文章编号:1005-0523(2023)02-0065-14

时序逻辑及其表达能力综述

杨 科,肖美华,钟小妹

(华东交通大学软件学院,江西 南昌 330013)

摘要:时序逻辑是研究状态随时间变化系统的逻辑特性,在软硬件验证中有着广泛应用,是模型检测的基础。基于对时间模型的不同描述以及为了处理更加复杂的计算特征,衍生出各种时序逻辑,具有不同的表达能力,正确理解其表达能力对于系统模型的形式化规约尤为重要。首先介绍基于离散时间模型的线性时序逻辑LTL、计算树逻辑CTL和CTL*,以及基于连续时间模型的区间时序逻辑ITL和投影时序逻辑PTL,对它们的表达能力及区别进行了详细阐述;然后概述为了描述随机、实时、混成、开放系统中的复杂行为而提出的不同时序逻辑,指出它们的特点及适用范围;最后对时序逻辑的未来研究方向进行展望。

关键词:时序逻辑;表达能力;形式化方法;逻辑系统;形式化规约

中图分类号:TP3-05 **文献标志码**:A

本文引用格式:杨科,肖美华,钟小妹.时序逻辑及其表达能力综述[J]. 华东交通大学学报,2023,40(2):65-78.

Review of Temporal Logic and Its Expressive Power

Yang Ke, Xiao Meihua, Zhong Xiaomei

(School of Software, East China Jiaotong University, Nanchang 330013, China)

Abstract: Temporal logic involves the logic characteristics of the system whose state changes with time. It is widely used in software and hardware verification and is the basis of model checking. Based on the different descriptions of time models and in order to deal with more complex computational characteristics, different kinds of temporal logics are derived, which have different expressive power. It is particularly important for the formal specification of system models to correctly understand their expressive power. Firstly, this paper introduces the discrete-time model Linear Temporal Logic LTL, Computation Tree Logic CTL and CTL*, the continuous time model Interval Temporal Logic ITL and Projection Temporal Logic PTL, and the expressive power and the differences between them are elaborated in details. Then, various temporal logic proposed to describe random, real-time, hybrid and open-system complex behaviors are summarized, and their characteristics and application scopes are presented. Finally, the future research direction of temporal logic is discussed.

Key words: temporal logic; expressive power; formal methods; logical system; formal specification

Citation format: YANG K, XIAO M H, ZHONG X M. Review of temporal logic and its expressive power[J]. Journal of East China Jiaotong University, 2023, 40(2): 65-78.

时序逻辑来源于哲学和语言学,由模态逻辑演变而来,是一种特殊模态逻辑。时序逻辑将时间因素引入到模态逻辑中,将模态算子□(必然)解释为“将来永远”,◇(可能)解释为“将来会”,它是由Prior

收稿日期:2022-04-24

基金项目:国家自然科学基金项目(61562026, 61962020);江西省主要学科学术和技术带头人计划项目(20172BCB22015);江西省2020年度研究生创新专项资金项目(YC2020-B1141)

于1955年首先提出的,是研究时间和时序的逻辑[1]。在过去几十年里,许多逻辑学家和计算机科学家对其进行研究和扩充,衍生出各种不同的时序逻辑,被广泛应用于数学、计算机科学以及人工智能等领域。

早期的形式化方法(20世纪70年代前)致力于研究经典一阶逻辑、Hoare逻辑在串行程序上的描述与验证,该类方法是以逻辑推理为基础的演绎证明。从20世纪70年代中期开始,并发程序设计成为理论计算机科学中的研究热点。不同于串行程序,并发程序涉及到不同任务之间的同步以及信息交换,可能会造成数据竞争、死锁、原子性违反等错误[2],在串行程序分析方法的基础上扩充并发性的推理规则对于并发程序的分析验证并不适用。由于连续运行的系统必然会涉及到时间概念,然而命题逻辑缺乏描述包含连续系统中与时间相关概念的能力。因此以色列科学家伯努利在1977年提出将时序逻辑作为并发系统的规格说明工具,基于时序逻辑对并发程序性质进行形式化规约和验证。20世纪80年代,随着超大规模集成电路、并发分布式系统等技术的迅猛发展, 鉴于演绎验证的局限性,对自动化验证技术提出了更高的要求。美国的Clarke和Emerson, 法国的Sifakis和Queille在1981年分别独立提出将时序逻辑与状态空间搜索相结合的方法,诞生了模型检测技术[3]。随着模型检测技术的发展,时序逻辑被广泛应用在系统形式化规约与验证中, 尤其是上世纪90年代符号模型检测技术的出现,利用二叉决策图的方式来表示模型的状态迁移关系,使得可验证的系统状态数多达10^{200},成功应用于超大规模集成电路和大型软件系统的验证中。时序逻辑作为一种描述与验证计算机系统的形式化规约语言,被广泛应用于各类系统的形式化验证。经过40多年的发展,时序逻辑已经发展成为包括线性时序逻辑、计算树逻辑和区间时序逻辑等众多分支的一门学科。特别是在计算机科学中,时序逻辑作为形式规约语言,已经成为形式化验证程序和并发系统的重要组成部分。

对于一种时序逻辑来说, 一般从可满足性、复杂性、表达性3个方面展开研究[4]。可满足性就是给定一个性质规约公式φ,是否存在一个结构M,该结构恰好是给定公式φ的模型,或者判定模型M是否满足公式φ; 复杂性就是研究逻辑公式可满足性的复杂度问题; 表达性是研究时序逻辑的表达能力,该逻辑公式能够描述哪一种类型的性质。本论文着重研究时序逻辑的表达能力问题,介绍各种时序逻辑和分析它们适合对哪些系统进行性质规约,以及不同时序逻辑在表达能力上的优劣。

1 时序逻辑

时序逻辑主要研究状态随时间变化系统的逻辑特性,由于硬件和软件的运行都是随着时间的变化而不断变化,因此时序逻辑特别适合对该类系统模型进行形式化规约,广泛应用在程序验证和硬件验证领域。在对系统形式化验证时,人们通常关心这两种性质:安全性(safety)和活性(liveness)[5]。它们分别表示“坏的事情永远不会发生”和“好的事情最终会发生”。系统的特性用时序逻辑来表示,比如系统是否会发生死锁、程序代码是否会陷入死循环和系统能否在规定时间内对输入信号做出正确响应等。

下面简要介绍时序逻辑属性、属性的可描述性以及表达能力[6]的形式化定义。

定义1(属性) 在一个无穷状态序列$\delta:\delta_0,\delta_1,\delta_2,\cdots(\delta\in S^\omega)$的集合$S^\omega$上定义属性$p$(property,也称之为性质)。一个属性$p\subseteq S^\omega$是所有满足该属性的无穷状态序列$\delta$的集合,也就是说一个无穷状态序列$\delta$满足属性$p$等价于该序列$\delta$属于$p$所表示的集合,记作$\delta\models p\Leftrightarrow\delta\in p$。

定义2(属性p的可描述性) 设φ是一个时序逻辑公式,如果$\delta\in p$ *iff* $\delta\models\varphi$成立,那么属性p能够用时序逻辑公式φ来描述。

定义3(表达能力) 在一类模型下,语言L_2至少具有和语言L_1同等的表达能力,记作$L_1\subseteq_M L_2$。有如下公式成立:

$$\forall\varphi_1\in L_1,\exists\varphi_2\in L_2,\forall M\in\mathcal{M},M\models\varphi_1 \textit{ iff } M\models\varphi_2$$

如果有$L_1\subseteq_M L_2$和$L_2\subseteq_M L_1$成立,我们称在模型M上语言L_1和L_2表达能力相同,记作$L_1\equiv_M L_2$。如果有$L_1\subseteq_M L_2$成立而$L_1\equiv_M L_2$不成立,称在模型M上语言L_2的表达能力比L_1更强,记作$L_1\subset_M L_2$。

2 基于离散时间模型的时序逻辑

本节介绍基于离散时间模型的时序逻辑,它们的语义被解释在孤立离散的点上,主要包括LTL,CTL和CTL*,并对它们之间的区别进行详细分析比较。

2.1　线性时序逻辑 LTL

LTL 由 Manna 和 Pnueli 提出,是一种非常重要的时序逻辑,可以用来描述并发分布式系统的属性,在模型检测中得到广泛使用。LTL 将时间序列设想成一个线性序列 $\delta:s_0,s_1,s_2,\cdots$,这是一种经典的时间模型。状态之间按照时间参数严格排序,在状态序列上解释其真值,并且其真值随时间变化而变化,这也是时序逻辑与经典逻辑的主要区别之处。

