基于标记 Büchi 自动机的时态描述逻辑 *ALC*-LTL 模型检测

朱创营 常 亮 徐周波 李凤英
(桂林电子科技大学广西可信软件重点实验室 桂林 541004)

摘 要 时态描述逻辑将描述逻辑的刻画能力引入到命题时态逻辑中,适合于在语义 Web 环境下对相关系统的时态性质进行刻画。为了对这些时态性质进行高效的验证,在 *ALC*-LTL 的基础上研究了时态描述逻辑的模型检测问题。一方面,使用时态描述逻辑 *ALC*-LTL 公式来表示待验证的时态规范;另一方面,在对系统建模时借助描述逻辑 *ALC* 对领域知识进行刻画。针对上述扩展后得到的模型检测问题,提出了基于自动机的 *ALC*-LTL 模型检测算法。模型检测算法由 3 个阶段组成:首先将时态规范的否定形式和系统模型分别构造成标记 büchi 自动机;接下来构造这两个自动机的乘积自动机,并将关于 *ALC* 的推理机制融入到乘积自动机的构造过程中;最后对该乘积自动机进行判空检测。与 LTL 模型检测相比,时态描述逻辑 *ALC*-LTL 的模型检测引入了描述逻辑的刻画和推理机制,可以在语义 Web 环境下对语义 Web 服务等复杂系统的时态性质进行刻画和验证。

关键词 线性时态描述逻辑,模型检测,标记 büchi 自动机,*ALC*-类,乘积自动机,判空问题,语义 Web
中图法分类号 TP301 **文献标识码** A

Model Checking of Temporal Description Logic *ALC*-LTL Based on Label Büchi Automata

ZHU Chuang-ying CHANG Liang XU Zhou-bo LI Feng-ying
(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract The temporal description logic introduces the description abilities of the description logic into the proposition temporal logic. It's suitable for describing the temporal properties of relevant systems under the semantic Web environment. To verify the temporal properties efficiently, the model checking problem of temporal description logic based on *ALC*-LTL was investigated in this paper. On the one hand, the temporal description logic *ALC*-LTL formula was used to express the specification to be checked. On the other hand, the description logic *ALC* was used to model the system which is investigated. For the resulted model checking problems, a model checking algorithm based on label büchi automation was presented. This algorithm composes of three steps. Firstly, the negation of the specification and the model of the system are constructed as two separate label büchi automation, and then constructing a product automation for the two label büchi automations, and the reasoning mechanisms of *ALC* is embedded in the process of constructing in this step, finally, detecting the emptiness problem for the product automation. Compared with the model checking of propositional temporal logic LTL, the model checking of *ALC*-LTL is provided with the representation ability and reasoning mechanisms of description logics, and therefore is suitable for the semantic Web environment.

Keywords Linear temporal description logic, Model checking, Label büchi automation, *ALC*-type, Product automation, Emptiness problem, Semantic Web

1 引言

模型检测[1]是一种被广泛应用的验证有限状态系统性质的自动化验证技术。该技术一方面基于状态转换系统对所考察的系统进行建模,另一方面应用线性时态逻辑 LTL 或分支时态逻辑 CTL 等对所需验证的性质进行刻画;在此基础上,通过对状态空间的穷举搜索来判断系统是否满足由时态逻辑公式所刻画的性质,若不满足则返回一条违反规范的路径(即反例),以便开发者对系统进行改进。在以 E. M. Clarke 和 E. A. Emerson 等为代表的研究者的努力下,时态描述逻辑 CTL 和 LTL 的模型检测问题已经得到了很好的解决,其不仅具有高效的搜索算法,而且具有 SPIN、SMV 等相应的验证工具予以支撑,能够自动化地进行验证。因此,以 CTL 和 LTL 为代表的时态逻辑模型检测技术在通讯协议验证、计算机系统验证等方面得到了广泛的应用。

为了将模型检测技术应用于更广泛的领域,在提出时态

到稿日期:2012-12-29 返修日期:2013-03-15 本文受国家自然科学基金(61363030,61100025,61262030),广西自然科学基金(2012GXNSFBA053169,2012GXNSFAA053220)资助。
朱创营(1986—),男,硕士生,主要研究方向为形式化验证、语义 Web 服务和信息安全,E-mail:zhuchuangying@yahoo.com.cn;常 亮(1980—),男,博士,教授,CCF 高级会员,主要研究方向为知识表示与推理、形式化方法、智能规划等;徐周波(1976—),女,博士,副教授,主要研究方向为符号计算、智能规划等;李凤英(1974—),女,博士,副教授,主要研究方向为形式化方法、智能规划等。

逻辑的各种扩展形式之后，研究者相应地对这些扩展形式的模型检测问题进行了研究。例如，知识逻辑与时态逻辑相结合的时态认知逻辑在分布式系统中具有广泛应用。对于基于LTL的时态认知逻辑CKLn，2002年Wiebe van der Hoek等[2]提出了借助LTL模型检测算法实现CKLn模型检测的方法，2004年Ron van der Meyden等基于OBDD技术开发出时态知识逻辑模型检测工具MCK[3]。对于基于CTL的时态认知逻辑CKCn，吴立军等提出了基于OBDD的模型检测算法[4]，并在CTL模型检测工具NuSMV的基础上开发了支持CKCn的模型检测工具MCTK。

无论是在传统时态逻辑CTL、LTL的模型检测中，还是在将它们扩展后得到的时态认知逻辑CKLn等的模型检测中，一个共同的特点是它们都基于命题逻辑对模型的状态以及时态规范中的原子性质进行描述。由于命题逻辑的描述能力有限，使得对相关知识尤其是领域知识的刻画受到了很大的限制。

相对于命题逻辑来说，描述逻辑[5]具有更强的表达能力，同时又保持了可判定性，并且有有效的判定算法和推理机制作为支撑。特别是近年来，描述逻辑成为了Web本体语言OWL的逻辑基础，是语义Web环境下进行知识表示和知识推理的关键所在。

对描述逻辑进行时态扩展是近年来描述逻辑领域的一个研究热点。通过采用不同的方式在描述逻辑中引入时态维，研究者提出和构造了各种不同类型的时态描述逻辑[6,7]，并证明了这些时态描述逻辑中的公式可满足性问题仍然是可判定的。从描述逻辑的角度看，时态描述逻辑将描述逻辑针对静态领域的知识表示和推理能力扩展到了动态领域。

