(1,0),(1,1),(0,1)\}$ 3 个状态的集合。给定一个状态 w,定义智能体 i 在状态 w 下的局部状态为 $w\cap O_i$。为方便起见,这里有时对集合和其特征函数不加区分。

2.2 基于知识结构 Γ 的符号化 Kripke 结构

把知识结构 Γ 和 Kripke 结构联系起来,构成基于知识结构 Γ 的 Kripke 结构 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$,$M(\Gamma)$是一个符号化的 Kripke 结构。其中:

(1)W 是知识结构的状态集合,即 W 是 $V\to B$ 的真值赋值集,其中 B 是布尔值向量;

(2)I 是初始状态集合。$w\in I$ 当且仅当 $w(\varphi)=true$,其中 $\varphi\in\Theta$;

(3)状态间的变迁关系 R 可以用变量集 $V\cup V'$ 上的公式 $\tau(V,V')$表示。状态 w'是状态 w 的后继,即 wRw',当且仅当 $\langle w,w'\rangle|=\tau(V,V')$。在这里,$\langle w,w'\rangle$是一个联合真值赋值,其中把 $v\in V$ 解释为 $w[v]$,把 v 的后继 v'解释为 $w'[v]$。假设 τ 是完全的,即对于每个 $w\in W$,有一个 $w'\in W$ 使得$\langle w,w'\rangle|=\tau(V,V')$。$\langle w,w'\rangle|=\tau(V,V')$表示为$\langle w,w'\rangle\in\tau$;

(4)AP 和 V 相同,即对 $\forall v\in V\Leftrightarrow v\in AP$;

(5)π 将每个状态 w 和对原子命题 p 的一个真值赋值联系起来,所以 $\pi(w)$与 w 相同,即对所有 $p\in AP,\pi(w)(p)=w(p)$;

(6)对于每个智能体 i 及两个状态 w 和 w',可得到 wK_iw' iff $w\cap O_i=w'\cap O_i$。

3 模型检查

CTL 和 LTL 都是 CTL* 的子逻辑。CTL* 公式中那些时态算子前都紧跟一个路径算子的公式为 CTL 公式,如 $EX\varphi,AF\varphi$等。那些不含路径算子 A 和 E 的 CTL* 公式则为 LTL 公式。显然,通过等价替换,X,U 和 E 足以表示 CTL* 中所有的路径和时态算子。

下面首先定义布尔公式的量化概念。

定义 3 给定一个命题变量集合 $V=\{v_1,\cdots,v_n\}$,设 φ 为 V 上的一个公式,则 V 上的量化公式可归纳地定义如下:

(1)$\exists\phi\varphi=\varphi$;

(2)$\exists\{v_i\}\varphi=\varphi(v_i\leftarrow\text{true})\vee\varphi(v_i\leftarrow\text{false})$;其中 $\varphi(v_i\leftarrow\text{true})$和 $\varphi(v_i\leftarrow\text{false})$分别表示 φ 中的 v_i 被 true 或 false 替换;

(3)$\exists V\varphi=\exists v_1.(\cdots(\exists v_n.\varphi)\cdots)$。

对于 $\forall$ 量化,和上述 $\exists$ 量化的定义方式相同,不过第(2)条要改为:$\forall\{v_i\}\varphi=\varphi(v_i\leftarrow\text{true})\wedge\varphi(v_i\leftarrow\text{false})$。

对于 CTL 公式的符号模型检查,有以下定理:

定理 1[9] 给定符号化 Kripke 结构 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$,CTL 公式 φ 和 ψ 为 V 上的公式,V'为 V 的后继版本,则 $EX\psi(V)=\exists V(\psi(V)\wedge R)(V\leftarrow V')$,简记为 $EX\psi(V)=\exists V'(\psi(V')\wedge R)$。而 EU 算子定义为某个单调算子的最小不动点,即 $EU(\varphi,\psi)=\nu Z.(\psi\vee(\varphi\wedge EXZ))$。

而对于 LTL 公式 φ 的检查,有以下定理:

定理 2[10] 任何 LTL 公式同时也是 CTL* 公式。它们的语义定义是一致的,即对于任何 $M(\Gamma)$,有 $M(\Gamma)|=_{LTL}\varphi$ iff $M(\Gamma)\models_{CTL^*}A\varphi$ iff $M(\Gamma)\models_{CTL^*}\varphi$。