LTL 采用的时间结构是线性、离散的,其语义模型是通过 Kripke 三元组 $M_L=<S,R,L>$来定义的,其中 S 是状态集合,$R\subset S\times S$ 是符合完全性约束的变迁关系(即对于任意的状态 $s\in S$,存在状态 $s'\in S$ 满足 $(s,s')\in R$),$L:S\to 2^{AP}$ 是标记函数,用于标记某个状态上所有满足的原子命题集合。

以领导人选举算法为例,对如何使用 LTL 描述领导人选举算法需要满足的性质进行阐述。领导人选举算法是分布系统中一类非常重要的算法,目的是在多个服务器中选出一个特殊的节点作为领导人,可以简化服务器之间的协作,有助于容错和节省资源。我们用 *elected* 表示选举发生,*num_leader* 表示领导人数量,*one_leader* 表示成功选举出领导人。

性质 1:选举结束后,最终会选出一位领导人:$\Diamond(num\_leader>0)$。

性质 2:选举结束后,至多有一位领导人当选:$\Box\neg(num\_leader>1)$。

性质 3:当某个节点被选为领导人之后,将永远成为领导人:$\bigwedge_i \Box(elected_i\to one\_leader)$。

定理 1　LTL 的表达能力与一阶谓词逻辑等价,不能表达 ω 正则性质[7]。

该定理表明虽然 LTL 与一阶谓词逻辑的表达能力等价,但对于两者的可满足性的复杂度却不相同。对于一阶谓词逻辑来说,它的可满足性问题的解决难度是非初等的,也就是说它的复杂度上界是无限增长的,而对于 LTL 来说是 PSPACE-完全的[8],这也是 LTL 能够得到广泛应用于程序验证、程序综合以及人工智能等领域的一个重要理论支撑[9]。

在显式模型检测中,通常是将软硬件系统用形式化语言建模成模型 M,同时利用 LTL 公式将系统的属性表示为 φ,其核心是判断系统模型的自动机 A_M 与时序逻辑公式 φ 取反得到的等价自动机 A_φ 做交运算后的自动机 A_P 是否为空。可以推断出 LTL 是可以用自动机来表示的,但研究表明 LTL 与自动机之间的转换并不是双向的,LTL 的表达能力弱于自动机[10]。

虽然 LTL 具有容易理解、表达能力较强的优点,但它的缺点是不能表达 ω-正则性质,具备完整的 ω-正则语言表达能力对于性质规约语言来说是十分重要的,尤其是对于无限状态系统的形式化验证。因此一些研究对 LTL 进行扩充,使得扩展得到的时序逻辑表达能力等价于 ω-正则语言[11]。一种方法是将时序逻辑构筑于二阶量词或不动点算子,如 Kozen[12]提出了 μ 演算,μ 演算在时间模型的基础上加入了不动点算子。通过引入不动点算子使得 μ 演算表达性增强,然而付出的代价是其可满足性的判定问题变得复杂,而且不动点算子的嵌套使用令公式难以理解。另外一种方法则是在时序逻辑基础上添加时序算子或者正则表达式,如 Wolper[13]提出了 ETL,通过在 LTL 中加入正则表达式使其表达能力等价于自动机。

2.2　计算树逻辑 CTL

CTL[14]的语义模型与 LTL 类似,都是通过 Kripke 结构来定义的。不同的是,由状态组成的序列 $\Pi:s_0,s_1,s_2,\cdots,s_k,\cdots$ 的路径中,从模型的初始状态作为根节点,对于 $k\geqslant 0$,都有 $(s_k,s_{k+1})\in R$,模型的未来时刻存在着多种不确定状态。因此,把所有路径展开就形成了树状的拓扑结构,这也是被称为计算树的原因,又因为每个节点存在分支,所以 CTL 也被称之为分支时序逻辑(branching temporal logic,BTL)。

CTL 的语义模型是通过 Kripke 三元组 $M=<S,R,L>$来定义的,其中 S 是状态集合,$R\subset S\times S$ 是符合完全性约束的变迁关系(即对于任意的状态 $s\in S$,存在状态 $s'\in S$ 满足 $(s,s')\in R$),$L:S\to 2^{AP}$ 是标记函数,用于标记某个状态上所有满足的原子命题集合。

为了更好地介绍如何使用 CTL 描述相关性质,以多个进程访问共享内存为例进行说明。假设有两个独立进程 $proc_1$ 和 $proc_2$ 访问同一片共享内存,并且要避免它们同时访问,一旦两个进程同时访问临界区,那么就会造成程序崩溃。"*entering*"状态表示当前某个进程请求访问共享内存,"*critical*" 状态表示当前有某个进程正在访问共享内存。

性质 4:两个进程不能同时访问共享内存,即两

个进程间存在互斥性(mutual exclusion)。

$$\forall \square(\neg proc_1.critical \vee \neg proc_2.critical) \quad (1)$$

性质5:如果进程1请求访问共享内存,那么该请求最终会得到响应,即要求满足公平性(fairness)。

$$\forall \square(proc_1.entering \rightarrow \forall \diamond proc_1.critical) \quad (2)$$

性质6:两个进程最终都将能够访问共享内存,即要求满足活性(liveness)。

$$(\forall\square\forall\diamond proc_1.critical) \vee (\forall\square\forall\diamond proc_2.critical) \quad (3)$$

2.3 计算树逻辑 CTL*

针对LTL与CTL在在处理计算分支上的不同,Clarke对CTL的语法和语义作适当的扩充,设计了兼容上述两种时序逻辑算子的语言CTL*[15],该语言融合两种时序逻辑于一种语言之中,同时还克服了时序逻辑公式的二义性解释(时序逻辑公式既可以解释成线性的,也可以解释成分支的)。

CTL*语义的语义与CTL类似,因此不做过多介绍。对于LTL,CTL,CTL*三者之间的关系,可以用图1表示。LTL和CTL存在交集,这部分性质既可以用LTL公式表示,也可以用CTL公式来表示。存在只能用LTL公式表示的性质(如A(FGp))却不存在着相应的CTL公式,也存在只能用CTL公式表示的性质(如AG(EFp))却不存在着相应的LTL公式。这两种性质的结合体A(FGp ∨ AG(EFp)既不能用LTL表示,也不能用CTL表示,只能用这两种时序逻辑的超集CTL*表示。随着CTL*表达能力的增强,CTL*的模型检测的复杂度随之提高,CTL*的模型检测问题是PSPACE-完全的。随着待验证系统规模状态增加,CTL* 模型检测中的状态爆炸问题更加突出,在实际验证中会根据具体情况牺牲部分表达能力而选择复杂度较小的LTL公式或CTL公式。在表达能力上LTL公式可以方便地描述路径范围的选择,但在路径量词方面有所缺失,CTL公式可以精细地描述某个路径,但欠缺对路径范围的选择。

一个逻辑公式的可满足性问题也称为模型检测问题,对于CTL* 有如下结论:

定理2 ① CTL*的模型检测问题是多项式时间完备的。② CTL*的模型检测问题具有NP难度和co-NP难度的[16]。

CTL*是CTL和LTL的超集,表达能力虽然得到了加强,但仍然存在着两个主要缺陷。首先CTL*公式描述的是系统从某一状态s开始后系统的所有可能发生的行为,并不关注系统是以一种什么样的方式到达状态s的,换句话说CTL*公式不能描述状态s之前的系统行为。另外CTL*的缺陷是只能定性地描述系统建立可能发生的行为,但却不能精确地描述该行为发生的确切时间,该类性质对于并发系统中区间相关性质尤为重要[17]。

2.4 LTL和CTL的比较

定义4 LTL公式和CTL公式的等价性。

如果一个LTL公式φ和一个CTL公式ϕ是等价的,记作$\varphi \equiv \phi$,那么对于任意迁移系统(Transition System,简称TS)上的原子命题来说,当且仅当下面的式子成立

$$TS \models \varphi \Leftrightarrow TS \models \phi \quad (4)$$

从公式形式上来看,在LTL公式上添加全称路径量词得到CTL公式,这是因为LTL公式默认包含量词。

然而并不是简单地去掉CTL公式中的路径量词就得到相应的LTL公式,例如CTL公式$\forall \diamond \forall \square a$和LTL公式$\diamond \square a$并不是等价的。LTL公式$\diamond \square a$的含义是从某个状态开始,命题$a$会永远成立,然而CTL公式$\forall \diamond \forall \square a$的语义解释是对于任意一条路径,最终会到达某个状态$s$,使得$s \models \forall \square a$,当且仅当对于任意一条路径$\pi = s_0, s_1, s_2, \cdots \in Path(s)$会有状态$s_j$使得$s_j \models \forall \square a$成立,这也意味着状态$s_j$之后的所有可达状态都满足原子命题$a$。