目前研究者关于描述逻辑与时态逻辑相结合的研究工作绝大部分都是从描述逻辑的角度来进行的，动机是增强描述逻辑的刻画能力，关注的问题主要是公式的可满足性问题。但实际上，从时态逻辑的角度来看，时态描述逻辑将LTL、CTL等时态逻辑中使用的命题逻辑提升到了描述逻辑的层面，在很大程度上扩展了时态逻辑的应用范围。尤其是，鉴于描述逻辑在Web本体语言和语义Web中所扮演的核心角色，时态描述逻辑尤其适用于在语义Web环境下对相关系统的时态性质进行刻画。而为了对这些时态规则进行验证和推理，首先有必要研究时态描述逻辑的模型检测问题。

本文对线性时态描述逻辑*ALC*-LTL的模型检测问题进行研究，提出基于自动机的模型检测方法。一方面使用Baader等[8]提出的时态描述逻辑*ALC*-LTL对时态规范进行刻画；另一方面参照*ALC*-LTL的解释结构引入状态迁移系统，作为待检测的模型。针对由此产生的模型检测问题，本文将描述逻辑*ALC*的推理机制与时态逻辑LTL的模型检测结合起来，给出了相应的算法。首先，将系统模型构造成一个标记büchi自动机，根据*ALC*-类将时态规范的否定形式同样构造成一个标记büchi自动机，然后构造出这两个自动机的乘积自动机，并将*ALC*的推理问题融入到构造乘积自动机的过程中，最后对该乘积自动机进行判空问题检测。如果时态规范得不到验证，则返回一条违反时态规范的路径。

本文第2节首先对时态描述逻辑*ALC*-LTL进行简单介绍，然后参照*ALC*-LTL的解释结构引入基于描述逻辑的状态迁移系统；第3节首先引入标记büchi自动机，然后描述如何根据*ALC*-LTL公式构造标记büchi自动机，在此基础上对乘积自动机的构造过程和判空问题进行考察，得到完整的模型检测算法；第4节对相关工作进行比较；最后总结全文。

2 系统建模

2.1 时态描述逻辑 *ALC*-LTL

本文将待验证规范建模为Baader等提出的时态描述逻辑*ALC*-LTL[8]公式。本节对*ALC*-LTL进行简单介绍。从时态逻辑的角度看，*ALC*-LTL是在命题线性时态逻辑LTL中引入描述逻辑*ALC*的刻画成分之后得到的逻辑系统。

ALC-LTL的基本符号包括由概念名组成的集合N_C、由角色名组成的集合N_R以及由个体名组成的集合N_I。从这些符号出发，通过描述逻辑*ALC*中的概念构造符和线性时态逻辑LTL中的公式构造符，可以递归地生成*ALC*-LTL的概念和公式。

定义1 *ALC*-LTL中的概念由如下产生式生成：

$$C,D::=C_i\mid\neg C\mid C\sqcup D\mid\forall R.C$$

其中，$C_i\in N_C$，$R\in N_R$。$\neg C$、$C\sqcup D$、$\forall R.C$的概念分别称为否定、析取以及值限定。

此外，可以引入形如$C\sqcap D$、$\exists R.C$、$\top$以及$\bot$的概念，分别作为$\neg(\neg C\sqcup\neg D)$、$\neg(\forall R.\neg C)$、$C\sqcup\neg C$以及$\neg\top$的缩写，分别称为合取、存在性限定、全概念以及空概念。

令C、D为任意两个概念，$R\in N_R$，$p,q\in N_I$，则将$C\sqsubseteq D$称为一般概念包含公理，将$C(p)$称为概念断言，将$R(p,q)$称为角色断言。$C(p)$和$R(p,q)$也可分别记为$p:C$和pRq。

将一般包含公理、概念断言与角色断言统称为*ALC*-公理。*ALC*-公理是构成*ALC*-LTL公式的最基本的原子公式。

定义2 *ALC*-LTL公式通过下列产生式递归生成：

$$\varphi,\psi::=C\sqsubseteq D\mid C(p)\mid R(p,q)\mid\neg\varphi\mid\varphi\wedge\psi\mid\boldsymbol{X}\varphi\mid\varphi\boldsymbol{U}\psi$$

式中，$p,q\in N_I$，$R\in N_R$，C、D为概念。

此外，也可以依次引入形如$\varphi\vee\psi$、$\varphi\rightarrow\psi$、$\boldsymbol{F}\varphi$以及$\boldsymbol{G}\varphi$的公式，分别作为$\neg(\neg\varphi\wedge\neg\psi)$、$\neg\varphi\vee\psi$、$\mathrm{true}\,\boldsymbol{U}\varphi$和$\neg\boldsymbol{F}\neg\varphi$的缩写。

在语义方面，*ALC*-LTL公式的解释结构与LTL公式解释结构类似。随着时间的推进，将各个状态线性地连接起来。不同的是，*ALC*-LTL公式的解释结构将每个状态映射为一个*ALC*的一个解释而不单纯的是一个原子命题集合。

定义3 *ALC*-LTL解释结构是一个二元组$\Im=(\mathrm{N},I)$，其中：

(1)N为自然数集合；

(2)函数I对每个自然数$n\in\mathrm{N}$赋予描述逻辑*ALC*的一个解释$I(n)=(\Delta,\cdot^{I(n)})$，其中的解释函数$\cdot^{I(n)}$满足以下条件：

(i)将每个概念名$C_i\in N_C$解释为Δ的某个子集$C_i^{I(n)}\subseteq\Delta$；

(ii)将每个角色名$R_i\in N_R$解释为Δ上的某个二元关系$R_i^{I(n)}\subseteq\Delta\times\Delta$；

(iii)将每个个体名$p_i\in N_I$解释为Δ中的某个元素$p_i^{I(n)}\in\Delta$，并且对于任一自然数$m\in\mathrm{N}$都有$p_i^{I(n)}=p_i^{I(m)}$。