另外,对于意向公式的符号化检查,有以下定理:

定理 3[11] 令 φ 为不包含任何意向模态词的公式,那么对于 $M(\Gamma)$中的状态 w,有

$(M(\Gamma),w)|=K_i\varphi\Leftrightarrow\forall(V-O_i)(\theta\to\varphi)$

由定理 3 可知,可以把意向公式的模型检查问题转化为 CTL* 公式的检查问题。当要检查意向公式 φ,或者 φ 的子公式中包含有意向公式时,首先根据定理 3 对意向公式进行处理,将其转化为不含意向算子的布尔公式,然后做进一步处理。

显然,一些 CTL* 公式的符号化检查不能通过定理 1 进行,如 $E\varphi,A\varphi$(φ 中没有时态算子)。为了能够对一般化的 $E\varphi$(φ 可能不包含时态算子,也可能包含时态算子)进行检查,我们采用文献[12]中的符号化检查 CTL* 公式的方法做如下处理:

设 φ 是一可能包含时态算子的 CTL* 公式,对于一个公式 ψ,用 $\psi\in\varphi$ 表示 ψ 是 φ 的一个子(可能等于)公式。公式 ψ 被称为主时态的,如果它的主要算子是时态算子,即 ψ 是如下形式:$X\alpha$ 或 $\alpha U\beta$。

给定一个知识结构 $\Gamma=(V,\Theta,\tau(V,V'),O_1,\cdots,O_n)$和一个公式 φ,我们定义另一个知识结构 $\Gamma_\varphi=(V_\varphi,\Theta_\varphi,\tau_\varphi(V_\varphi,V'_\varphi),O_{1\varphi},\cdots,O_{n\varphi})$如下:

(1)系统变量:Γ_φ 的系统变量 V_φ 包括 V 及附加的布尔变量集:$\vec{v_\varphi}=\{v_\psi\mid\psi$ 是 φ 的一个主时态子公式$\}$,即 $V_\varphi=V\cup\vec{v_\varphi}$。附加变量 v_ψ 在某个状态为真当且仅当公式 ψ 在该状态成立。

(2)另外,定义一个把 φ 的每一个子公式映射到 V_φ 上的布尔函数 χ:

$$\chi(\psi)=\begin{cases}\psi,\psi\text{ 是 }V\text{ 中的一个变量};\\ \neg\chi(\alpha),\psi=\neg\alpha;\\ \chi(\alpha)\wedge\chi(\beta),\psi=\alpha\wedge\beta;\\ v_\psi,\psi\text{ 是主时态公式}\end{cases}$$

用 χ 对 Θ 中的布尔公式进行映射,构成 Θ_φ,因此 Θ_φ 是 V_φ 上的布尔公式集合,即 $\Theta_\varphi=\{\chi(\varphi(V))\}$,则 θ_φ 是 Θ_φ 中所有布尔公式的合取。

(3)令 v_ψ'是 v_ψ 的后继版本(primed version)。对于 V_φ 上的公式 ψ,我们用 $\chi'(\psi)$或 $\psi(V_\varphi')$来表示公式 $\psi(V_\varphi\leftarrow V_\varphi')$,即 ψ 的后继版本,其中 $\psi(V_\varphi\leftarrow V'_\varphi)$表示把 ψ 中的 V_φ 用 V_φ'替换,则变迁关系 $\tau_\varphi(V_\varphi,V_\varphi')=\tau\wedge\bigwedge_{X\psi\in\varphi}(v_{X\psi}\leftrightarrow\chi'(\psi))\wedge\bigwedge_{\alpha U\beta\in\varphi}(v_{\alpha U\beta}\leftrightarrow(\chi(\beta)\vee(\chi(\alpha)\wedge v'_{\alpha U\beta})))$。

(4)此时认为每个智能体 i 的可观察变量集合 O_i 不发生

改变，即 $O_{i\varphi}=O_i$。给定一个状态 w，定义智能体 i 在状态 w 下的局部认知状态为 $w\cap O_{i\varphi}$ 仍旧等于 $w\cap O_i$。

设基于知识结构 Γ 的 Kripke 结构为 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$，我们定义一个基于知识结构 Γ_φ 的 Kripke 结构是 $(\Gamma_\varphi)=(W_\varphi,I_\varphi,R_\varphi,AP_\varphi,\pi_\varphi,K_{1\varphi},\cdots,K_{n\varphi})$ 如下：