对于LTL公式$\diamond \square a$来说是满足图2中的状态迁移系统的,初始状态s_0满足公式$\diamond \square a$,从s_0出发最终会停留状态s_0或者s_2,是满足该公式的语义"从某一状态开始a永远为真"。然而对于CTL公式$\forall \diamond \forall \square a$来说不满足图3中的状态迁移系统,该公式在状态$s_0$处不成立,记作$s_0^{\omega} \models \neq \diamond \forall \square a$,这是因为路径$s_0^* s_1 s_2^{\omega}$经过了一个不满足原子命题$a$的状态$s_1$,其中表示由状态$s_0$构成的有穷序列,$s_0^{\omega}$表示由状态$s_2$构成的无穷序列。

另外,许多学者也对LTL和CTL的区别进行过系统研究。这两种逻辑使用同一套符号体系,造成语义上的模糊,关于这两种逻辑孰优孰劣,学者们有着不同的争论。两种时序逻辑的公式是一样的,关键在于语义的区别。LTL以路径作为命题的论断对象,而CTL以状态作为命题的论断对象。图灵奖得主Lamport的结论是[18]:LTL和CTL一般来说是不能相比的,但在证明并发程序的某些性质时LTL要优于CTL,而在证明非确定性程序的某些性质时

CTL 要优于 LTL。Lamport 和 Emerson 等曾讨论过 LTL 和 CTL 的本质区别,及其用于程序验证时功能上的不同。那么它们的不同之处在于语义上的区别,主要体现在下列事实上:

定理 3　在 LTL 中$\Box p \equiv \vdash\!\!\rightarrow p$, 但在 CTL 中$\Diamond p \neg\equiv \vdash\!\!\rightarrow p$。

其中$\vdash\!\!\rightarrow p$ 的含义表示"有时 p 成立"。该定理表明在不同语义模型下,LTL 和 CTL 是不等价的,针对这两种不同时序逻辑的表达能力问题,Lamport 定义了强等价性。

定义 5　给定时序结构 $M=(S,X,D)$,其中 S 为非空状态集,X 是非空路径集,D 是一个映射。如果对于每条路径 $a \in X$ 来说都有$(L,M,a)\models p$ 成立,则称时序公式 p 是在线性语义下 M 为真,记作$(L,M)\models p$。类似的,如果对于每个状态 $t \in S$ 都有$(B,M,t)\models p$ 成立,则称时序公式 p 在分支意义下 M 为真。其中 L 表示线性语义,B 表示分支语义。

定义 6　令 p 和 q 表示两个时序公式,C 是由一组时序结构组成的类,如果对 C 中的每个时序结构,有如下公式成立

$$(I,M)\models p \Leftrightarrow (J,M)\models q \qquad (5)$$

式中:I 和 J 是两个语义解释,可以是线性语义 L 或分支语义 B, 则称时序公式 p 和 q 分别在语义解释 I 和 J 下,相对于结构类 C 是强等价的,记作 $p=q(I,J,C)$。

基于上述定义,可以证明得到如下两个定理:

定理 4　存在这样的结构类 C, 使得 CTL 公式 $\Diamond p$ 不强等价于任何 LTL 公式。

定理 5　存在这样的结构类 C, 使得 LTL 公式$\vdash\!\!\rightarrow \Diamond p$ 不强等价于任何 CTL 公式。

上述定理的证明过程在文献[19]中已经给出,对于上述结论, 可以用下面的例子进行说明。设 P 是一个不确定性程序,则 P 的计算路径和计算结果有多种可能性, 如果要证明不管 P 走哪一条路径,它的计算最终必将终止,那么这个性质用 CTL 公式表示为$\forall \Diamond \text{terminated}(P)$,根据定理 4,该公式是无法用 LTL 公式表示的,因此在这一点上 CTL 是强于 LTL 的。另一方面,设 Q 是一个并发程序,其中包含许多并发执行的分量, 如果要证明 Q 对各分量 Q_i 的调度符合公平性原则,即任何一个分量,只要获取无数多次可执行机会,则一定会被执行,该性质可以用 LTL 表示为$\Box\neg\text{enabled}(Q_i) \vee \text{execute}(Q_i)$,

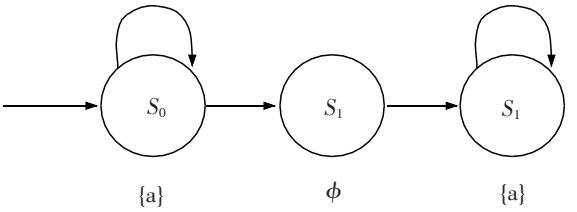

图 1　LTL 公式$\Diamond\Box a$ 的状态迁移系统
Fig.1　The state transition system of LTL $\Diamond\Box a$

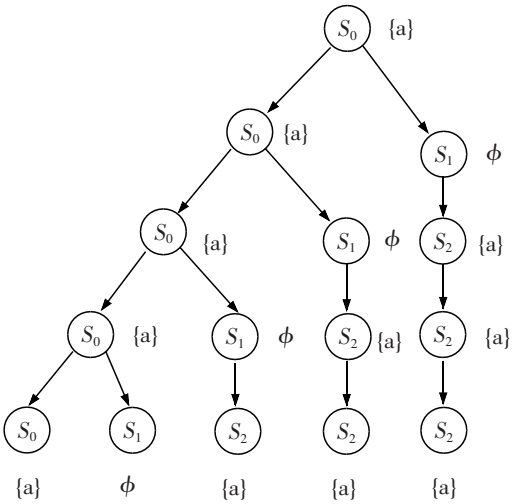

图 2　CTL 公式$\forall \Diamond \forall \Box a$ 的计算树展开形式
Fig.2　The expansion form of computation tree of formula CTL formula $\forall \Diamond \forall \Box a$

根据定理 5,该公式是无法用 CTL 表示的。因此在这一点上 LTL 又强于 CTL。对于该结论的可靠性,Emerson 指出 Lamport 的强等价性的要求过高,该结论是以某个断言在所有状态或所有路径上为真的前提下进行比较的。

对于这两种时序逻辑的比较,从逻辑的角度看这两种语言各有长短。实际应用于模型检测中,普遍认为对于安全属性如公平性和活性的描述,使用 LTL 表示更具有优势。但是从计算的角度来看,实现 CTL 模型检测相对简单,LTL 模型检测问题是 PSPACE-完全的,而 CTL 模型检测时间是线性或多项式的。从模型检测的发展历程来看,在模型检测早期,以 SMV 为代表的 CTL 模型检测器对硬件、数字电路验证占主导地位。但对于刻画系统性质的规约,使用 LTL 更加简洁且容易理解,再加上 LTL 模型检测算法有巨大进步,Cadence SMV 和 SPIN 这类 LTL 模型检测工具得到广泛应用。

对于 LTL,CTL 的模型检测,由于两者的语义解

释不同造成模型检测方法的不同。在对LTL公式φ进行模型检测时,模型检测工具会对φ取反得到有限自动机$A_{\neg\varphi}$,然后与系统模型的有限自动机A_M进行相乘,验证两个自动机的交集是否为空,若不为空,则表示设计中有不满足属性的行为。而对CTL公式进行模型检测是以Kripke结构图的所有状态为根节点进行搜索,如果存在不满足属性的模型,即得到违反属性的反例路径。通过对比这两种不同公式的模型检测方法,可以得出LTL模型检测可以快速完成对路径上的状态搜索,所需要的状态数比CTL模型检测更少。

3 连续时间模型的时序逻辑

本节介绍基于连续时间模型的时序逻辑,包括ITL和PTL,对两种逻辑系统进行分析比较,指出它们的适用范围。

3.1 区间时序逻辑ITL

LTL、CTL公式的语义在独立的点上被解释,一个点代表一个状态,点之间的关系就代表时序之间的关系,这样的时序关系无法描述正则表达式所表达的性质。由于基于连续时间模型的时序逻辑的表达能力较弱,使得对一些性质的验证无法进行。因此很多关于LTL的扩展已经被提出,如Kozen提出了μ演算[12],Vardi提出了ETL[13]。尽管这些时序逻辑具备完全正则语言能力,但仍难以表达一些状态敏感的性质,如原子命题p在某个具体状态或具体区间上成立,此外如顺序、循环等结构难以表达。