定义4 给定任一*ALC*-LTL解释结构$\Im=(\mathrm{N},I)$，对*ALC*-LTL中概念和公式的语义递归定义如下。

首先，相对于任一自然数$i\in\mathrm{N}$，将每个概念C解释为Δ

的某个子集 $C^{I(i)}$；递归定义为：

(1)$(\neg C)^{I(i)}:=\Delta\backslash C^{I(i)}$，其中的“\”为集合差运算；

(2)$(C\sqcup D)^{I(i)}:=C^{I(i)}\cup D^{I(i)}$，其中的“∪”为集合并运算；

(3)$(\forall R.C)^{I(i)}:=\{x\mid$ 对于任一 $y\in\Delta$：如果$(x,y)\in R^{I(i)}$，则必然有 $y\in C^{I(i)}\}$。

其次，相对于任一自然数 $n\in\mathrm{N}$，用$(\Im,i)\models\varphi$ 表示公式 φ 在结构$\Im$中的时间点 i 下成立，递归定义如下：

(4)$(\Im,i)\models C\sqsubseteq D$ iff $C^{I(i)}\subseteq D^{I(i)}$；

(5)$(\Im,i)\models C(p)$ iff $p^{I(i)}\in C^{I(i)}$；

(6)$(\Im,i)\models R(p,q)$ iff$(p^{I(i)},q^{I(i)})\in R^{I(i)}$；

(7)$(\Im,i)\models\neg\varphi$ iff$(\Im,i)\not\models\varphi$(即公式 φ 在解释结构$\Im$中的时间点 i 中不成立)；

(8)$(\Im,i)\models\varphi\wedge\psi$ iff$(\Im,i)\models\varphi$ 并且$(\Im,i)\models\psi$；

(9)$(\Im,i)\models\boldsymbol{X}\varphi$ iff$(\Im,i+1)\models\varphi$；

(10)$(\Im,i)\models\varphi\boldsymbol{U}\psi$ iff 存在某个整数 $k\geqslant0$，使得$(\Im,i+k)\models\psi$，并且对于任意 $0\leqslant j<k$ 都有$(\Im,i+j)\models\varphi$。

时态描述逻辑 ALC-LTL 中的知识库与描述逻辑 ALC 中的知识库相同，由 TBox 和 ABox 组成。其中的 TBox 是由一般概念包含公理组成的有限集合；ABox 是由概念断言、角色断言及其否定形式组成的有限集合。

令 $\boldsymbol{T}$ 是一个 TBox，$\boldsymbol{A}$ 是一个 ABox，那么称知识库$\langle\boldsymbol{T},\boldsymbol{A}\rangle$是一致的当且仅当存在描述逻辑 ALC 的一个语义解释 $I=(\Delta,\cdot^{I})$使得 $I\models\boldsymbol{T}$ 并且 $I\models\boldsymbol{A}$，也记为 $I\models\langle\boldsymbol{T},\boldsymbol{A}\rangle$。

令$\langle\boldsymbol{T},\boldsymbol{A}\rangle$是一个知识库，$p$ 是一个 ALC-公理。如果对于描述逻辑 ALC 的任一语义解释 $I=(\Delta,\cdot^{I})$来说，当 $I\models\langle\boldsymbol{T},\boldsymbol{A}\rangle$时必然有 $I\models p$，则称知识库$\langle\boldsymbol{T},\boldsymbol{A}\rangle$能推导出 p，记为$\langle\boldsymbol{T},\boldsymbol{A}\rangle\models p$。

2.2 系统模型

在本文中，将待考察的系统建模为一个基于描述逻辑 ALC 的状态迁移系统。定义如下：

定义 5 基于描述逻辑 ALC 的状态迁移系统是一个六元组 $\mathrm{M}=(T,S,s_0,R,Label,F)$，其中：$T$ 为描述逻辑 ALC 中的一个 TBox，对系统中涉及的领域知识进行描述，这些知识不随着时间的变化而变化；S 为一个有限的非空状态集；s_0 表示初始状态；$R:S\to S$ 为一个状态转移函数，使得任一状态 $s\in S$ 都存在一个唯一的直接后继状态 $R(s)$；$Label$ 是标记函数，将每个状态 $s\in S$ 映射到描述逻辑 ALC 中的一个 ABox；F 是在模型中无限多次出现的状态的集合。

由于任一状态 $s\in S$ 都存在唯一后继 $R(s)$，因此上述状态迁移系统是一个有限状态的无限序列：s、$R(s)$、$R(R(s))$、$R(R(R(s)))\cdots$。为了表述方便，在后面的内容中将初始状态及其后继状态依次记为 $s_0,s_1,s_2\cdots$。

定义 6 令 $\mathrm{M}=(T,S,s_0,R,Label,F)$ 是基于描述逻辑 ALC 的一个状态迁移系统。如果存在一个 ALC-LTL 解释结构$\Im=(\mathrm{N},I)$，使得对于任一自然数 $i\in\mathrm{N}$ 都有 $I(i)\models\langle T,Label(s_i)\rangle$，则将$\Im=(N,\mathrm{I})$称为状态迁移系统的一个解释，记为 $\Im\models\mathrm{M}$。

定义 7 对于给定状态迁移系统 $\mathrm{M}=(T,S,s_0,R,Label,F)$和时态规范 φ，如果对于 M 的任一解释$\Im=(\mathrm{N},I)$都有$(\Im,0)\models\varphi$，那么称时态规范 φ 在模型 M 中成立，记为 $\mathrm{M}\models\varphi$。

2.3 案例

为了便于对后面的算法进行介绍，这里选取一段中国历史作为案例进行研究说明，以验证在这段历史中是否具有某种规律——我国历史上的唐王朝历经 22 位皇帝、23 次政权交替、24 个执政时期。这 22 位皇帝上台执政或是因为接受禅让或是因为发动政变夺取皇位，在这里仅抽出几位皇帝，按照他们在位顺序将其列举，如图 1 所示。

