软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(3):421−432 [doi: 10.3724/SP.J.1001.2013.04162]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向事件图和事件时态逻辑的模型检验方法*

夏 薇[1,2], 姚益平[1], 慕晓冬[2]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](第二炮兵工程大学 计算机系,陕西 西安 710025)

通讯作者: 夏薇, E-mail:weiwei32329@163.com

摘 要: 针对目前没有适合直接对事件图模型进行性质规约的时态逻辑语言,提出一种基于事件的时态逻辑(event temporal logic,简称 ETL).ETL 以事件作为原子命题,根据事件图的特点增加了对事件取消操作、模型实例化、时间约束和同时事件优先级的表达能力,便于仿真领域的用户在模型检验过程中简洁地对基于事件图的模型应满足的性质进行描述.然后,在 ETL 公式和自动机理论的基础上,给出了面向事件图和 ETL 的模型检验方法来判断事件图模型是否满足ETL描述的性质规约.实例验证了 ETL 对事件图模型具有足够的表达能力以及该方法的有效性.

关键词: 事件图;事件时态逻辑;模型检验;Büchi 自动机;转换

中图法分类号: TP301 **文献标识码**: A

中文引用格式: 夏薇,姚益平,慕晓冬.面向事件图和事件时态逻辑的模型检验方法.软件学报,2013,24(3):421−432. http://www.jos.org.cn/1000-9825/4162.htm

英文引用格式: Xia W, Yao YP, Mu XD. Model checking for event graphs and event temporal logic. Ruanjian Xuebao/Journal of Software, 2013,24(3):421−432 (in Chinese). http://www.jos.org.cn/1000-9825/4162.htm

Model Checking for Event Graphs and Event Temporal Logic

XIA Wei[1,2], YAO Yi-Ping[1], MU Xiao-Dong[2]

[1](College of Computer, National University of Defense Technology, Changsha 410073, China)

[2](Department of Computer Science, Xi'an Hi-Tech Institute, Xi'an 710025, China)

Corresponding author: XIA Wei, E-mail: weiwei32329@163.com

Abstract: This paper proposes an event temporal logic (ETL) because there has been no suitable temporal logic that can directly describe the properties of event graph (EG) models. ETL takes events as its atomic propositions and has the abilities to describe the event canceling edge, the passing parameter values between events, time constraints and the priorities of coinstantaneous events, which can facilitate the description of properties for EGs. A model checking method for EG and ETL on the basis of theory of automata is also proposed in this paper to check whether properties hold for models, according to the accepted language is an empty set or not. The experimental results show ETL is powerful enough to describe EG models, and the model checking method for EG and ETL is effective.

Key words: event graph; event temporal logic; model checking; Büchi automaton; transformation

模型检验是由 Clarke 等人提出的一种被广泛应用的形式化验证方法,其基本思想是通过遍历系统模型的状态空间来检验系统模型是否满足给定的性质[1].模型检验方法涉及到两种形式化语言:一种是模型描述语言,用于描述系统行为,通常为自动机、Petri 网或进程代数等数学模型;另一种是性质描述语言,用于描述系统的性质,通常为各种时态逻辑,这类逻辑主要关心事件发生的顺序及时刻.简单说,模型检验方法就是用于检验由模型描述语言描述的系统模型是否满足由性质说明语言描述的系统性质[2].

采用形式化方法对仿真模型进行形式化验证和分析是构造可靠、安全仿真系统的一个重要途径[2],然而模

* 基金项目: 国家自然科学基金(61170048); 国家教育部博士点基金(200899980004)
收稿时间: 2011-06-23; 修改时间: 2011-10-08; 定稿时间: 2011-11-17

型检验方法目前并没有在离散事件仿真领域得到广泛的应用,主要原因可以归纳为以下两个方面:(1) 从模型描述语言来看,目前的离散事件仿真建模语言大都是基于高级编程语言(如 C,C++,Java)或离散事件系统规范(discrete event system specification,简称 DEVS)实现的,而模型检验方法的输入模型通常是时间自动机、Petri 网以及进程代数等数学模型,对基于高级编程语言实现的仿真模型使用模型检验方法较为复杂,需要形式化定义程序语言的语义,处理复杂的数据结构和动态特性[1];(2) 由于时态逻辑语言结构较为复杂,用自然语言很容一描述的时态属性需求,用时态逻辑公式却很难准确描述.在工业应用中,用手工写时态逻辑公式是不现实的,故在自然语言和时态逻辑公式之间存在着一个难以逾越的鸿沟[3].现阶段,要采用模型检验方法对离散事件仿真模型进行形式化验证,必须根据所选定的模型检验工具的模型描述语言要求构建该工具所识别的模型;同时,还要根据性质描述语言的要求归纳出仿真模型应该具备的性质.然而,仿真参与者大都是面向各个领域的专家,对他们而言,由于缺乏一定的形式化规约基础,使用基于时态逻辑的模型检验方法不仅费时而且容易出错.模型检验方法需要特殊的逻辑知识和好的工具支持才能应用到仿真领域,因而减缓了形式化验证方法应用到仿真模型验证领域的步伐.