与基于点语义的LTL不同,系统的时间特征行为可能会出现在一系列时间区间上,那么就需要使用一些连续时间区间,称之为区间时间模型。Moszkowski在其博士论文中提出了基于区间语义的ITL[20],可用于描述区间中的状态及其时序关系的性质。

定理6 ITL具有正则表达能力,能够方便地表达状态敏感的性质,以及顺序、迭代等行为[20]。

由于ITL定义在有穷区间范围内,主要包括*chop*,*next*以及投影符号*proj*等时序操作符,因此使得ITL比LTL具有更强的表达能力。作为时序逻辑的一个重要分支,现在已经扩展应用在并发软件系统的规约和验证,但ITL的缺陷在于只能应用于有穷区间。

在ITL中有两个最具特色的动作算子:分割算子";"(chop operator)和投影算子*proj*(projection operator)。分割算子";"的作用是把一个有穷区间分割成两个部分,可以表示成p,q,该公式的含义为前一部分性质p成立,后一部分性质q成立。投影算子*proj*具有可重复行为,P *proj* Q把一个有穷区间Q分割成多份具有相同长度的子区间。投影算子的主要作用是能产生重复操作的结构,可用于描述Loops循环,例如动作P每隔时间n就执行一次可描述为P *proj* $len(n)$。

3.2 投影时序逻辑PTL

由于ITL是定义在有穷区间上的时序逻辑,所以该逻辑的缺陷是无法描述系统中的无穷行为。针对ITL存在的问题,西安电子科技大学段振华等人提出了投影时序逻辑PTL[21],对ITL进行扩充到无穷区间范围内,并引入一个全新的投影操作结构$(P_1,\cdots,P_m)prj\ P$,从而得到一种描述有穷、无穷模型和时段的逻辑系统。

通过引入投影操作符号$(P_1,\cdots,P_m)prj\ Q$将ITL推广到无穷区间范围,与原有的投影结构$P\ prj\ Q$相比,新的投影操作更加灵活易用。PTL相比ITL的优势在于:由于扩充了投影算子*prj*,PTL可以应用于无穷区间,允许用不同的时间粒度来描述计算的进程,由此可定义顺序、循环等操作。PTL相比ITL更适合描述并发软件[22-23]。

定理7 PPTL具备完全正则表达能力,能够对顺序、并行、区间相关以及周期重复的性质进行描述[24-25]。

PTL的一个命题子集也就是命题投影时序逻辑PPTL(propositional PTL,PPTL)的表达能力等价于Büchi自动机[26],因而证明其具有完全正则表达能力。PPTL不仅能够表达离散时间模型上的时序性质,还可以表达如区间相关性质以及周期性重复的闭包性质,主要使用区间长度算子$len(n)$,分割算子*chop*,*chop star*和投影算子*prj*来实现,具体性质如下[4]:

(1) 区间相关的性质。例如"命题p在时间长度为n的区间上成立"可以表示为$p\wedge len(n)$;"命题p在第5个和第10个时间单元内成立"可以表示为$len(5);\Diamond p\wedge len(5)$。

(2) 周期重复的性质。例如"命题p在偶数状态上成立"可以表示为$p\wedge(\bigcirc^2(p\wedge\varepsilon))^*$。因此对于验证实时系统的性质不仅局限于传统的安全性和活

性,即描述性质“事件 p 在将来会成立”,还可以通过时间间隔描述对于时间有严格要求的性质,例如性质“事件 p 将在一段时间间隔内发生”。

(3) 顺序、循环、迭代等结构。PPTL 中的分割算子 *chop* 和 *chop star* 可以对该类结构方便地进行描述。

(4) 并发自洽性质。例如 CPU 的任务调度中,每个任务被周期性激活且不受其它任务的影响。令 p_i 代表任务 t_i 的开始,任务 t_i 的时间周期为 Ti,则任务 t_i 周期性地被激活执行可以表示为 $(len(Ti)^{\#})prj \square p_i$,并发自洽的所有任务被激活执行可表示为 $\sum_{i=1}^{n}(len(Ti)^{\#})prj\square p_i$[27]。

总的来说,相比较其它时序逻辑如 LTL、CTL 和 ITL,PTL 的表达能力是完全正则的,等价于 Büchi 自动机,能够表达区间相关的性质以及周期性重复的闭包性质,可描述并发情况以及实时性相关性质。虽然该逻辑的表达能力得到增强,但不可避免地带来更高的模型检测复杂度,减少系统状态空间的方法如偏序规约、限界模型检测以及组合验证等技术被引入到 PPTL 模型检测中[28-29]。

4　处理复杂计算特征的时序逻辑

为了处理复杂的计算特征,如实时、随机、混成、开放系统中的行为,许多时序逻辑被相继提出。例如:为了描述随机系统的高度不确定性,通过在计算树逻辑中引入概率建立了概率计算树逻辑以及连续随机逻辑;为了描述实时系统中关于时间敏感的性质,提出了时段演算等一系列逻辑语言;为了处理混成系统中同时包含离散变量和连续变量的复杂行为,提出了微分动态逻辑;为了处理开放系统中多进程间的任意交互以及系统内部与外界环境的交互,计算树逻辑被扩充为交替时序逻辑,下面将对这些时序逻辑变体及其适合描述哪些性质进行概述。

4.1　面向随机系统的时序逻辑

随机系统具有随机不确定性,为了表达这种性质概率时序逻辑被提出。概率属性规约通常采用概率计算树逻辑 (probabilistic computation tree logic, PCTL)[30]和连续随机逻辑(continuous stochastic logic,CSL)[31]等概率时序逻辑描述。其中 PCTL 是 CTL 的概率扩展,应用在离散时间马尔科夫链和马尔科夫决策,CSL 是在 CTL 和 PCTL 基础上扩展得到,应用在连续时间马尔可夫链,下面对这两种概率时序逻辑作详细介绍与分析。

PCTL 适用于描述随机不确定系统,能够表达定量性质。这种能力在随机不确定性系统中是非常实用的,在形式化验证中有时并不要求系统性质的绝对正确性,而是保证系统性质在一定程度上的相对正确性。例如,在网络通信过程中经常会发生数据传输失败的情况,因此需要对数据进行重新发送。对如何使用 PCTL 表达协议所期望的功能正确性和协议性能为例进行说明。在网络通信过程中,组件 A 和 B 在发送消息时会有一定几率发送失败,如果发送失败则会接着发送直到消息发送成功。

$P_{<0.05}[X\ error\ A]$:组件 A 发送消息失败的概率小于 0.05。

$P_{<0.05}[true\ U^{\leqslant k}\ message\_num \geqslant 5]$:在经过 k 步迁移之后,至少 5 条消息被组件 A 或 B 成功发送的概率小于 0.05。

$P_{>0.9}[F<100\ \text{“enabled”}]$:在经过 100 步迁移之后,消息全部发送成功达到稳定状态的概率大于 0.9。

CSL 也是由 CTL 的概率扩展得到,它是在 CTL 的基础上添加了两个新的算子:路径概率算子 P 和稳定状态算子 S,可以用来描述连续时间马尔科夫链的稳态行为,公式 $S_{\sim p}(\varphi)$表示一个状态满足 φ 的稳态概率$\sim p$。CSL 的语法中引入了一个新的公式 $\phi_1 \cup^{I} \phi_2$,其中 I 是 R 的时间间隔,而 PCTL 中操作符 $\cup$ 被离散时间 k 约束。因此 CSL 与 PCTL 的区别在于 CSL 能够表达某个自然数区间的行为,而 PCTL 是不能的。除了 $S_{\sim p}(\varphi)$以外,在 CSL 中的状态公式与 PCTL 没有区别,它类似于 PCTL 中的 $P_{\sim p}(\varphi)$,表示在满足$\sim p$ 的概率条件下,将会保持在状态 φ 很长时间。CSL 适用于描述在连续时间段之内需要满足的性质,例如“在紧急情况下,飞行控制系统中的阀门必须在 1 秒之内打开的概率大于 99%”,表示为

$$P_{>0.99}[valve\_closed \cup^{[0,1]} valve\_opened] \qquad (6)$$

目前 PCTL 和 CSL 已经在概率模型检测器 PRISM 中得到支持和应用,被用于规约概率系统的期望属性。此外 PRISM 通过扩展回报,一个形如的 $r: S\times S \rightarrow \Re_{\geqslant 0}$ 回报结构将非负数的回报值添加到状

态迁移过程中,可用于分析与回报期望值相关的属性,这是通过扩充 R 算子实现的。以分布式系统中的自稳定算法为例进行说明,在分布式环境下往往要求系统具有自稳定性,这就要求自稳定算法在没有外界干预下,经过有限次步骤后从一个杂乱无章的初始状态到达一个稳定状态。对于自稳定算法关于时间消耗的属性,通过一个简单的回报结构 *time* 将系统中每一步的状态迁移的回报值设定为 1,有如下公式

$R^{time}_{<T}[F\text{“}terminate\text{”}]$:算法的预期结束时间小于规定时间 T 是否成立?