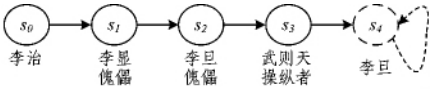

图 1 唐朝皇帝执政顺序

我们知道这几位皇帝具有如下的血缘关系：

父子关系：李治→李显；李治→李旦。

母子关系：武则天→李显；武则天→李旦。

在这个案例中，将傀儡政权的操纵者也视为是政变者。在此基础上来判断是否能从这段历史中得到如下结论：如果现任皇帝的太子不是下任皇帝，则新君一定是发动政变获取的皇位。

简便起见，假设武则天之后的皇帝的状态不再改变。

对于该案例，在调用模型检测算法之前，首先需要将案例中所需要用到的领域知识建模为一个术语集 Tbox。如表 1 所列。

表 1 领域知识

Tbox **T**	emperor, nextemperor, coupmakers, manipulator, haschild$\subseteq\top$; manipulator$\subseteq$coupmakers

对于这段历史，根据表 1 中定义的术语，进一步将其建模为一个基于描述逻辑的状态迁移系统 M。为方便起见，文中用 $s_0,\cdots,s_4$ 来表示从李治开始的各皇帝执政时期的状态，那么有：

$R(s_i)=s_{i+1}(0\leqslant i<4)$；$R(s_4)=s_4$；

$Label(s_0)=\{$haschild(emperor, nextemperor)$\}$；

$Label(s_1)=\{\}$；

$Label(s_2)=\{\}$；

$Label(s_3)=\{$manipulator(emperor)；haschild(emperor, nextemperor)$\}$；

$Label(s_4)=\{$haschild(emperor, nextemperor)$\}$；

$F=\{s_4\}$。

对于所考察的结论，可以将其形式化描述为：

$$\boldsymbol{G}(\neg\text{haschild(emperor, nextemperor)}\to\boldsymbol{X}\text{coupmakers(emperor))}\tag{1}$$

3 模型检测算法

在系统模型中验证某个时态公式是否成立，可以通过构造自动机的方法来实现。在这里首先引入标记 büchi 自动机的概念。

定义 8 一个标记 büchi 自动机是一个六元组$\mathcal{G}=(\mathrm{S},\Sigma,S_0,\mathrm{F},\rho,\ell)$，其中：$S$ 为有限非空状态集；Σ 为非空符号集；$S_0\subseteq2^S$ 为非空起始状态集；$\mathrm{F}\subseteq2^S$ 是一个接受状态集；$\rho:S\to2^S$ 是转移函数；$\ell:S\to\Sigma$ 是状态标记函数。

我们用上标符号 ω 表示无限序列。

定义 9 标记 büchi 自动机的一个运行是一个由状态组成的无限序列 $\sigma=s_0,s_1,s_2,\cdots\in S^{\omega}$。其中，$s_0\in S_0$，$s_i\in S$ 且

$s_i \rightarrow s_{i+1}(i \geqslant 0)$。由运行中每个状态的标记所构成的无限序列 $\omega = a_0 a_1 a_2 \cdots \in \Sigma^{\omega}$（其中对于 $i \geqslant 0, a_i \in \ell(s_i)$）称为关于该运行 σ 的词语，σ 称为关于词语 ω 的运行。

给定标记 büchi 自动机 $\mathcal{G}$ 上的一个运行 $\sigma = s_0, s_1, s_2, \cdots$，令 $\lim(\sigma)$ 是出现在运行 σ 中无限多次的状态的集合，那么我们称这个运行 σ 是被自动机 $\mathcal{G}$ 所接受的，当且仅当 $\lim(\sigma) \cap \mathrm{F} \neq \emptyset$，即在运行 σ 中存在有无限循环的状态属于 F。令词语 $\omega = a_0 a_1 a_2 \cdots$（其中对于 $i \geqslant 0, a_i \in \ell(s_i)$）是关于该运行 σ 的词语。那么称词语 ω 是被自动机 $\mathcal{G}$ 所接受的词语。将所有被自动机 $\mathcal{G}$ 所接受的词语的集合称为 $\mathcal{G}$ 的语言，记为 $\mathrm{L}_{\omega}(\mathcal{G})$。

在基于自动机的 LTL 模型检测算法中，对于任一 LTL 公式 φ 都可以构造出一个标记 büchi 自动机 $\mathcal{G}$，使得 $\mathcal{G}$ 的语言 $\mathrm{L}_{\omega}(\mathcal{G})$ 等于满足 φ 的原子命题的序列集合。类似地，对于任意一个 *ALC*-LTL 公式 φ，如果能够基于 φ 构造一个标记 büchi 自动机 $\mathcal{G}$，并且 $\mathcal{G}$ 的语言 $\mathrm{L}_{\omega}(\mathcal{G})$ 等于满足 φ 的 *ALC*-公理的序列集合，即 $\mathcal{G}$ 恰好能接受所有满足规范性质的运行，那么就能够基于自动机实现对 φ 的模型检测。下面给出根据 *ALC*-LTL 公式构造标记 büchi 自动机的方法。

3.1 为 *ALC*-LTL 公式构造标记 büchi 自动机

任意一个 *ALC*-LTL 公式都存在有无限多个解释，因此要为 *ALC*-LTL 公式构造一个能够满足所有解释的标记 büchi 自动机，就要构造一个无限的字符集，这显然是不现实的。为解决这个问题，可以将每个 *ALC*-LTL 公式的解释进行分类，只考虑它们的特殊解释——*ALC*-类。

我们将公式 φ 中出现的 *ALC*-公理称为 φ-公理，φ-公理及其否定称为 φ-文字。

定义 10 φ-文字的集合 T 是公式 φ 的一个 *ALC*-类，当且仅当下列两个性质成立：

(1)对每个 φ-文字 $\alpha, \alpha \in T$，当且仅当 $\neg\alpha \notin T$。

(2)*ALC*-布尔知识库是一致的，即 $\mathrm{B}_T = \wedge_{\alpha \in T} \alpha$ 是一致的。

将公式 φ 所有的 *ALC*-类记为 Σ^{φ}。

定义 11 给定某个 *ALC*-LTL 知识库的一个解释 I，它的 φ-类 $\tau_{\varphi}(I)$ 为：该知识库在 I 的解释下，所有具有相同解释的 φ-文字的集合。

容易看出，$\tau_{\varphi}(I)$ 是 φ 的一个 *ALC*-类。相反地，给定 φ 的一个 *ALC*-类 T，将布尔知识库 B_T 的模型 I 表示为 $\tau_{\varphi}(I) = T$。