(1)W_φ 是 $V_\varphi\to B$ 的真值赋值集；

注意，这时 W_φ 中的每个状态(也即真值赋值)包括对 V_φ 中 $\vec{v}_\varphi$ 部分的赋值。

(2)I 映射为 M_φ 的初始条件 I_φ。$w\in I_\varphi$ 当且仅当 $w(\varphi)=$ true，其中 $\varphi\in\Theta_\varphi$。M_φ 的初始条件 I_φ 和 M 的初始条件 I 实际上是一样的。

(3)变迁关系 R_φ 可以用变量集 $V_\varphi\cup V'_\varphi$ 上的公式 $\tau_\varphi(V_\varphi,V'_\varphi)$ 表示。

(4)π_φ 是相应的 W_φ 上的真值赋值函数，和 $w\in W_\varphi$ 相同。

(5)AP_φ 和 V_φ 相同。

(6)$K_{1\varphi},\cdots,K_{n\varphi}$ 是相应的 W_φ 上的等价关系。$K_{i\varphi}(w)=\{w'|(w,w')\in K_{i\varphi}\}$，即 $K_{i\varphi}(w)$ 是智能体 i 在状态 w 中认为可能的状态集合。虽然此时每个状态进行了扩展(能够对 $\vec{v}_\varphi$ 部分赋值)，但智能体 i 在状态 w 中认为可能的状态集合实际上并未发生改变。

令 J_φ 为公式 $\neg v_{\alpha U\beta}\vee\chi(\beta)$ 的集合，这里 $\alpha U\beta$ 是 φ 的一个子公式。我们考察如下公平性约束(fairness constraints)：

C_φ：存在一条路径，使得 J_φ 中的每个公式在该路径上成立无限多次。

因为可以把 true 加入 J_φ，不失一般性，我们假设 J_φ 是一非空集合。约束条件 C_φ 可以被定义为一个最大不动点函数：

$$C_\varphi=\upsilon Z.\left[\bigwedge_{J\in J_\varphi}EX(EU(\text{true},Z\wedge J))\right]$$

称 M_φ 中的路径 σ 是公平的，如果 J_φ 中的每个 J 沿着 σ 被满足无限多次。

由文献[9]可知一状态满足约束条件 C_φ 当且仅当该状态在某条公平路径上，该路径使得每个 $J\in J_\varphi$ 成立无限多次。

定理 4 令 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$ 为一 Kripke 结构，φ 是一个可能包含时态算子的公式，$M(\Gamma_\varphi)=(W_\varphi,I_\varphi,R_\varphi,AP_\varphi,\pi_\varphi,K_{1\varphi},\cdots,K_{n\varphi})$，则

$M(\Gamma),w|=E\varphi\Leftrightarrow\exists V_\varphi(C_\varphi\wedge\chi(\varphi))$。证明方法和文献[12]是类似的，这里不再赘述。另外我们有以下定理。

定理 5 令 φ 为不包含任何意向模态词的公式，对于 $M(\Gamma)$ 中的每个可能状态 w，有

$(M(\Gamma),w)|=K_i\varphi\Leftrightarrow\forall(V-O_i)(\theta\to\varphi)\Leftrightarrow\forall(V_\varphi-O_{i\varphi})(\theta_\varphi\to\chi(\varphi))$

证明：因为 θ_φ 是 Θ_φ 中所有布尔公式的合取，所以有 $\theta_\varphi=\chi(\theta)$。因此 $\theta_\varphi\to\chi(\varphi)=\chi(\theta\to\varphi)$；

设 $P=V-O_i$，又因为 $O_i=O_{i\varphi}$，且 O_i 和 $\vec{v}_\varphi$ 相互独立，则 $V_\varphi-O_{i\varphi}=V\cup\vec{v}_\varphi-O_i=(V-O_i)\cup(\vec{v}_\varphi-O_i)=P\cup\vec{v}_\varphi$；

因此 $\forall(V_\varphi-O_{i\varphi})(\theta_\varphi\to\chi(\varphi))\Leftrightarrow\forall(P\cup\vec{v}_\varphi)\chi(\theta\to\varphi)\Leftrightarrow\chi(\forall P(\theta\to\varphi))$