$R^{time}_{=?}[F\text{“}terminate\text{”}]$:算法的预期结束时间是多少?

4.2 面向实时系统的时序逻辑

实时系统是一种能够在有限时间范围内对来自外部输入等做出快速响应的系统, 例如控制系统、监测系统、通信系统等。实时系统的部件之间具有并发性,运行通常并不终止,对规定动作的进行和完成时间也有很高的要求。

在验证安全性至关重要的实时系统中,周巢尘院士、英国计算机科学家 Tony Hoare 和 Anders Ravn 共同提出了时段演算[32](duration calculus),将连续时间概念(积分)引入到计算机科学中,该方法被公认为国际学术界关于实时系统形式设计和验证的一种有效方法。时段演算是 ITL 的实时扩展,以实数作为时间模型,增加的成分主要包括状态表达式 S 和状态时段 $\int S$。状态指的是一个系统内各个分系统的实时状态, 状态表达式的语法是 $S::=0\ |1\ |P|\neg S\ |S\vee S$,状态的值是二元的,每个状态变量在时间点上的取值为 0 或 1,表示为 $I(P):Time\rightarrow\{0,1\}$。状态表达式的时间积分就是状态时段 $\int S$,它是时间区间和状态变量的乘积到一维实数空间的映射,通过状态时段即可刻画实时系统的全局行为[33]。

时段演算适用于对实时系统的实时需求进行刻画[34],不仅能够像其它实时逻辑描述刻度时间性质和时间边界特性,还可以描述一段时间范围内性质的累积。以燃气燃烧器为例,对如何使用时段演算形式化刻画离散状态系统的实时特性进行说明。我们希望控制煤气燃烧器的点火机制和关闭机制,把煤气泄漏控制在安全阈值之内。需求说明为:在煤气灶使用期间如果对其连续观察超过 60 s,那么煤气泄漏的时间不能超过观测时间的 1/20。引入 b 和 e 分别表示观测的起始、截止时间,只有当煤气被释放且没有火焰出现的状态被称为煤气泄漏,用布尔函数 $\mathrm{Leak}(t)$来表示煤气的泄漏状态,可表示 $\mathrm{Leak}(t)\triangleq\mathrm{Gas}(t)\wedge\neg\mathrm{Flame}(t)$为,那么该安全需求形式化表示为

$$(e-b)\geqslant 60\ \mathrm{s}\Rightarrow 20\int\mathrm{Leak}(t)\mathrm{d}t\leqslant(e-b)\tag{7}$$

根据该安全需求说明,给出如下的设计原则:

设计原则 1:任何情况下泄露时间不超过 1 s。

$$\forall c,d{:}b\leqslant c,d\leqslant e,\mathrm{Leak}[c,d]\Rightarrow(d-c)\leqslant 1\ \mathrm{s}\tag{8}$$

设计原则 2:任何两个泄露时段的时间间隔不能少于 30 s。

$$\forall c,d,r,s{:}b\leqslant c<r<s<d<e$$
$$\mathrm{Leak}[c,r]\wedge\neg\mathrm{Leak}[r,s]\wedge\mathrm{Leak}[s,d]\Rightarrow(d-c)\geqslant 30\ \mathrm{s}\tag{9}$$

通过上面的示例可以看出, 时段演算继承了 ITL 的优点,克服了基于点语义的 CTL* 及其子集只能描述某个时刻系统状态的性质,能够描述时段累积性质,那么一个状态的前驱序列的性质也可以通过时段演算公式表示,因此时段演算被广泛应用于实时系统的形式化验证当中。虽然时段演算强大的性质表达能力使其对系统规约带来很大的方便性,但与此同时基于时段演算的自动化验证是十分困难的, 时段演算没有绝对意义上的完备公理系统,时段演算公式的可满足性与模型检测问题都是不可判定的。目前的做法是针对时间模型做一些限制(如对离散时间域的长度进行限制得到有界离散时间模型)或者限制模态算子的使用来得到其可判定的子集和模型检测子集[35]。

此外,针对实时系统的性质规约与验证,还有许多时序逻辑被相继提出,对它们的研究主要分为两个方向。其中一种是通过在时序逻辑中引入含时间界限的时序算子 (如$\Box_{\leqslant t}\varphi$,$\Box_{\leqslant[t_1,t_2]}\varphi$,$\Diamond_{\leqslant t}\varphi$,$\Diamond_{\leqslant[t_1,t_2]}\varphi$)来表示实时性质,包括时间计算树逻辑[36]和度量区间时序逻辑[37]等。另外一种是在时序逻辑中引入显式的时钟变量来表示时间约束,代表性的有实时时序逻辑[38],例如性质“命题 p 在时间区间 $[t_1,t_2]$内成立”可形式化表示为:$\forall u(t=u\wedge\Diamond(p\wedge(u+t_1\leqslant t\wedge t\leqslant u+t_2)))$, 其中 t 是全局时钟变量,u 是刚性变量。上述时序逻辑适用于作为实时与混成系统的性质规约语言,对于实时系统通常是利用时间

自动机或者混成自动机来建模,模型构建和属性规约分别采用两种不同的语言来描述。针对该方法的局限性李广元提出了面向实时系统的连续时序逻辑[39],这是 LTL 在实时领域的一种扩展,该逻辑语言继承了时序逻辑程序设计语言的优点,既能够表示实时系统的实现与性质规约,又能在统一的语义框架下表示高层的需求规约与低层的实现模型之间不同抽象级别的描述。

4.3　面向混成系统的时序逻辑

混成系统是一种特殊的实时系统,混成系统的研究对象包括嵌入式系统和信息物理融合系统等[40]。该类系统的特征是既有离散的逻辑控制又有连续的时间行为,并且这两部分行为相互交织混杂在同一系统中。混成系统的状态变化既随时间连续变化,也受到离散事件驱动[41]。20 世纪 90 年代以来,随着混成系统在工业、航空航天领域得到广泛应用,对该类系统的安全性和可靠性提出了严格要求,保障混成系统的可靠性是目前的研究热点。普通的时段演算主要关注离散状态系统中某类状态在给定观测时间区间(时段)上的积分,而对于混成系统中的连续动态行为缺乏相应的描述机制。Raven 提出了扩展的时段演算 (extended duration calculus,EDC),在原有的时段演算基础上引入连续实值函数对混成系统的连续动态行为进行描述,混成系统的模型以及性质被扩展的时段演算公式形式化表示,然后基于时段演算的公理系统和推理规则对混成系统进行推理验证。

由于混成系统的并发性和时间属性使得混成系统的状态空间极为庞大,而且系统中的状态可达性不可判定使得基于模型检测的混成系统形式化验证难以开展。André Platzerti[42]提出了一类描述混成系统属性的规约语言:微分动态逻辑(differential dynamic logic,DDL)家族,该类逻辑是离散一阶动态逻辑的扩展,DDL 在离散一阶动态逻辑的基础上引入了描述连续状态变化的结构,利用混成程序(hybrid program) 建模混成系统中状态以及状态变迁,具体为使用离散跳跃集表示混成系统中的离散变迁,使用微分方程组描述系统动态行为中的连续变迁,然后通过控制结构操作符(U,*,:)将系统的离散和连续行为组合在一起。

DDL 适用于描述混成系统中离散变迁与连续动态行为特征[43]。这表明 DDL 可用于描述混成系统中连续变化的物理层和离散变化的控制层之间的相互交互过程,并且相关推理规则已经被提出,能对混成系统的行为特性进行推理验证。基于 DDL 对混成系统的形式化验证主要是采用定理证明方法,同时结合了组合验证思想,在验证过程中对待验证属性公式进行逐步分解然后独立验证,能够明显克服基于混成自动机、混成 Petri 网验证混成系统时面临的状态爆炸问题,提高验证效率并能够保持系统模型特征与变迁结构。