根据以上定义，对时态规范的解释结构进行处理就可以抽象为对该结构中 φ-类的处理。给定一个 *ALC*-LTL 解释结构 $\mathfrak{I} = (\mathrm{N}, I)$，它的 φ-类是基于 Σ^{φ} 的无限字 $\tau_{\varphi}(\mathfrak{I}) = \tau_{\varphi}(I_0)\tau_{\varphi}(I_1)\tau_{\varphi}(I_2)\cdots$。

在为时态规范 φ 构造自动机 $\mathcal{G}_{\varphi}$ 时，需要根据 φ 的结构进行分解，这里将 φ 的子公式和其子公式的否定称为 φ 的子文字。

定义 12 时态规范 φ 的子文字的集合 T 称为 φ 的一个 *ALC*-LTL-类，当且仅当下面的性质成立：

(1)对于 φ 的每一个子公式 $\psi \in T$，当且仅当 $\neg\psi \notin T$；

(2)对于 φ 的每一个子公式 $\psi_1 \wedge \psi_2$，我们有 $\psi_1 \wedge \psi_2 \in T$，当且仅当 $\{\psi_1, \psi_2\} \subseteq T$；

(3)对于 φ 的每一个子公式 $\psi_1 \boldsymbol{U} \psi_2$ 有：

(i)$\psi_2 \in T \Rightarrow \psi_1 \boldsymbol{U} \psi_2 \in T$，

(ii)$\psi_1 \boldsymbol{U} \psi_2 \in T$ 并且 $\psi_2 \notin T \Rightarrow \psi_1 \in T$；

(3)T 对于其 *ALC*-LTL-类的限制与 φ 的 *ALC*-类的限制相同。

将时态规范 φ 所有的 *ALC*-LTL-类集合记为 Q^{φ}。

在构造 büchi 自动机的过程中，需要用到广义标记 büchi 自动机的概念。

定义 13 广义标记 büchi 自动机是一个六元组 $\mathcal{G} = (S, \Sigma, S_0, \mathrm{F}, \rho, \ell)$，$\mathcal{G}$ 中除了 F 元素以外，其它元素的定义都与标记 büchi 自动机中的相应元素的定义相同。而 $\mathrm{F} = \{\mathrm{F}_1, \cdots, \mathrm{F}_k\}$ 是一个接受状态集合的集合，其中 $k \geqslant 0$ 且 $\mathrm{F}_i \subseteq S (1 \leqslant i \leqslant k)$。

给定广义标记 büchi 自动机 $\mathcal{G}$ 上的一个运行 $\sigma = s_0, s_1, s_2, \cdots$，令 $\lim(\sigma)$ 是出现在运行 σ 中无限多次的状态的集合。那么称运行 σ 是被自动机 $\mathcal{G}$ 所接受的，当且仅当 $\lim(\sigma) \cap \mathrm{F}_i \neq \emptyset$，也即 σ 无限多次地经过接受集合中的每一个接受子集。令词语 $\omega = a_0 a_1 a_2 \cdots$（其中对于 $i \geqslant 0, a_i \in \ell(s_i)$）是关于该运行 σ 的词语，那么称词语 ω 是被自动机 $\mathcal{G}$ 所接受的词语。当 F 为空集时，所有的运行都是被该自动机所接受的。

基于上述知识，给定一个 *ALC*-LTL 公式 φ，可以相应地构造一个 büchi 自动机。构造过程由 3 个步骤组成：首先从 φ 出发构造一个图，接下来将图转化为广义 büchi 自动机，最后将广义标记 büchi 自动机转化为标记 büchi 自动机。

首先是图的构造，在给出具体构造算法之前首先引入处理过程中用到的几个集合和概念：

1)图 $G = \langle V, E, L \rangle$：其中 V 是生成图的结点集合，结点用 *ALC*-LTL-类的集合标识；E 是边的集合，L 是标记函数，将每个顶点映射到一个 *ALC*-LTL-类的集合或空集。

2)待分解顶点 $V = (P, N, O, Sc)$：其中 $P(v) \in 2^{V \cup \{init\}}$，表示顶点 v 的所有直接前继顶点的集合。在 P 中用符号 $init$ 来指向初始顶点，$init \notin V$ 仅表示一个占位符；$N(v)$ 表示顶点 v 中等待分解处理的公式的集合；$O(v)$ 表示顶点 v 中已经分解处理的公式的集合；$Sc(v)$ 表示顶点 v 所有后继顶点中需要处理的公式集合。

3)序列 S 为一个栈，用来存储算法中待分解处理的顶点。当所有顶点分解处理完成后，也即序列 S 为空时，构造结束。

另外在下面处理过程中，如果 φ 的一个子文字 ψ 是析取形式 $\psi_1 \vee \psi_2$，那么令 $F_1(\psi) = \{\psi_1\}, F_2(\psi) = \{\psi_2\}$；如果 ψ 是算子 $\psi_1 \boldsymbol{U} \psi_2$ 形式，那么令 $F_1(\psi) = \{\psi_1 \wedge \boldsymbol{X}(\psi_1 \boldsymbol{U} \psi_2)\}, F_2(\psi) = \{\psi_2\}$。

给定公式 φ，根据公式的 φ 结构，对其结构进行分解，得到一个带有标记的状态迁移图 $G = \langle V, E, L \rangle$。详细算法过程描述如下：

(1)$S := \{\{\{init\}, \{\varphi\}, \{\}, \{\}\}\}, G := \{V = \{init\}, E = \{\}, L(init) = \emptyset\}$；

(2)从 S 栈首取出一个顶点 v；如果 S 为空栈，则转向(13)；

(3)如果顶点 v 的 $N(v)$ 不为空集，则转向(7)；

(4)如果在图 G 中不存在一个结点 $w \in V$ 使得 $w = O(v)$、$L(w) = Sc(v)$，则转(6)；

(5)生成边 $(P(v)L(v), wO(w))$ 存入集合 E，在 S 栈首删除顶点 v，之后转向(2)；

(6)在 S 栈首加入顶点$\{\{v\},\{Sc(v)\},\{\},\{\}\}$,将 $O(v)$作为图 G 的新结点存入集合 V,$Sc(v)$作为该结点的标记,对每一个顶点 $w\in P(v)$,生成相应的边$(wL(w),O(v)Sc(v))$并存入集合 E,从 S 栈首删除顶点 v,之后转向(2);