因为 $\vec{v}_\varphi$ 中的附加变量 v_ψ 在某个状态为真当且仅当 φ 的主时态子公式 ψ 在该状态成立，则一个公式在进行 χ 映射前后是等价的，表示基于状态空间的 Kripke 结构中的同一个状态集合，即 $\varphi\Leftrightarrow\chi(\varphi)$；因此 $\chi(\forall P(\theta\to\varphi))\Leftrightarrow\forall P(\theta\to\varphi)$，即

$\forall(V-O_i)(\theta\to\varphi)\Leftrightarrow\forall(V_\varphi-O_{i\varphi})(\theta_\varphi\to\chi(\varphi))$。

由定理 5 可知，构造 $M(\Gamma_\varphi)$ 不会影响检查意向公式 $K_i\varphi$ 的正确性。

定理 6 给定符号化 Kripke 结构 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$，$V'$ 为 V 的后继版本，φ 是一个可能包含时态算子的公式，$M(\Gamma_\varphi)=(W_\varphi,I_\varphi,R_\varphi,AP_\varphi,\pi_\varphi,K_{1\varphi},\cdots,K_{n\varphi})$。$f(V)$ 为 V 上的公式，$f(V_\varphi)=\chi(f(V))$ 为 V_φ 上的公式，则 $\exists V'(f(V')\wedge R)\Leftrightarrow\exists V'_\varphi(f(V'_\varphi)\wedge R_\varphi)$。

证明：$V_\varphi=V\cup\vec{v}_\varphi$，$\vec{v}_\varphi$ 中的附加变量 v_ψ 在某个状态为真当且仅当 φ 的主时态子公式 ψ 在该状态成立。在该前提下：

(1) 又因为 $f(V_\varphi)=\chi(f(V))$，所以 $f(V)$ 在某个状态为真当且仅当 $f(V_\varphi)$ 在该状态成立。

(2) 又由 $M(\Gamma_\varphi)$ 的构造方式可得，Kripke 结构 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$ 和 Kripke 结构 $M(\Gamma_\varphi)=(W_\varphi,I_\varphi,R_\varphi,AP_\varphi,\pi_\varphi,K_{1\varphi},\cdots,K_{n\varphi})$ 都是同一个基于状态空间的 Kripke 结构的等价表示。

那么 $f(V)$ 在 Kripke 结构 $M(\Gamma)$ 的映像 $R(f(V))$ 和 $f(V_\varphi)$ 在 Kripke 结构 $M(\Gamma_\varphi)$ 的映像 $R_\varphi(f(V_\varphi))$ 也是基于状态空间的 Kripke 结构中的同一个状态集合。而 $R(f(V))=\exists V'(f(V')\wedge R)$，$R_\varphi(f(V_\varphi))=\exists V'_\varphi(f(V'_\varphi)\wedge R_\varphi)$，因此 $\exists V'(f(V')\wedge R)\Leftrightarrow\exists V'_\varphi(f(V'_\varphi)\wedge R_\varphi)$。

由定理 6 知，构造 $M(\Gamma_\varphi)$ 不会影响检查 CTL 公式 $EX\varphi$ 的正确性。

定义 4 令 f 和 f' 为 CTL* 公式，称 f' 为 f 的最大子公式，如果有下列条件成立：

(1)f' 是 f 的子公式；

(2)f' 的形式为 Ef''，或者 f' 是一个原子命题；

(3)f' 至少有一次不出现在 f 中的 E 的作用范围内。

4 符号模型检查算法

令 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$ 为基于知识结构 Γ 的 Kripke 结构，f 是一个 CTL*K 公式。

符号模型检查函数 *check* 将一个 CTL*K 公式 f 作为其参数，返回在 Kripke 结构 $M(\Gamma)=(W,I,R,AP,\pi,K_1,\cdots,K_n)$ 中满足该公式的状态集合(可由 ROBDD 表示)。

function *check*(f)

$f=map(f)$;

case

$f\in AP$: return f;

$f=\neg\varphi$: return $\neg check(\varphi)$;

$f=\varphi\wedge\psi$: return $check(\varphi)\wedge check(\psi)$;

$f=EX\varphi$: return $checkEX(check(\varphi))$;

$f=E(\varphi U\psi)$: return $checkEU(check(\varphi),check(\psi),$ False);