事件图[4]是一种基于事件调度仿真策略的图形化离散事件仿真建模方法,它将事件作为基本建模单元,按照事件发生的先后顺序不断执行相应的事件,从而改变仿真系统的状态,它在许多方面是离散事件仿真的机器语言[5].文献[4]证明,事件图具有与图灵机一样的表达能力,能够对所有能由计算机实现的系统进行建模.但是,目前各种成熟的模型检验工具所用的性质描述语言大都是基于状态或属性的时态逻辑,如 LTL(linear temporal logic),CTL(computation tree logic),PSL(property specification language)以及它们的各种变形,不能够直接用于表达基于事件的性质[1].部分研究者也提出了对事件驱动的模型进行模型检验的方法:一种是利用事件与状态的关系,在状态与事件之间进行转换[6];另一种是提出新的基于事件的时态逻辑语言,Jahanian 和 Mok 最先将带有时间戳的事件引入到线性时态逻辑中,提出了实时逻辑 RTL(real-time logic)[7];Yang 等人设计了一种同步实时事件逻辑 SREL(synchronous real-time event logic)[8]等.但是,这些基于事件的时态逻辑都是针对特殊的模型设计的,例如,SREL 是对基于 ModeChart 的模型进行性质描述的.这些方法提出较早且具有一定的针对性,不利于应用到对事件图模型的描述中.行为时态逻辑 TLA(temporal logic of action)[9]是由 Lamport 于 20 世纪 90 年代初提出的将时态逻辑与行为逻辑相结合的一种新的逻辑,是目前模型检验方法中一个较新的研究方向.TLA 模型的基本单元是动作(action),它将状态变量的改变刻画为动作,与事件图通过事件来改变系统状态在本质上是一样.TLA 模型检验方法的成功应用,为基于事件图的模型检验提供了可能性.

课题组已经实现了基于事件图的仿真模型并行模型检验方法[10],但是该方法中的性质描述语言仍然是基于状态的 LTL 语言,用它对事件图模型应满足的性质进行规约较为繁琐,而且不直观.因此,为了让离散事件仿真领域人员能够直观、简便地利用模型检验方法对基于事件图构建的仿真模型进行形式化验证,提高模型的逻辑正确性,本文首先提出了一种事件时态逻辑(event temporal logic,简称 ETL)简单、直观地描述事件之间的时序、逻辑关系.然后,提出了面向事件图和 ETL 的模型检验方法.

本文第 1 节首先针对事件图的特点对事件时态逻辑的各个要素进行定义和说明,提出模型实例化、取消边操作、时间约束以及同时事件优先级的时态逻辑表示方法.第 2 节详述事件时态逻辑 ETL 的语法、语义及 ETL 的易用性、可用性.第 3 节介绍基于自动机理论的面向事件图和 ETL 的模型检验方法框架,详述 ETL 公式转换为 Büchi 自动机的方法.第 4 节结合实例说明 ETL 如何描述事件图模型的性质,并通过实验验证面向事件图和 ETL 的模型检验方法的有效性.最后对全文进行总结.

1 事件时态逻辑

TLA 以动作作为基本建模单元,与事件图中的事件在本质上是相同的.与基于 Petri 网和时间自动机的检验方法相比,它们都是通过引起系统状态变化的动作/事件来对系统进行描述.显然,前者更符合离散事件建模与仿真的思想.但是,由于 TLA 能够在一种语言中同时表达模型行为与逻辑规则,但也增加了 TLA 语言本身的复杂度.因此,本文针对事件图以事件作为基本建模单元的特点,在 LTL 的基础上,参考 TLA 基于动作对模型性质

进行规约的思想,定义了事件时态逻辑 ETL,使得仿真领域的用户对基于事件图的仿真模型的性质规约过程更加直观、简便.