(7)从 $N(v)$中取出一个 φ 的子文字 ψ,并将 ψ 从 $N(v)$中删除;如果 ψ 还不能继续分解,即 ψ 不为 φ-文字,则转(9);

(8)如果在集合 $O(v)$中含有 ψ 的否定形式$\neg\psi$,则将顶点 v 从 S 栈首删除,转向(2);否则转向(12);

(9)如果 ψ 是合取形式$(\psi_1\wedge\psi_2)$,那么在 $O(v)$中如果不含 ψ_1 或 ψ_2,则将 ψ_1 或 ψ_2 加入集合 $N(v)$中,转向(3);

(10)如果 ψ 是 $\boldsymbol{X}\psi_1$ 算子,则将 ψ_1 加入集合 $Sc(v)$,转(3);

(11)将顶点 v 分割成两个顶点 w_1、w_2,并将 v 的值赋值给这两个顶点,在 $O(w_1)$中如果不含有 $F_1(\psi)$,则将 $F_1(\psi)$加入集合 $N(w_1)$中;在 $O(w_2)$中如果不含有 $F_2(\psi)$,则将 $F_2(\psi)$加入集合 $N(w_2)$中;在集合 $O(w_1)$和集合 $O(w_2)$中分别加入 φ -文字 ψ;将顶点 w_1 和 w_2 加入 S 栈首,转向(2);

(12)在 $O(v)$中加入 φ 的子文字 ψ,转向(3);

(13)输出图 G。

容易知道,上述构造过程中用到的集合 O 和 Sc 都是 ALC-LTL-类。由算法 1 得到的图,可以生成一个关于 φ 的广义标记 büchi 自动机。

给定由算法 1 生成的图 $G=\langle V,E,L\rangle$,可以生成其相应的广义 büchi 自动机$\mathcal{G}'=(\mathrm{S}',\Sigma',\mathrm{S}_0',\mathrm{F}',\rho',\ell')$,其中,$S':=\{(sl(s))\mid s\in V,l(s)\in L\}$;$\Sigma'$为公式 φ 所有 ALC-类构成的集合;$\mathrm{S}_0':=\{(sl(s))\in S'\mid(init,sl(s))\in E\}$,$\mathrm{F}':=\{\{(sl(s))\in S'\mid\psi_1\boldsymbol{U}\psi_2\notin s$ 或 $\psi_2\in s\}\mid\psi_1\boldsymbol{U}\psi_2$ 是 φ 的子公式$\}$;$\rho'(sl(s)):=\{s'l(s')\in S'\mid(sl(s),s'l(s'))\in E\}$;$\ell'((sl(s)):=\{\mathrm{P}\subseteq\Sigma'\mid(s\cap\Sigma'\subseteq\mathrm{P})\wedge(\mathrm{P}\cap Neg(s)=\emptyset)\}$,其中 $Neg(s)=\{p\in\Sigma'\mid\neg p\in s\}$。

定理 1 对于一个 ALC-LTL 公式 φ 的解释结构$\mathfrak{I}$,总存在一个自动机$\mathcal{G}_\varphi$ 接受的无限字 $\omega\in(\Sigma^\varphi)^\omega$,使得 $\omega=\tau_\varphi(\mathfrak{I})$。

定理 1 表明自动机$\mathcal{G}_\varphi$ 结构的正确性。证明步骤类似于对命题 LTL 自动机结构的证明[9],这里不再累述。

可将上面得到的广义标记 büchi 自动机转换为一个标记 büchi 自动机:

给定一个已知的广义 büchi 自动机$\mathcal{G}'=(S',\Sigma',S_0',\mathrm{F}',\rho',\ell')$,其中,$\mathrm{F}'=\{\mathrm{F}_1,\cdots,\mathrm{F}_k\}$中含有 k 个集合。那么可得到其相应的标记 büchi 自动机$\mathcal{G}=(S,\Sigma,S_0,\mathrm{F},\rho,\ell)$,其中,$S:=S'\times\{i\mid0<i\leqslant k\}$是由状态集合 S'生成的 k 个副本;$\Sigma:=\Sigma'$;$S_0:=S_0'\times\{i\}$,其中 $0<i\leqslant k$,为任一副本中的起始状态集合;$\mathrm{F}:=\mathrm{F}_i\times\{i\}$,其中 $0<i\leqslant k$,为任一副本中的接收集合的任意一个子集;$\rho:(s,i)\to'(s',i)$如果 $s\to s'$且 $s\notin\mathrm{F}_i$,$(s,i)\to'(s',(i+1)\bmod k)$如果 $s\to s'$且 $s\notin\mathrm{F}_i$;$\ell(s,i):=\ell'(s)$。

3.2 时态规范的验证

根据前面一节的内容,已经可以构造出关于时态规范的标记 büchi 自动机$\mathcal{G}_\varphi$。而基于描述逻辑 ALC 的状态迁移系统 $\mathrm{M}=(T,S,s_0,R,Label,F)$则可以直接转化为一个标记 büchi 自动机 $\mathrm{M}_{sys}=(S_{sys},\Sigma_{sys},s_{0sys},\mathrm{F}_{sys},\rho_{sys},\ell_{sys})$,其中 $S_{sys}=S,\Sigma_{sys}$是在 T 上定义的所有可能的 ABox 集合,$s_{0sys}=s_0$,$\mathrm{F}_{sys}=F$,$\ell_{sys}(s)=(T,Label(s))$。由此可以得出基于自动机的 ALC-LTL 模型检测方法:

(1)为系统模型构造一个 büchi 自动机 M_{sys};

(2)为 ALC-LTL 公式$\neg\varphi$ 构造一个标记 büchi 自动机$\mathcal{G}_{\neg\varphi}$;

(3)验证 $\mathrm{L}_\omega(\mathrm{M}_{sys})\bigcap\mathrm{L}_\omega(\mathcal{G}_{\neg\varphi})=\emptyset$ 是否成立。