$f=E\varphi$(φ 为 LTL 公式): return $\exists V_\varphi(GetCons(J_\varphi,\text{true})\wedge\chi(\varphi))$;

$f=E\varphi$(φ 不是 LTL 公式)：设 $f_1,f_2,\cdots,f_n$ 是 φ 中最大子公式的序列，且 $\varphi=\psi(f_1,f_2,\cdots,f_n)$，return $Check(\psi(Check(f_1),Check(f_2),\cdots,Check(f_n)))$;

end case

end function;

注意，*check* 函数中的 $f=E\varphi$ 情况实际上是包含 $f=EX\varphi$ 和 $f=E(\varphi U\psi)$ 两种情况的，但因为后两种情况中出现的是

CTL 公式，对于 CTL 公式的模型检查，可以直接根据定理 2 和 3 行求解，不需要构造 Kripke 结构 $M(\Gamma_\varphi)=(W_\varphi,R_\varphi,AP_\varphi,\pi_\varphi,K_{1\varphi},\cdots,K_{n\varphi})$。将这两种情况单独列出，算法的效率自然会更高。

出现在 *check* 函数中的 *map* 函数把 f 转化为一个除算子 E,X 和 U 外不包含别的模态算子的 CTL * 公式，其伪代码如下：

function $map(f)$

case

$f=g$（g 为除算子 E,X 和 U 外，不包含其它模态算子的 CTL * 公式）：return g；

$f=Ag$：return $map(\neg E\neg g)$；

$f=Fg$：return $map(\text{true}Ug)$；

$f=Gg$：return $map(\neg F\neg g)$；

$f=K_i\varphi$（φ 为不包含任何意向模态词的公式）：return $map(\forall(V-O_i)(\theta\rightarrow\varphi))$；

$f=A-G_i\varphi$（φ 为不包含任何意向模态词的公式）：return $map(K_i(\neg\varphi)\wedge AF(K_i\varphi))$；

$f=M-G_i\varphi$（φ 为不包含任何意向模态词的公式）：return $map(K_i(\neg\varphi)\wedge AF(K_i(G\varphi))$；

设 $f_1,f_2,\cdots,f_n$ 是 φ 中最大子公式序列，且 $f=\psi(f_1,f_2,\cdots,f_n)$，

return $\psi(map(f_1),map(f_2),\cdots,map(f_n))$；

end case

end function

函数 *checkEX* 的伪代码分别为：

function $checkEX(\varphi(V))=\exists V'(\varphi(V')\wedge R)$；

由定理 5 可知，如果 *checkEX* 的输入是 $\psi(V_\varphi)$，则结果为 $\exists V_\varphi{}'(\psi(V_\varphi{}')\wedge R_\varphi)$。这一点在下面计算公平性约束 C_φ 的函数 *GetCons* 中需要注意。

函数 *checkEU* 的伪代码分别为：

function $checkEU(\varphi,\psi,b)$

$b'=\varphi\vee(\psi\wedge checkEX(b))$；

if $b'=b$

then return b

else return $checkEU(\varphi,\psi,b')$

end function

通过 *CheckEX* 和 *CheckEU* 和计算最大不动点的算法[9]，可得到计算公平性约束 C_φ 的函数 *GetCons*。*GetCons* 的主要输入是公平性公式集合 J_φ，伪代码如下：

function $GetCons(J_\varphi,b)$

$a:=$true；

for $i:=1$ to count of J_φ do

$a:=a\wedge checkEX(checkEU(\text{true},b\wedge J_\varphi[i],\text{false}))$；

$b':=a$；

if $b'=b$

then return b

else return $GetCons(J_\varphi,b')$；

end function

5 案例研究

下面用 CTL * K 逻辑对译解密码者问题进行建模。译解密码者问题由 Chaum 在文献[13]中进行了介绍，而文献[14,15]中也都曾用智能体方法对其进行建模。译解密码者之间允许进行匿名消息传递，其场景如下[13]：

“三个译解密码者坐在他们所喜爱的三星级餐厅里就餐。最后可以由某个译解密码者进行匿名付款，也可以通过 NSA（美国国家安全局）来付款。三个译解密码者彼此尊重匿名付款的权利，但他们又想知道是否是通过 NSA 进行付款。因此通过执行如下协议进行解决：