定义 1(事件(event)). 事件是系统中一种不具持续性的动作,它具有瞬时性,在某个特定的时间点上发生,可以使系统从一个状态迁移到另一个状态.

定义 2(状态(state)). 状态是对系统在某一个时间段内的描述,在一定的时间间隔内保持不变.

系统的一个状态就是对状态变量的一个赋值,而事件则是用来指定状态变量是如何发生变化的,即表示状态变量在新状态和旧状态关系的公式.一对连续的状态序列称为一个事件,以离散事件仿真中典型的单服务员排队系统为例,[Q=0]→[Q=1]定义为一个到达事件(arrival),其中每一个状态是对状态变量的一个赋值:[Q=1]是一个系统状态,表示状态变量 Q 的值为 1.

定义 3(状态函数(state function)). 状态函数是包含状态变量和常量的一类表达式.

定义 4(状态谓词(state predicate)). 状态谓词是值为布尔型的状态函数.

在事件时态逻辑中,用公式$[E]_f$表示一个事件,其中:E 是事件 *event*;f 是状态函数,表示事件的发生将根据状态函数 f 定义的表达式来改变系统状态.通常情况下,事件的发生会引起系统从一个状态转换到另一个状态.但是在某些情况下,为了便于系统建模,允许某些事件的发生不引起系统状态迁移.这样不会引起状态迁移的事件,本文称为静事件(static event),表示为$\{E\}_{f'=f}$或者省略状态变量$\{E\}$;相反,总是会引起系统状态变量发生变化的事件称为动事件(dynamic event),表示为$\langle E\rangle_f$.为了方便起见,这类事件可以直接表示为 E.

当事件的激活条件满足时,称该事件被激活,用 *ENABLE* 来表示.

定义 5[11]. $ENABLED\ E=\exists c_1,...,c_n: E(c_1/v_1',...,c_n/v_n'), s[[ENABLED\ E]]=\exists t\in S: s[[E]]t$, 其中,$s[[E]]t=E(\forall v: s[[v]]/v, t[[v]]/v')$表示在事件 E 的作用下,状态 s 的下一状态是 t.

例如,在单服务员排队系统中,一个服务事件被激活,当且仅当队列 $Q\neq 0$ 且 S=1.事件图中,有些事件的发生具有一定的时间延迟,因而这类事件的激活条件通常还需要定义其时延.例如,一个离开事件被激活,当且仅当一定的服务时间达到后,即 $t_s\geqslant servicetime$.

1.1 事件图中模型实例化的表示方法

事件图通过参数化的有向边表示模型的不同实例.例如,在流水线系统的事件图模型中(如图 1 所示)[12],到达的顾客由 n 个工作站串行地处理其服务需求,每个工作站相当于一个排队系统.当顾客在一个工作站的服务完成之后,该顾客将进入到下一个工作站接受服务.当顾客在所有工作站的服务都完成之后,他将离开该系统.图 1 中,事件所带的参数表示该事件发生的工作站标号,该参数值由调度边上的表达式来确定.为了避免将每个工作站的状态变化都一一表达出来,同时也为了避免过多地引入状态变量,ETL 采用以下表达方法:

(1) 将两个变量 S,Q 用一个含有两个域的不变量 *Record* 来表示,例如,[*Record*=0,1],表示 Q=0,S=1.更一般地,假设 r 是一个含有 n 个域的记录,$r.Q$ 表示它的 Q 域,$r.S$ 表示它的 S 域.*Record* 的声明定义为 *Record*=[Q:*Nat*, S:{0,1}],表示了不同变量的取值范围,这样的声明称为类型不变量声明.声明中的域可以是无序的,r 只是所有这样记录中的一个元素.当系统规模很大时,这样的表示方法将会减少很多状态变量的引入.

(2) 不同模块的实例描述方法.例如,一个工厂的生产线以看作是由多个单服务员模型组成的系统(如图 1 所示)[12],工件的每个加工阶段可以看作是单服务员模型的一个实例.假设一个工件在生产线上需要经过打磨、锻造、装配等几个阶段的处理,建模时就需要用单服务员模块的 3 个实例分别表示.在事件图模型中,这种实例化是通过参数化的边来实现的;在事件时态逻辑中,实例化通过 $E(i)$来定义,*Arrival*(0),*Arrival*(1),*Arrival*(2)分别表示了工件处理 3 个阶段中的到达事件($0\leqslant i\leqslant n-1$,$n$ 表示实例化的个数).不同实例中的状态变量、参数以及属性值的表示方法都不相同,因此只需将每个实例中与单服务员模块不同的部分进行替换即可.