为了验证 $\mathrm{L}_\omega(\mathrm{M}_{sys})\bigcap\mathrm{L}_\omega(\mathcal{G}_{(\neg\varphi})$是否为空集,可以首先为 M_{sys}和$\mathcal{G}_{\neg\varphi}$构造一个乘积自动机 $\mathrm{M}_{sys}\otimes\mathcal{G}_{\neg\varphi}$,使得该乘积自动机接受的语言为 $\mathrm{L}_\omega(\mathrm{M}_{sys})\bigcap\mathrm{L}_\omega(\mathcal{G}_{\neg\varphi})$。最后验证该乘积自动机的接受语言是否为空集即可。

给定根据时态规范的否定构造的标记 büchi 自动机$\mathcal{G}_{\neg\varphi}=(\mathrm{S}_{\neg\varphi},\Sigma_{\neg\varphi},\mathrm{S}_{0\neg\varphi},\mathrm{F}_{\neg\varphi},\rho_{\neg\varphi},\ell_{\neg\varphi})$,根据系统模型构造的自动机 $\mathrm{M}_{sys}=(S_{sys},\Sigma_{sys},s_{0sys},\mathrm{F}_{sys},\rho_{sys},\ell_{sys})$,那么乘积自动机 $\mathrm{M}_{sys}\otimes\mathcal{G}_{\neg\varphi}=(S,\Sigma,S_0,\mathrm{F},\rho,\ell)$由如下产生式生成:$S:=\{(s,s')\in S_{\neg\varphi}\times S_{sys}\mid\ell_{sys}(s')\models\ell_{\neg\varphi}(s)\}\times\{1,2\}$;$\Sigma:=\Sigma_{\neg\varphi}$;$S_0:=((S_{0\neg\varphi}\times s_{0sys})\times\{1\})\cap S$;$\rho$:如果$(s_1\to s_1')\in\rho_{\neg\varphi}$且$(s_2\to s_2')\in\rho_{sys}$,那么:(1)如果 $s_1\in\mathrm{F}_{\neg\varphi}$,那么$(s_1,s_2,1)\to(s_1',s_2',2)$,(2)如果 $s_2\in\mathrm{F}_{sys}$,那么$(s_1,s_2,2)\to(s_1',s_2',1)$,(3)其他情况,$(s_1,s_2,i)\to(s_1',s_2',i)$,$i=1,2$;$\mathrm{F}:=((\mathrm{F}_{\neg\varphi}\times\mathrm{F}_{sys})\times\{1\})\cap S$;$\ell(s,s',i):=\ell(s)$。

很明显可以看出该乘积自动机接受的语言为 $\mathrm{L}_\omega(\mathrm{M}_{sys}\otimes\mathcal{G}_{\neg\varphi})=\mathrm{L}_\varphi(\mathrm{M}_{sys})\bigcap\mathrm{L}_\omega(\mathcal{G}_{\neg\varphi})$。证明过程不再累述。

给定一个自动机$\mathcal{G}=(S,\Sigma,S_0,\mathrm{F},\rho,\ell)$,如果存在一个初始状态 $s_0\in S_0$ 以及一个接受状态 $s'\in\mathrm{F}$,并且从 s_0 到 s'是可达的,从 s'到 s'也是可达的,那么自动机$\mathcal{G}$的接受语言是非空的。判空算法是一项成熟的技术,在这里不再累述。

通过对乘积自动机 $\mathrm{M}_{sys}\otimes\mathcal{G}_{\neg\varphi}$的判空,即可得出 $\mathrm{M}_{sys}\otimes\mathcal{G}_{\neg\varphi}$中是否存在一条可达的循环路径。如果存在这样的路径,那么这条路径就是一条违反时态规范 φ 的路径,返回该路径。如果不存在这样的路径,那么时态规范 φ 在模型 M_{sys} 中成立,返回 true。

3.3 案例验证

在 2.3 节已经将案例模型建模为一个基于描述逻辑的状态迁移系统 M。为了在 M 中验证规范(1),首先需要将规范(1)的否定形式构造为一个标记 büchi 自动机,先将规范(1)的否定形式规范化:

$$true\boldsymbol{M}(\neg haschild(emperor,nextemperor)\wedge\boldsymbol{X}\neg coupmakers(emperor)) \tag{2}$$

可将规范 φ 的否定形式(2)构造成一个状态迁移图,如图 2 所示。

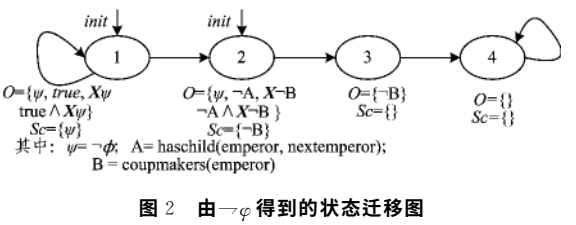

图 2 由$\neg\varphi$得到的状态迁移图

将图 2 转化成相应的广义标记 büchi 自动机$\mathcal{G}'=(S',\Sigma',S_0',\mathrm{F}',\rho',\ell')$,其中:$S'$为 4 个顶点构成的集合,简记为 1,2,3,4;Σ'为 φ 所有的 ALC-类构成的集合;S_0'为顶点 1 和顶点 2 构成的集合;F'为顶点 2、顶点 3 和顶点 4 构成的集合;ρ'为图中箭头标记的转移关系;ℓ'定义如下:

$$\ell'(1)=\{\emptyset,\{coupmakers(emperor)\},\{haschild(emperor,nextemperor)\},\{coupmakers(emperor),haschild(emperor,nextemperor)\}\}$$

$\ell'(2)=\{\emptyset,\{coupmakers(emperor)\}\}$

$\ell'(3)=\{\emptyset,\{haschild(emperor,nextemperor)\}\}$

$\ell'(4)=\{\emptyset,\{coupmakers(emperor)\},\{haschild(emperor,nextemperor)\},\{coupmakers(emperor),haschild(emperor,nextemperor)\}\}$

由于该自动机$\mathcal{G}'$仅有一个接受集合$\{\{2,3,4\}\}$，因此无需转变，$\mathcal{G}'$即是一个标记 büchi 自动机$\mathcal{G}$。