每个译解密码者在他和他右边的译解密码者之间用菜单挡着掷硬币，结果只有他们两个人看到。然后，每个译解密码者大声说出他所看到的两枚硬币（他自己掷的和他左边的人掷的）是否是同一面。但如果某个译解密码者是付款的人，他所说出的要和他所看到的结果相反。如果最后听到结果为不同面的次数是奇数，表明是通过某个译解密码者进行付款；如果是偶数，则说明是通过 NSA 付款。通过这种方式，即使是某个译解密码者进行付款，另外两个人也无法知道具体是谁在付款。”

该协议可以扩展到大于 3 个人的任意 n 个译解密码者。对问题的建模，首先引入 n 个智能体 $C_1,\cdots,C_n$。其中 C_i 有 3 个可观察变量 $equal_i,paid_i,even_i$，分别表示 C_i 看到的硬币面是否是相同的、C_i 是否是付款的人以及最后听到的不同面的次数是否是偶数，并且都初始化为空值。当每个译解密码者喊出结果时，对这些变量进行更新。其中 $even_1=even_2=even_3$。该问题的关键属性[17]很容易用 CTL * K 进行表达。例如：

$$(\neg even_i\wedge\neg paid_1)\rightarrow AX(K_{C1}(paid_2\vee paid_3)\wedge\neg K_{C1}(paid_2)\wedge\neg K_{C1}(paid_3)) \quad (1)$$

$$even_i\rightarrow AX(K_{C1}(\neg paid_2\wedge\neg paid_3)) \quad (2)$$

其中，式(1)表示，如果第一个译解密码者没有付款，并且最后听到的结果为奇数，那么更新他的局部变量，第一个译解密码者知道另外两个人付了款，但不知道究竟是谁。式(2)表示，如果最后听到的结果为偶数，第一个译解密码者知道没有人付款。

通过对 NuSMV[16]提供的开放源码进行扩充，添加进文中给出的 CTL * K 的模型检查算法，可以对上述公式进行检查，并且检查了从 3 个人到 9 个人的不同情况，结果如表 1 所列。其中第 1 列给出了检查的译解密码者人数；第 2 列给出了检查时所需要的布尔变量数目（参见文献[9]中的技术细节），并提供了对模型大小的评估。例如对 7 个译解密码者进行编码需要 134 个布尔变量，对应于模型的最大状态空间为 $2^{134}\approx10^{40}$；第 3 列给出了检查公式(1)和(2)所需要的时间。

表 1 实验结果

译解密码者的人数	布尔变量数目	检查时间
3	58	0.31 秒
4	75	0.43 秒
5	96	0.79 秒
6	113	2.90 秒
7	134	59 秒
8	153	6 分钟 27 秒
9	172	48 分钟 39 秒

结束语 本文主要讨论了基于 CTL * 的时态认知逻辑 CTL * K 的语法、语义和模型，能够在表达力很强的时态逻辑 CTL * 基础上描述智能体的知识、目标等意向特征。并给出

了 CTL * K 的模型检查算法，其核心思想就是将 CTL * K 公式的检查问题转化为 CTL * 公式的模型检查问题。这种算法不需要显式地构建状态空间，可以使检查的系统规模得以大幅度提高。并且将算法编码后，容易集成到 NuSMV 模型检查器，从而使只能够对 CTL 进行检查的 NuSMV 也可以对 CTL * K 进行检查。

参 考 文 献

[1] Mcmillan K L. Symbolic model checking : 1 0^{20} states and beyond[J]. Information and Computation, 1992, 98(2): 142-170

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

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

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

[5] 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[J]. 软件学报，2004, 15 (7): 1012-1020

[6] 吴立军，苏开乐，等. 多主体系统时态认知规范的"On the Fly"模型检测算法研究[J]. 计算机研究与发展，2006, 43 (8): 1417-1424

[7] Emerson E A, Lei Chin - laung. Modalities for model checking (extended abstract): Branching time strikes back[C]// Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. New York, 1985: 84-96

[8] Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons using branching-time temporal logic[C]// Proceedings of Logic of Programs Workshop. Yorktown Heights, New York, 1981: 52-71

[9] Grumberg O, Clarke E M, Peled D A. Model Checking. Cambridge[M] MA: The MIT Press, 2000