例如,*Queue*(1)=*Queue* WITH $Para_1\leftarrow Para_2, Para_3\leftarrow Para_4$ 是对打磨阶段的实例化,其中,WITH $Para_1\leftarrow Para_2, Para_3\leftarrow Para_4$ 表示将 *Queue* 中的参数 $Para_1$ 换成 $Para_2$,$Para_3$ 换成 $Para_4$.如果实例化中只是名称不同,其他信息不需要改变,则省去 WITH 替换部分.

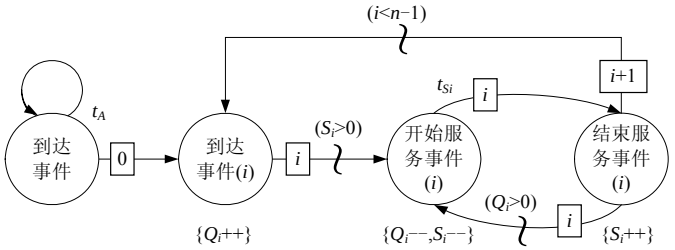


Fig.1 Event graph model of transfer line

图 1 流水线系统的事件图模型

(3) 模型中的事件是可以反复被激活的,因此,ETL 中用关键字"#"来表示 ETL 中事件的发生次数,#(*i*)*E* 表示事件 *E* 的第 *i* 次发生.例如,在单服务员排队系统中需要保证第 *i* 个到达的顾客最终会被服务,可以表示为 $[\#(i)Arrival]_{q^{++}} \rightarrow \Diamond[\#(i)Server]_{q^{--}}$.那么根据上述规则,流水线中的第 *k* 个到达打磨模块的工件最终会被加工这一公平性原则可以表述为$\#(k)Arrival(i) \rightarrow \Diamond \#(k)Server(i)$.

1.2 事件图中取消边的表示方法

事件图通过取消边来处理已经被调度到未来事件列表中但需要取消的事件.虽然取消边操作的引入并没有扩展事件图的建模能力,但是它增强了事件图对某些特定系统的表达能力,因而为了便于 ETL 表达有这样特定需求的系统,本文定义了取消算子∇.假设有这样一个单服务员排队系统,顾客分为 VIP 会员和非 VIP 会员两种,当一个 VIP 会员到达后,如果未来事件列表中已经调度了一个非 VIP 会员的服务事件,那么将取消该服务事件,将 VIP 会员的服务事件置于未来事件列表的首位.事件图的取消操作在 ETL 中可以表达为 $\varphi_1 \nabla \varphi_2$,读作:"$\varphi_1$ 取消 φ_2".我们用 *ArrivalVip* 表示 VIP 顾客的到达事件,*Server* 表示普通顾客的服务事件,那么 VIP 的到达事件将取消非 VIP 的服务事件,可表示为$\Box(ArrivalVip \nabla Server)$.

1.3 事件图中时间元素的表示方法

在事件图中,有的事件之间的调度关系需要时间延迟,有的则不需要.时间元素是事件图的重要组成部分,表示了事件调度的时间约束,而 LTL 则无法描述这样的时间约束关系.为了能够表达事件调度的时间约束,本文在 ETL 中引入了时间表示方法.

定义 6(时间约束). *T* 是一个时间变量集合,时间约束 γ 的集合定义为 $\gamma ::= [t_1,t_2] | (t_1,t_2) | [t_1,t_2) | (t_1,t_2] | \neg\gamma | \gamma_1 \wedge \gamma_2$,其中,$t_1,t_2 \in \mathfrak{R}^+$.例如,$E_1 \rightarrow \Diamond_{[0,t)} E_2$ 表示事件 E_1 发生后,事件 E_2 将在 *t* 个时间单位内发生.

事件之间的调度关系可由以下几个表达式定义,E,E_1,E_2 表示事件:

(1) 当 E_1 发生后,E_2 将在 *t* 个单位时间内发生:$E_1 \rightarrow \Diamond_{[0,t]} E_2$;

(2) 当 E_1 发生后,E_2 必须在 *t* 个单位时间后立刻被调度:$E_1 \rightarrow \Box_{[t,t]} E_2$;

(3) 连续两次事件 *E* 的发生时间间隔至少是 *t* 个时间单位:$E \rightarrow \Box_{(0,t)} \neg E$;

(4) 连续两次事件 *E* 的发生时间间隔等于 *t* 个单位时间:$E \rightarrow (\Box_{(0,t)} \neg E) \wedge (\Box_{[t,t]} E)$.