由$\mathcal{G}$与 M，可以得到如图 3 所示的乘积自动机 M $\otimes\mathcal{G}$。

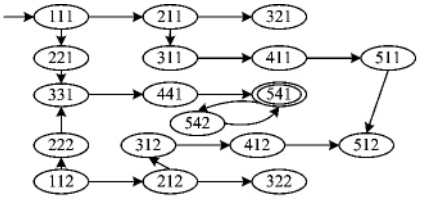

图 3　乘积自动机 M $\otimes\mathcal{G}$

图 3 中每个状态的标号(i,j,k)分别表示 M 的第 i 个状态，$\mathcal{G}$的第 j 个状态，第 k 个副本。在状态 111，511，541，112，512，542 中的 ℓ 值为{haschild(emperor，nextemperor)}，在状态 411，441，412，442 中的 ℓ 值为{haschild(emperor，nextemperor)，coupmakers(emperor)}，其余的状态中 ℓ 值为空集。

由图 3 得到的标记 büchi 自动机容易得出存在一条可接受的途径：111-221-331-441-541-542，它是一条该乘积自动机所接受的路径，所以规范 φ 在模型 M 中不成立。其中一条违背的路径是：{haschild(emperor，nextemperor)}，{∅}，{∅}，{haschild(emperor，nextemperor)，coupmakers(emperor)}，{haschild(emperor，nextemperor)}。

4　工作讨论与比较

线性时态描述逻辑 *ALC*-LTL 最早由 Baader 等提出。与之前文献中的时态描述逻辑相比，*ALC*-LTL 仅仅将时态算子用于对公式的构造而没有用于对概念的构造，从而在很大程度上降低了公式可满足性问题的计算复杂度。Baader 等借助 büchi 自动机证明了 *ALC*-LTL 中公式可满足性问题的可判定性，然后研究了基于 *ALC*-LTL 的运行时验证问题[10]。之后，常亮等[11]给出了时态描述逻辑 *ALC*-LTL 的 Tableau 判定算法并开发了推理工具。与这些工作相比，本文关注的是 *ALC*-LTL 的模型检测问题。

到目前为止，国内外与时态描述逻辑模型检测相关的工作主要有两项：Zhisheng Huang 等[12]将含有过去算子的线性时态逻辑 LTL 与描述逻辑 *ALC* 结合起来，基于模型检测问题对本体演化过程进行推理和验证。但是在这项研究中并没有提出模型检测算法，而只是将研究问题通过 LTL 的模型检测得到了实现。

Ivan Di Pietro 等[13]将分支时态逻辑 CTL 与描述逻辑 *ALC* 结合起来，得到带有描述逻辑 *ALC* 标注的时态逻辑 AnCTL，并提出了 AnCTL 的模型检测算法。算法通过查找和替换将 AnCTL 模型检测问题化简成了 CTL 模型检测问题，最终将 AnCTL 的模型检测算法应用在了对语义 Web 服务组合的验证中。这项研究没能给出直接的模型检测算法，并且对于不成立的规范该算法不能返回一条违反规范的反例。相比于该项工作，本文研究的是 *ALC* 和 LTL 相结合的时态描述逻辑，并且给出了基于自动机的直接的模型检测算法，对于不成立的规范能够返回一条违反规范的路径。

结束语　本文提出的基于自动机的 *ALC*-LTL 的模型检测算法，使用时态描述逻辑 *ALC*-LTL 公式对待验证规范进行描述，在状态迁移系统中引入描述逻辑知识库对系统进行建模。这样增加了对系统领域知识的表示能力，使得时态逻辑的模型检测能够应用到更广阔的空间，尤其适用于在语义 Web 环境下对复杂系统的时态性质进行刻画和验证。

本文算法中采用了闭世界假设，在一定程度上限制了算法的应用范围。下一步将研究如何在采用开世界假设的情况下进行 *ALC*-LTL 模型检测。

参 考 文 献

[1] Clarke E M Jr，Grumberg O，Peled D A. Model Checking [M]. Cambridge，Massachusetts：The MIT Press，1999

[2] van der Hoek W，Wooldridge M. Model checking knowledge and time [C]// 9th Workshop on SPIN(Model Checking Software). April 2002：25-26

[3] Gammie P，van der Meyden R. Mck：Model checking the logic of knowledge [C]// International Conference on Computer Aided Verification(CAV'04). Springer，2004：479-483

[4] 吴立军，苏开乐. 多智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

[5] Baader F，Calvanese D，McGuinness D，et al. The Description Logic Handbook：Theory，Implementation and Applications [M]. Cambridge：Cambridge University Press，2002

[6] Artale A，Franconi E. A survey of temporal extensions of description logics [J]. Annals of Mathematics and Artificial Intelligence，2000，30(1-4)：171-210

[7] Lutz C，Wolter F，Zakharyaschev M. Temporal description logics：a survey[C]// Demri S，Jensen C S，eds. Proceedings of the 15th International Symposium on Temporal Representation and Reasoning. Los Alamitos：IEEE Computer Society Press，2008：3-14

[8] Baader F，Ghilardi S，Lutz C. LTL over description logic axioms [C]// Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning. Cambridge：AAAI Press，2008：684-694

[9] Katoen J-P. Concepts，Algorithms，and Tools for Model Checking [J]. Arbeitsberichte des Instituts fiir Mathematische Maschinen und Datenevarbeitung，1999，31-32：292

[10] Baader F，Bauer A，Lippmann M. Runtime Verification Using a Temporal Description Logic [C]// Proceedings of the 7th International Conference on Frontiers of Combining System. 2009：149-164

[11] 常亮，王娟，古天龙，等. 时态描述逻辑 *ALC*-LTL 的 Tableau 判定算法[J]. 计算机科学，2011，38(8)：150-154

[12] Huang Zhi-sheng，Stuckenschmidt H. Reasoning with Multi-version Ontologies：A Temporal Logic Approach [C]// Galway，International Semantic Web Conference(ISWC2005). 2005：398-412

[13] Pietro I D，Pagliarecci F，Spalazzi L. Model Checking Semantically Annotated Services [J]. IEEE Transactions on Software Engineering(TSE)，2012，38(3)：592-608