DDL 具有良好的扩展性,基于 DDL 的定理证明是在给定的公理和推理规则上进行,为了使 DDL 具有更强大的证明能力,可以对 DDL 的公理系统和推理规则进行扩充。目前在 DDL 基础上根据不同的应用情况以及不同的操作模型衍生出许多变体,包括微分代数动态逻辑 (differential-algebraic dynamic logic,DAL),微分时序动态逻辑(differential temporal dynamic logic,dTL),微分代数时序动态逻辑(differential-algebraic temporal dynamic logic,DATL)以及量化微分动态逻辑(quantified differential dynamic logic,QdL)[44-46]。DDL 缺乏描述物理环境中连续动态行为特性的有效方法,同时存在微分方程不可解的情况,针对该问题在一阶动态逻辑的基础上提出了 DAL,使用微分不变量技术进行推理验证而不必求解微分方程。为了验证混成系统中的时序属性,将 DDL,DAL 分别与时序逻辑相结合得到 dTL 和 DATL,DATL 将动态逻辑的模态操作符和时序逻辑的时序操作符完美结合在一起,得到形如的逻辑公式,可以同时描述系统的时序行为和非时序行为。为了处理分布式混成系统的形式化验证,基于一阶动态逻辑和量化的微分方程相结合得到 QdL。DDL 家族之间的关系可以用图 3 来说明。

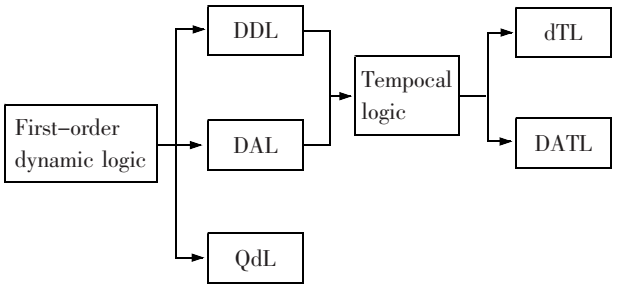

图 3　微分动态逻辑家族示意图

Fig.3　Schematic diagram of differential dynamic logic family

4.4 面向开放系统的时序逻辑

封闭系统的行为由当前系统所处的状态决定，与外部环境无关，针对封闭系统的模型检测主要采用LTL或CTL来描述系统的性质。开放系统的行为不仅由系统当前所处状态决定，同时也受到外部环境的约束[47]。与封闭系统相比，开放系统的外部环境是不确定的，开放系统中的不同组件之间的交互会导致难以预测的情况发生，如可观测性以及可控性等。因此，开放系统的形式化验证需要将外部环境的各种不确定情况纳入考虑范围内[48]。Alur提出了交替时序逻辑(alternating-time temporal logic，ATL)[49]，该逻辑描述的是开放系统与外界环境之间的交互以及内部组件之间的合作与竞争关系，将系统中的各组件之间的交互最终到达的稳定状态规约为开放系统与外界环境之间的博弈结果，所以又被称为博弈逻辑。

ATL适用于描述开放系统与外界环境之间的交互以及内部组件之间的合作与竞争关系[50]。这表明对于系统行为同时受到内部结构和外部环境输入的开放系统，系统不同组件之间交互后最终达到的状态，可以使用ATL规约为开放系统与外部环境之间的博弈。以几个简单的例子介绍如何使用ATL来描述开放系统中各个主体之间的合谋与对抗行为，来说明ATL对于开放系统的表达能力。假设存在参与者集合$\sum=\{a,b,c\}$，不同的参与者存在着策略使得命题p成立，同时不同参与者之间存在着合谋与竞争，有如下ATL公式：

$\langle\langle a\rangle\rangle\Diamond p$：针对参与者$b$和$c$，参与者$a$存在一个获胜策略使得命题$p$最终成立。

$\langle\langle b,c\rangle\rangle\Box p$：针对参与者$a$，无论参与者$b$和$c$如何合谋，永远不能避免系统最终达到一个状态使得命题p成立。

$\langle\langle a,b\rangle\rangle\bigcirc(p\wedge\neg\langle\langle c\rangle\rangle\Box p)$：针对参与者$c$，参与者$a$和$b$合谋能够使得下一状态命题$p$成立，并且从下一状态开始参与者$c$永远不能使命题$p$成立。

上述公式形式化地描述了一个开放系统中不同成员之间的合作与对抗行为，这种表达能力对于描述电子商务协议这种开放系统是非常合适且有效的。与传统的安全协议中协议参与方是诚实的相比，电子商务协议中除了可信第三方(trusted third party，TTP)是诚实可信的，其他参与主体都是互不信任且会做出欺骗行为。例如电子支付协议中存在消费者已经收到货物但是却拒不承认的情况，或者消费者与攻击者合谋一起欺骗商家等恶意行为，电子商务协议中的安全威胁不仅仅来自外部环境，更多的是协议参与方之间的合谋与对抗。在基于Dolev-Yao模型的协议分析过程中，安全协议被当作一个封闭系统进行研究，不能有效地描述协议与外部环境之间的交互。而基于ATL的电子商务协议安全性分析中，充分考虑了协议主体间的相互博弈，所有协议参与方根据自身利益最大化选择适合自己的策略，根据博弈结果决定自己下一步动作，能够对协议主体间的合谋与对抗行为进行精确地描述。目前实验结果表明ATL比LTL，CTL更适合分析电子商务协议这一类开放系统，已经成功发现合同签署协议、公平交换协议中存在不满足公平性等缺陷[51-52]。

此后有许多研究工作对ATL进行扩展形成了ATL家族，主要体现在策略表示、与其它逻辑相融合等方面的扩展。ATL规定每一个路径选择量词《A》必须有一个时序算子相匹配，针对该缺陷提出的ATL*支持这两种算子之间的任意嵌套，因此ATL*的性质表达能力比ATL更强。ATL中只能对策略进行量化，模糊地表示存在某一策略使得性质φ成立，但不能明确指出是哪一个策略使得性质φ成立，针对该问题具体策略交替时序逻辑(ATL with explicit strategies，ATLES)[53]被提出。为了将行动的实施与其所掌握的知识前提(knowledge precondition)形式化表示，文献[54]将ATL与认知逻辑(epistemic logic)相结合提出了交替认知时序逻辑(alternating-time temporal epistemic logic，ATEL)。为了解决开放系统的验证过程中没有考虑随机行为的问题，通过引入概率分布到认知模型中来描述系统中的随机性，文献[55]基于ATEL提出了概率交替时序认知逻辑(probabilistic alternating-time temporal epistemic logic，PATEL)，能够对开放系统中的知识前提及其它性质进行定量验证。文献[56]将ATL与投影时序逻辑PTL结合提出了交替投影时序逻辑(alternating projection temporal logic，APTL)，该逻辑融合了这两种逻辑的优点，不仅可以描述有穷区间、无穷区间内的性质，还可以描述开放系统中与博弈相关的性质。上述ATL家族之间的关系如图4所示。

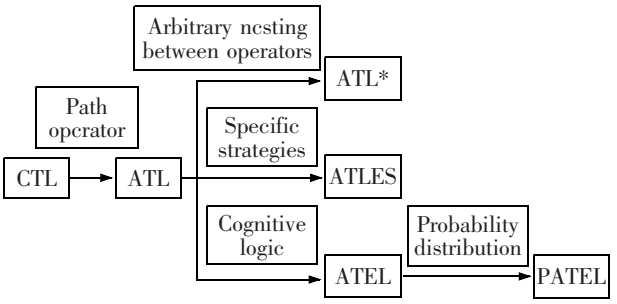

图 4　交替时序逻辑家族示意图

Fig.4　Schematic diagram of alternating-time temporal logic family

5　时序逻辑未来的研究方向

从时序逻辑的发展趋势来看,今后一段时间时序逻辑的研究工作包括以下几个方面:

1) 扩充已有的时序逻辑,使其能够适用于复杂系统的形式化规约。例如,随着大规模集成电路和 EDA 技术的发展,片上系统(system on chip)成为嵌入式系统的发展方向, 减少片上系统中的设计错误,提高系统的设计可靠性是关键性问题,基于仿真的方法中存在着覆盖率不足、仿真时间过长等问题。信号时序逻辑[57]是在模拟和混合信号电路背景下提出的规约语言,以 LTL 为基础同时具有实时和实值约束, 可以描述离散和连续系统的实时属性。在量子计算方面,随着量子计算机在计算速度方面有超越经典计算机的绝对优势,量子程序的开发及验证是发挥量子计算机能力的关键因素,目前基于 CTL 的量子计算树逻辑 (quantum computation tree logic,QCTL)[58-59] 已被提出用于量子程序的性质规约。此外,新的计算模式如大数据、机器学习算法不断涌现,这些计算模式下程序行为与经典的串行与并发程序有显著差异,因此需要扩充或设计新的时序逻辑来描述这些新的计算模式[60-61]。