[10] Schnoebelen P h. The Complexity of Temporal Logic Model Checking[J]. Advances in Modal Logic, 2002, 4: 1-44

[11] Su Kaile, LüGuanfeng, Chen Qingliang. Knowledge structure approach to verification of authentication protocols[J]. Science in China (Series E), 2005, 35 (4): 337 -351

[12] 苏开乐，骆翔宇，吕关锋. 符号化模型检测 CTL * [J]. 计算机学报，2005, 28(11): 1798-1805

[13] Chaum D. The dining cryptographers problem : Unconditional sender and recipient untraceability[J]. Journal of Cryptology, 1988, 1(1): 65-75

[14] van der Meyden R, Su Kaile. Symbolic model checking the knowledge of the dining cryptographers[C]// Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW' 04). Washington, DC, USA, IEEE Computer Society, 2004: 280-291

[15] Kacprzak M, Lomuscio A, Niewiadomski A, et al. Comparing BDD and SAT based techniques for model checking Chaum's dining cryptographers protocol[J]. Fundamenta Informaticae, 2006

[16] Climatti A, Clarke E M, Giuchiglia E. NuSMV Version 2: An OpenSource tool for symbolic model checking[C]// Proceedings of the International Conference on Computer-Aided Verification (CAV 2002). Copenhagen, Denmark, 2002: 359-364

(上接第 205 页)

高，从另一个侧面说明了聚类结果更合理。

WDBC 数据集上的实验结果同样显示出了改进的粗糙 Leader 算法的优越性，聚类的纯度和精度都更高，而且聚类结果受样本排列顺序的影响相对较小。

结束语 聚类分析是数据挖掘领域一个非常活跃的研究课题。经典的 Leader 聚类算法有很多优点，例如只需扫描一遍数据集且无需指定簇的个数。然而 Leader 算法也存在着一些缺点，例如没有考虑到聚类分析中固有的不确定性，聚类的结果受样本排列顺序的影响较大。

本文在现有研究基础之上提出了一种改进的基于粗糙集和粒计算思想的 Leader 聚类算法。该算法使用粗糙集中上、下近似之差，即边界区域来捕捉聚类分析中内在的不确定性，对边界区域中的样本进行迭代聚类，并通过调整上、下近似的阈值对边界区域进行细分，细化其聚类的粒度。实验验证了改进的粗糙 Leader 算法相对于经典的 Leader 算法的优越性，不但聚类结果更加合理，且不易受样本排列顺序的影响。

参 考 文 献

[1] Han Jiawei, Kamber M. Data Mining Concepts and Techniques [M]. 范明，孟小峰，译. 北京：机械工业出版社，2001

[2] Theodoridis S, Koutroumbas K. Pattern Recognition[M]. 李晶皎，译. 北京：电子工业出版社，2004

[3] Spath H. Cluster Analysis Algorithms for Data Reduction and Classification of Objects[M]. Chichester: Ellis Horwood Limited, 1980

[4] Gowda K C, Diday E. Symbolic Clustering Using a New Dissimilarity Measure[J]. Pattern Recognition, 1991, 24(6): 567-578

[5] Asharaf S, Murty M N, Shevade S K. Rough Set Based Incremental Clustering of Interval Data[J]. Pattern Recognition Letters, 2006, 27(6): 515-519

[6] Pawlak Z. Rough Sets [J]. International Journal of Computer and Information Sciences, 1982, 11(2): 341-356

[7] 王珏，苗夺谦，等. 关于 Rough Set 理论与应用的综述[J]. 模式识别与人工智能，1996, 9(4): 337-344

[8] Lin T Y, Zadeh L A. Special Issue on Granular Computing and Data Mining[J]. International Journal of Intelligent Systems, 2004, 19(7): 565-566

[9] 苗夺谦，王国胤，姚一豫，等. 粒计算：过去、现在和展望[M]. 北京：科学出版社，2007

[10] Blake C L, Merz C J. UCI Repository of Machine Learning Database[EB/OL]. 2007. http://archive.ics.uci.edu/ml

[11] Tao L, Zheng C. An Evaluation on Feature Selection for Text Clustering[C]// Proceedings of the 20th International Conference on Machine Learning. Washington D. C., 2003

[12] 刘涛，陈正. 一种高效的用于文本聚类的无监督特征选择算法[J]. 计算机研究与发展，2005, 42(3): 381-386