仍以图 1 所示的排队系统为例,连续两次 *Arrival* 事件之间的时间间隔为 t_A 个时间单位,用 ETL 可表示为

$$Arrival \rightarrow (\Box_{(0,t_A)} \neg Arrival) \wedge (\Box_{[t_A,t_A]} Arrival).$$

StartService 事件在 t_S 个时间单位后调度 *EndService* 事件,在 ETL 中表示为

$$StartService \rightarrow (\Box_{(0,t_S)} \neg EndService) \wedge (\Box_{[t_S,t_S]} EndService).$$

事件之间的取消关系是不需要任何时间延迟的,$E_1 \nabla E_2$ 表示只要事件 E_1 发生,事件 E_2 立刻被取消,因而取消算子没有时间约束.

1.4 事件图中事件优先级的表示方法

事件图采用对调度边设置优先级的方法解决同时事件的执行顺序,同时,事件处理的不得当可能会引起仿

真结果的不正确.在 ETL 中,算子▷(precedence)表示事件之间的优先级关系.$\Diamond_{[t,t]}(E_1\wedge E_2)\rightarrow E_1\triangleright E_2$ 表示如果在 t 时刻 E_1 和 E_2 事件同时被调度,那么 E_1 优先于 E_2 执行.

假设在排队系统模型中机器会间歇性发生故障,那么当机器发生故障时,需要及时进行维修(Repair),维修事件的优先级就高于其他任何事件.因而当维修机器事件与其他事件同时发生时,系统应优先处理机器维修事件.上述性质在 ETL 中可以表示为$\Diamond_{[t,t]}(Repair\wedge E)\rightarrow Repair\triangleright E$,其中,$E$ 表示模型中的其他事件.

2 ETL 的语法及语义

ETL 是由符号、公式、定理以及推理规则组成的逻辑形式系统,ETL 用到的符号见表 1.

Table 1 Symbols in ETL

表 1 ETL 中的符号

名称	符号
事件	$E,E_1,E_2,\ldots,E_n$
状态变量	$x,y,z,\ldots$
状态函数	f, g
状态谓词	P, Q, R
时态算子	$\Box, \Diamond, \mathrm{O}, \rightarrow, \cup, R, \nabla, \triangleright$
逻辑算子	$\neg, \wedge, \vee$
量词	$\forall, \exists$
其他符号	$=, \neq, \langle, \rangle, [,], (,), \{, \}, \in, \notin$

2.1 ETL的语法和语义

定义 7. 给定一个原子命题集合 AP,ETL 公式的语法:

(1) 如果 $p\in AP$,那么 p 是一个 ETL 公式;

(2) 如果 f 和 g 是 ETL 公式,那么$\neg f,f\wedge g,f\vee g,\Box f,\Diamond f,\mathrm{O}f,f\cup g,f\mathrm{R}g,f\triangleright g,f\nabla g$ 都是 ETL 公式.

定义 8. 归纳定义 ETL 公式表示为

$$\varphi::=e|\mathrm{true}|\mathrm{false}|\neg\varphi|\varphi_1\vee\varphi_2|\Diamond_{[l,u]}\varphi|\Box_{[l,u]}\varphi|\mathrm{O}\varphi|\varphi_1\nabla\varphi_2|\varphi_1\mathrm{R}\varphi_2|\varphi_1\triangleright\varphi_2|\varphi_1\rightarrow\varphi_2|\varphi_1\cup\varphi_2,$$

其中,

- E:事件;
- true,false:逻辑常量,表示真、假;
- ◇:finally 算子,将来会有φ,$[l,u]$表示时间约束,$\Diamond_{[0,t)}E$ 表示事件 E 将在 t 个单位时间内发生.若$[l,u]$为$[0,+\infty]$,那么时间约束可以省略;
- □:always 算子,总是会有φ,“必然φ”或“总是φ”;
- O:下一时刻算子(next);
- ∪:直到算子(until);
- →:蕴含算子(lead to),它可以由基本逻辑算子¬,∨来定义$\varphi_1\rightarrow\varphi_2=(\neg\varphi_1\vee\varphi_2)$;
- R:释放算子(release),是 until 算子的逻辑对偶算子,$\varphi_1\mathrm{R}\varphi_2=\neg(\neg\varphi_1\cup\neg\varphi_2)$;
- ∇:取消算子(cancel);
- ▷:优先级算子(precedence).