2) 研究不同形式化逻辑系统之间的组合,使得组合后的逻辑能够吸收不同逻辑的特性。例如分离逻辑是对霍尔逻辑的扩展,对于验证具有复杂数据结构的动态内存程序非常实用, 但从表达能力来看,分离逻辑并未突破一阶逻辑的范畴,在描述和验证内存的时序性质方面仍有欠缺[62]。时序逻辑虽然能够描述状态随时间变化系统的时序性质,但不能够描述带有指针的程序性质,文献[63-64]提出一种结合分离逻辑和 PPTL 的二维(时间和空间)时序逻辑 PPTLSL,该逻辑继承了分离逻辑和 PPTL 的优势,能够描述指针程序的时序性质。此外,利用 until 算子描述的性质在 LTL 中可以轻松实现,但却不能被 PPTL 有效支持。文献[65]将 LTL 中的 until 算子在有穷区间和无穷区间上重新定义并扩充到 PPTL 中,提出一种新的统一时序逻辑 UTL(unified temporal logic,UTL),该逻辑完美地结合了 LTL 和 PPTL 的特性。

3) 时序逻辑的判定问题、模型检测问题和公理化问题。对于时序逻辑而言,除了需要研究它的表达能力之外,还需要研究该逻辑的判定性、模型检测和公理系统。时序逻辑的判定问题即该逻辑的(可满足)判定性及复杂度,该问题在一种新的时序逻辑被提出时是必须考虑的。基于时序逻辑的模型检测即自动化验证系统模型是否满足性质规约的重要手段,设计高效的模型检测算法与开发相关支撑工具是人们追求的目标。公理化问题即基于该逻辑提出相应的公理和推理规则,并保证该公理系统的可靠性和完备性。针对特殊的无穷状态系统的模型检测,结合定理证明技术能够有效缓解系统状态空间的膨胀。

6　结束语

形式化方法是验证安全攸关系统安全性与可靠性的重要手段, 时序逻辑作为形式化规约语言,能够描述软硬件系统中基于时间变化的状态迁移,是形式化验证的基础。本文首先介绍基于离散时间模型的 LTL,CTL 和 CTL*, 以及基于连续时间模型的 ITL 和 PTL,对它们之间的区别进行阐述;然后介绍为了处理复杂计算特征(如随机、实时、混成和开放系统中的行为)而提出的各种时序逻辑;最后指出时序逻辑未来的研究方向。

参考文献:

[1] 张广泉,孙敏. 时态逻辑的比较与分析[J]. 渝州大学学报(自然科学版),1999,16(2):18-21.
ZHANG G Q,SUN M. Comparison and analysis of temporal logic[J]. Journal of Yuzhou University(Natural Science Edition),1999,16(2):18-21.

[2] 蒋炎岩,许畅,马晓星,等. 获取访存依赖:并发程序动态分析基础技术综述[J]. 软件学报,2017,28(4):747-763.
JIANG Y Y,XU C,MA X X,et al. Approaches to obtaining shared memory dependences for dynamic analysis of concurrent programs:A survey[J]. Journal of Software,2017,28

(4):747-763.

[3] CLARKE E M,EMERSON E A,SISTLA A P. Automatic verification of finite-state concurrent systems using temporal logic specifications[J]. ACM Transactions on Programming Languages and Systems,1986,8(2):244-263.

[4] 田聪. 命题投影时序逻辑的判定性、复杂性、表达性及模型检测[D]. 西安:西安电子科技大学,2010.
TIAN C. Decidability,complexity,expressiveness and model checking of propositional projection temporal logic[D]. Xi′an:Xidian University,2010.

[5] 陈钢,于林宇,裘宗燕,等. 基于逻辑的形式化验证方法:进展及应用[J]. 北京大学学报(自然科学版),2016,52(2):363-373.
CHEN G,YU L Y,QIU Z Y,et al. Logic based formal verification methods:progress and applications[J]. Acta Scientiarum Naturalium Universitatis Pekinensis,2016,52(2):363-373.

[6] CHRISTEL B,JOOST P K. Principles of Model Checking[M]. Cambridge,Massachusetts:The MIT Press,2008.

[7] WOLPHER P. Temporal can be more expressive[J]. Information and Control,1983,56(1-2):72-99.

[8] HALPERN J Y,REIF J H. The proposition dynamic logic of deterministic,wellstructured programs[J]. Theoretical Computer Science,1983,27:127-165.

[9] 朱淑芳. 有限线性时序逻辑程序综合的理论与算法研究[D]. 上海:华东师范大学,2019.
ZHU S F. Program synthesis of linear temporal logic over finite traces[D]. Shanghai:East China Normal University,2019.

[10] 李建文. 线性时态逻辑中若干基础问题的研究[D]. 上海:华东师范大学,2014.
LI J W. Research on some fundamental problems of linear temporal logic[D]. Shanghai:East China Normal University,2014.

[11] 刘万伟. 扩展时序逻辑的推理及符号化模型检验技术[D]. 长沙:国防科学技术大学,2009.
LIU W W. Reasoning and symbolic model checking of extended temporal logics[D]. Changsha:National University of Defense Technology,2014.

[12] KOZEN D. Results on the propositional mu-calculus[J]. Theoretical Computer Science,1983,27(3):333-354.

[13] VARDI M Y,WOLPER P. Reasoning about infinite Computations[J]. Information and Computation,1994,115(1):1-37.

[14] BEN M,PNUELI A,MANAN Z. The temporal logic of branching time[J]. Acta Informatica,1983,9(20):207-226.

[15] EMERSON E A,HALPERN J Y. "Sometimes" and "not never" revisited:On branching versus linear time temporal logic[J]. Journal of the ACM,1986,33(1):151-178.

[16] VARDI M Y,STOCKMEYER L. Improved upper and lower bounds for modal logics of programs[R]. Technical report,in:STOC′85:Proceedings of the seventeenth annual ACM symposium on Theory of computing,Lecture Notes in Computer Science,1985:240-251.

[17] 梁爱丽,朱嘉奇,王捍贫,等. 一种新的时段演算及其验证[J]. 计算机研究与发展,2008(S1):169-174.
LIANG A L,ZHU J Q,WANG H P,et al. A new duration calculus and its verification[J]. Journal of Computer Research and Development,2008(S1):169-174.

[18] LAMPORT L. What good is temporal logic [J]. Infermation Processing,2016,83:657-668.

[19] 陆汝钤. 计算系统的形式语义[M]. 北京:清华大学出版社,2015.
LU R Q. Formal semantics of computing systems[M]. Beijing:Tsinghua University Press,2015.

[20] MOSZKOWSKI B. Reasoning about digital circuits [D]. California:Stanford University,1983.

[21] DUAN Z H. An extended interval temporal logic and a framing technical for temporal logic programming[D]. Newcastle:University of Newcastle,1996.

[22] 朱维军. 时间区间时序逻辑模型检测:理论、算法及应用[D]. 西安:西安电子科技大学,2011.
ZHU W J. Model checking time interval temporal logic:theory algorithm and application[D]. Xi′an:Xidian University,2011.

[23] 舒新峰,段振华. 投影时序逻辑在系统建模中的应用[J]. 西北大学学报(自然科学版),2010,40(3):410-414.
SHU X F,DUAN Z H. Application of projection temporal logic in system modeling[J]. Journal of Northwest University(Natural Science Edition),2010,40(3):410- 414.

[24] DUAN Z H,TIAN C,ZHANG L. A decision procedure for propositional projection temporal logic with infinite models [J]. Acta Informatica,2008,45(1):43-78

[25] 于斌. MSVL 程序的高效运行时验证方法研究[D]. 西安:西安电子科技大学,2019.
YU B. Research on efficient runtime verification for MSVL programs[D]. Xi′an:Xidian University,2019.

[26] TIAN C,DUAN Z H. Expressiveness of propositional projection temporal logic with star[J]. Theoretical Computer Science,2011,412(18):1729-1744.

[27] 张琛. 基于 UML2.0 模型的测试与验证方法[D]. 西安:西安电子科技大学,2012.
ZHANG C. Testing and verification methods based on UML2.0 models[D]. Xi′an:Xidian University,2012.

[28] 逄涛,段振华,刘晓芳. 一个命题投影时序逻辑符号模型检测器[J]. 软件学报,2015,26(8):1968-1982.
PANG T,DUAN Z H,LIU X F. Symbolic model checker for propositional projection temporal logic[J]. Journal of Software,2015,26(8):1968-1982.

[29] 舒新峰,王昌太,王燕,等. 一种命题投影时序逻辑的分布式模型检测方法[J]. 西安电子科技大学学报,2020,47(4):39-47.