取消算子∇与 until 算子类似,但不同的是:until 算子表达的意思是,φ_2 成立之前,φ_1 一直为真;而取消算子恰恰相反,φ_1 成立时,φ_2 必定不成立(如图 2 所示).取消算子可以通过 until 算子来定义:$\varphi_1\nabla\varphi_2=(\neg\varphi_1\cup\neg\varphi_2)\wedge\varphi_1$,读作 φ_1 取消φ_2.例如,在单服务员排队系统中,如果某台机器在处理任务时发生故障(failure),那么它将取消已经调度的第 1 个结束服务事件(EndService),表示为 $EndService\nabla Failure$.

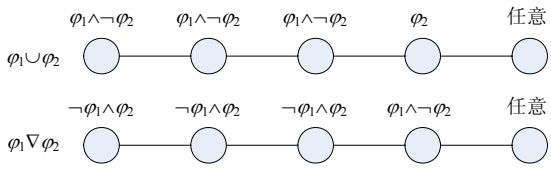


Fig.2 until operator and cancel operator

图 2 until 算子和取消算子

▷算子也可以用∪算子来定义[13],$E_1 \triangleright E_2 \equiv \neg((\neg E_1) \cup E_2)$.

上述算子的优先级如下:一元算子的优先级高于二元算子;一元算子¬和 O 具有相同的优先级;∪,∇和▷算子的优先级高于∧,∨,→,□;◊的优先级最低.

定义 9(路径(trace)). 一条路径是由一组事件序列组成的元组$\langle E_1,E_2,\ldots,E_n\rangle$,记为$\rho$.不包含任何事件的路径称为空路径,表示为$\varepsilon$.路径的长度是其所包含的事件的总数(包含重复事件).

定义 10(ETL 公式的语义). 对于字母集 2^{AP} 上的一个无穷路径$\rho=s_1,s_2,\ldots$,其中,$\rho^i=\rho_i\rightarrow\rho_{i+1}\rightarrow\rho_{i+2}\rightarrow\ldots$,那么 ETL 公式的解释如下[14]:

- $\rho\models p$ iff $p\in x_0$, for $p\in AP$;
- $\rho\models\neg f$ iff $\rho\not\models f$;
- $\rho\models f\wedge g$ iff $\rho\models f$ and $\rho\models g$;
- $\rho\models f\vee g$ iff $\rho\models f$ or $\rho\models g$;
- $\rho\models \mathrm{O}f$ iff $\rho_1\models f$;
- $\rho\models f\cup g$ iff there is an $i\geqslant 0$, such that $\rho^i\models g$ and $\rho^j\models f$ for all $0\leqslant j<i$;
- $\rho\models f\mathrm{R}g$ iff $\rho\models\neg(\neg f\cup\neg g)$;
- $\rho\models\Diamond f$ iff $\rho\models \mathrm{true}\cup f$;
- $\rho\models\Box f$ iff $\rho\models\neg\Diamond\neg f$;
- $\rho\models f\triangleright g$ iff $\rho\models\neg(\neg f\cup g)$;
- $\rho\models f\nabla g$ iff $\rho\models(\neg f\cup\neg g)\wedge f$.

定理 1. 每个 ETL 公式都有相应的 LTL 公式与之对应.

证明:由于每个状态都可以由一对事件来表示:$s=(I_e,T_e)$,其中,I_e为起始事件,T_e为终止事件,且$I_e\cap T_e=\varnothing$.同样,每个事件也都可以由一对状态来表示:$e=(I_s,T_s)$,其中,I_s为初始状态,T_s为终止状态.但是由于 ETL 允许有静态事件,因而初始状态 I_s可能等于终止状态 T_s,即 $e=(I_s,T_s)$且 $I_s=T_s$.因此,每个 ETL 公式都有 LTL 公式与之对应. □

2.2 ETL对事件图模型性质的描述

本节以实例来说明 ETL 的简单、易用性.为了对 LTL 性质表述得方便,本节的事件图模型将状态进行了显示标记,用方框表示,事件用圆圈表示.