SHU X F，WANG C T，WANG Y，et al. Propositional projection temporal logic based distributed model checking method[J]. Journal of Xidian University，2020，47(4)：39–47.

[30] HASSON H，JONSSON B. A logic for reasoning about time and reliability[J]. Formal Aspect and Computing，1994，6(5)：102–111.

[31] AZIZ A，SANWAL K，SINGHAL V，et al. Model–checking continuous–time Markov chains[J]. ACM Transactions on Computational Logic，2000，1(1)：162–170.

[32] ZHOU C C，HOARE C，ANDERS P R. A calculus of duration[J]. Information Processing Letters，1991，40(5)：269–276.

[33] 李晓山，周巢尘.时段演算综述[J].计算机学报，1994(11)：842–851.
LI X S，ZHOU C C. Duration calculi：an overview[J]. Chinese Journal of Computers，1994(11)：842–851.

[34] MICHAEL R，ZHOU C C. Duration calculus：logical foundations[J]. Formal Aspect of Computing，1997，9(3)：283–330.

[35] 安杰，张苗苗. 基于实时自动机的连续时段演算的验证[J]. 软件学报，2019，30(7)：1953–1965.
AN J，ZHANG M M. Verifying continuous–time duration calculus against real–time automaton[J]. Journal of Software，2019，30(7)：1953–1965.

[36] ALUR R，COURCOUBETIS G，DILL D. Model–checking in dense real–time[J]. Information and Computation，1993，104：2–34.

[37] ALUR R，FEDER T，HENZINGER T A. The benefits of relaxing punctuality[J]. Journal of the ACM，1996，43(1)：116–146.

[38] FERRERE T，MALER O，NICKOVIC D，et al. From real–time logic to timed automata[J]. Journal of the ACM，2019，66(3)：1–31.

[39] 李广元. LTLC：面向实时与混成系统的连续时序逻辑[D]. 北京：中国科学院软件研究所，2001.
LI G Y. LTLC：a continuous–time temporal logic for real–time and hybrid systems[D]. Beijing：Institute of Software Chinese Academy of Sciences，2001.

[40] 解定宝. 混成系统有界模型检验优化技术研究[D]. 南京：南京大学，2016.
XIE D B. Research on optimization techniques for bounded model checking of hybrid systems[D]. Nanjing：Nanjing University，2016.

[41] 卜磊，解定宝. 混成系统形式化验证[J]. 软件学报，2014，25(2)：219–233.
BU L，XIE D B. Formal verification of hybrid system[J]. Journal of Software，2014，25(2)：219–233.

[42] PLATZER A. Differential dynamic logic for hybrid systems[J]. Journal of Automated Reasoning，2008，41(2)：143–189.

[43] PLATZER A. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems[J]. Logical Methods in Computer Science，2012，8(4)：205– 214.

[44] 陈乔乔，李必信，吉顺慧. 一种基于微分代数动态逻辑的 CPS 建模与验证方法[J]. 计算机研究与发展，2013，50(4)：700–710.
CHEN Q Q，LI B X，JI S H. A modeling and verification method of CSP based on differential–algebraic dynamic logic[J]. Journal of Computer Research and Development，2013，50(4)：700–710.

[45] PLATZER A. A complete uniform substitution calculus for differential dynamic logic[J]. Journal of Automated Reasoning，2017，59(2)：219–265.

[46] PLATZER A. Differential–algebraic dynamic logic for differential–algebraic programs[J]. Logic and Computation，2010，20(1)：309–352.

[47] 张业迪，宋富. 异构多智能体系统模型检查[J]. 软件学报，2018，29(6)：1582–1594.
ZHANG Y D，SONG F. Model–checking for heterogeneous multi–agent systems[J]. Journal of Software，2018，29(6)：1582–1594.

[48] 王海洋. APTL 模型检测方法及多智能体系统验证的研究[D]. 西安：西安电子科技大学，2018.
WANG H Y. Study on the method of APTL model checking and the verification multi–agent systems[D]. Xi′an：Xidian University，2018.

[49] ALUR R，THOMAS V，KUPFERMAN O. Alternating–time temporal logic[J]. Journal of the ACM，2002，5(49)：672–713.

[50] HOEK W，WOOLDRIDGE M. Cooperation，knowledge，and time：alternating–time temporal epistemic logic and its applications[J]. Studia Logica，2003，75(1)：125–157.

[51] 文静华，李祥，张焕国，等. 基于 ATL 的公平电子商务协议形式化分析[J]. 电子与信息学报，2007(4)：901–905.
WEN J H，LI X，ZHANG H G，et al. Formal analysis of fair e–commerce protocols based on ATL[J]. Journal of Electronics & Information Technology，2007(4)：901–905.

[52] 李群，陈清亮. 基于 ATL 的公平交换协议的形式化验证[J]. 计算机工程与应用，2015，51(19)：32–36.
LI Q，CHEN Q L. Formal verification of fair exchange protocols based on alternating–time temporal logic[J]. Computer Engineering and Applications，2015，51(19)：32–36.

[53] LAROUSSINIEA F，MARKEY N. Augmenting with strategy contexts[J]. Information and Computation，245(1)：98–123.

[54] HOEK W，WOOLDRIDGE M. Cooperation，knowledge and time：alternating–time temporal logic and its application[J]. Studia Logica，2003，75(1)：125–157.

[55] 徐伟峰. 多主体模型定量验证方法研究[D]. 长春：吉林大学，2014.
XU W F. Quantitative verification of multi–agent model[J]. Changchun：Jilin University，2014.

[56] 王海洋，段振华，田聪. 用于验证多智能体系统的 APTL 模型检测器[J]. 软件学报，2019，30(2)：231–243.
WANG H Y，DUAN Z H，TIAN C. APTL model checker

for verifying multi-agent systems[J]. Journal of Software, 2019,30(2):231-243.

[57] JYOTIRMOY V D,ALEXANDRE D,SHROMONA G. Robust online monitoring of signal temporal logic[J]. 2017,51(1):5-30.

[58] FENG Y,YU N K,YING M S. Model checking quantum Markov chain[J]. Journal of Computer and System Sciences, 2013,79(7):1181-1198.

[59] 冯元,应明生. 量子程序验证[J]. 软件学报,2018,29(4):1085-1093.
FENG Y. YING M S. Verification of quantum programs[J]. Journal of Software,2018,29(4):1085-1093.

[60] CCF 形式化专业委员会. 形式化方法的研究进展与趋势[R]. 2017—2018 年中国计算机科学技术发展报告,北京:机械工业出版社,2018:1-68.
CCF Formal methods Technical Committee. Advance and trends on formal methods[R]. In:The Progress Report of Computer Science and Technology in China from 2017 to 2018. Beijing:China Machine Press,2018:1-68.

[61] 肖美华,周浩洋,朱志亮,等. 基于模型检测的区块链智能合约公平性形式化验证[J]. 华东交通大学学报,2021,38(3):52-60.
XIAO M H,ZHOU H Y,ZHU Z L,et.al. Formal verification of fairness of blockchain smart contract based on model checking[J]. Journal of East China Jiaotong University,2021,38(3):52-60.

[62] 秦胜潮,许智武,明仲. 基于分离逻辑的程序验证研究综述[J]. 软件学报,2017,28(8):2010- 2025.
QIN S C,XU Z W,MING Z. Survey of research on program verification via separation logic[J]. Journal of Software, 2017,28(8):2010-2025.

[63] 陆旭,段振华,田聪. 二维逻辑 PPTLSL 的可满足性检查[J]. 软件学报,2016,27(3):670-681.
LU X,DUAN Z H,TIAN C. Checking satisfiability of two-dimensional logic PPTLSL[J]. Journal of Software,2016,27(3):670-681.

[64] 陆旭. 时空逻辑 PPTLSL 及其应用研究[D]. 西安:西安电子科技大学,2017.
LU X. Research on Spatio-Temporal logic PPTLSL and its applications[D]. Xi′an:Xidian University,2017.

[65] ZHANG N,DUAN Z H,TIAN C. Unified temporal logic[J]. Theoretical Computer Science,2021,864(5):123-129.

第一作者:杨科(1993—),男,博士研究生,研究方向为形式化方法, 安全协议的形式化分析。Email:landexplorer@163.com。

通讯作者:肖美华(1967—),男,博导,教授,江西省主要学科学术与技术带头人,研究方向为形式化方法,安全协议形式化分析。Email:xiaomh@ecjtu.edu.cn。

(责任编辑:李 根)