如图 3 所示,一旦信号灯变红后,那么它在变为绿色之前必然先变为黄色,该性质用 LTL 语言可以表示为$\Box(red\rightarrow \mathrm{O}(red\cup(yellow\wedge(yellow\cup green))))$[15],而用 ETL 语言则可表示为

$$\Box(turnred\rightarrow \mathrm{O}(turnyellow\rightarrow \mathrm{O}turngreen)).$$

图 4(a)是以办理出国手续为例[16]的事件图模型.在该模型中,要求只能是得到签证后的公民才能进入要去的国家.通常,基于 LTL 可以将该性质描述为$\Box(in\ country\rightarrow\Diamond stamp)$,其中,*in country* 表示已经处于要到达的国家,*stamp* 表示获得了签证.但是这条 LTL 性质并不严谨,具有一定得歧义.例如,考虑有人非法伪造签证,那么如图 4(b)所示模型也满足上述 LTL 公式表示的性质.但是,任何国家都不希望一个伪造签证者进入他们的国家,因此,性质$\Box(in\ country\rightarrow\Diamond stamp)$对于添加的伪造签证的转换关系也是有效的.因此,建模者需要清楚地表达通过

入境检查的人必须得到了有效的签证.基于 ETL 可以简单表示上述性质而不存在歧义:

$$\Box(check \rightarrow \Diamond get\ visa).$$

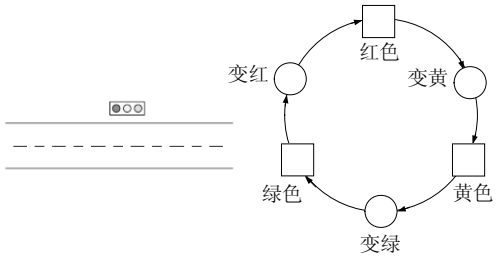


Fig.3 Event graph model of traffic light

图 3 交通信号灯系统的事件图模型

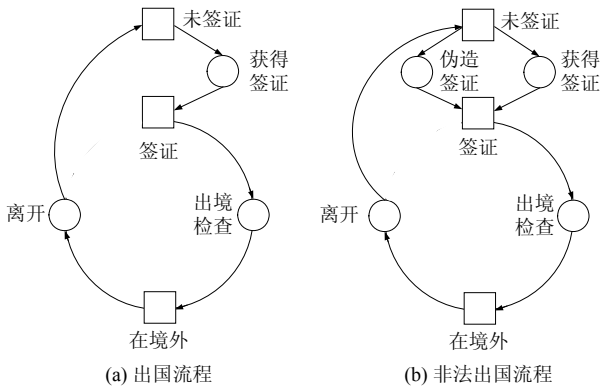


Fig.4 Event graph model of enter a foreign country

图 4 出国流程的事件图模型

从上述两个例子可以看出,与 LTL 相比,ETL 在对事件图模型进行规约时更为简单、直观、易懂,在某些情况下还能够消除歧义.

定理 2. ETL 具有不弱于 LTL 的表达能力.证明类似于定理 1.

评价一种时态逻辑可用性的标准包含以下 3 个方面[17]:

(1) 直观性:用该种时态逻辑对模型应满足的性质规约时是否直观、易懂;

(2) 表达能力:该种时态逻辑是否有足够的能力来对待验证的性质进行规约;

(3) 易实现性:以该种时态逻辑作为性质规约语言的模型检验算法是否易于实现.

由本节的实例可以看出,ETL 在对事件图模型进行性质规约时直观、易懂;由定理 2 以及本节实例可以看出,ETL 具有不弱于 LTL 的表达能力,而且在有些性质表达时不仅简单,还能消除歧义;由定理 1 可知,每个 ETL 公式都有与之相对应的 LTL 公式,因此,通过有效的转换算法将 ETL 公式转换为 LTL 公式,就能实现以 ETL 作为性质规约语言的模型检验.综上所述,对基于事件图的模型进行性质规约的时态逻辑语言 ETL 是可用的.

3 面向事件图和 ETL 的模型检验方法

3.1 面向事件图和ETL的模型检验方法框架

课题组已经实现了在并行模型检验环境 DiVinE(distributed and parallel verification environment)下对基于

事件图的仿真模型进行形式化验证的方法[10],但是该方法使用的仍然是基于 LTL 的性质规约语言.为了使仿真领域人员直观、简便地利用模型检验方法对基于事件图构建的系统模型进行形式化验证,本文构建了一个面向事件图和 ETL 的模型检验环境,该环境仍以 DiVinE 作为基础验证平台.

DiVinE 是一种基于 LTL 的模型检验环境,它以自动机理论为基础,基本思想是:对于给定的模型 M 以及 LTL 公式 φ,要检验 $M\models\varphi$是否成立时,首先将模型 M 转化为 Büchi 自动机 A_M,使得 $L(A_M)=L(M)$,并且由 LTL 公式 φ得到 Büchi 自动机 $A_{\neg\varphi}$,然后构造 A_M 与 $A_{\neg\varphi}$的乘积 $A_M\otimes A_{\neg\varphi}$,使得 $L(A_M\otimes A_{\neg\varphi})=L(A_M)\cap L(A_{\neg\varphi})$.这样,$M\models\varphi$当且仅当 $L(A_M\otimes A_{\neg\varphi})=0$.因此,基于自动机理论的模型检验方法最终转换为对自动机 $A_M\otimes A_{\neg\varphi}$接收的语言的判空问题:语言交集若为$\varnothing$,则 $M\models\varphi$;否则,$M\not\models\varphi$,并且 $L(A_M\otimes A_{\neg\varphi})$中的任意一个线性结构都是 M 中违反φ的执行序列[15].文献[10]中已经实现了事件图模型到 DiVinE 所接受的自动机模型的转换,因而本文只需要将 ETL 描述的性质公式φ的否定形式$\neg\varphi$转换为相应的 Büchi 自动机 $A_{\neg\varphi}$输入到 DiVinE 中,然后计算两个自动机的同步积,得到它们所能共同接收的语言集并对该语言集进行判空:如果语言集为$\varnothing$,表示模型满足性质;否则,说明模型不能满足性质,并且该语言集是违反该性质的反例.

因此,对一个事件图模型 M 是否满足 ETL 性质规约 F 的检验步骤可以归纳如下:(1) 输入模型 M 和性质规约 F;(2) 将事件图模型 M 转换为相应的 DVE 模型 M',模型转换器的具体转换过程见文献[10];(3) 将 ETL 公式 F 转换为相应的 LTL 公式 F';(4) 利用 DiVinE 工具验证模型 M'是否满足性质 F'.本文的验证环境结构如图 5 所示.

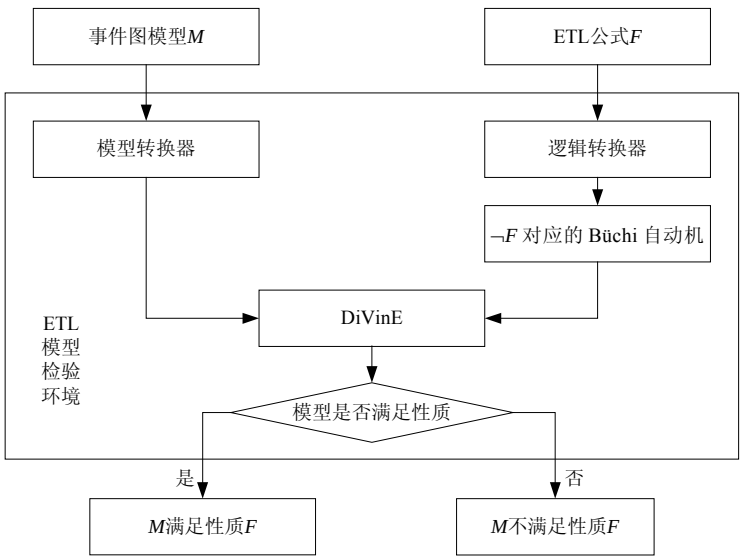


Fig.5 Event graph and ETL oriented model checking framework

图 5 面向事件图和 ETL 验证环境结构图

3.2 逻辑转换器

逻辑转换器的功能是将 ETL 公式φ的否定形式转换为 Büchi 自动机 $A_{\neg\varphi}$,它首先根据定理 1 将 ETL 公式φ转换成等价的 LTL 公式,然后由 LTL 公式到 Büchi 自动机的转换算法得到 ETL 公式φ的 $A_{\neg\varphi}$.

定义 11. Büchi 自动机是一个 5 元组[18]:$BA=\langle S,\Sigma,\Delta,Q_0,F\rangle$,其中:$S$ 是有限的状态集合;Σ是有限的字母集合,实际上,该字母集合相当与事件图中的事件集合,是触发状态发生变化的;$\Delta\subseteq S\times\Sigma\times S$ 是一个有字母标识的迁移关系;$Q_0\subseteq S$ 是初始状态集合;$F\subseteq S$ 是可接受状态集合.

Büchi 自动机的一个有限路径是:满足对所有 $0\leqslant i<n$ 存在 $a_i\in\Sigma$,使得$(s_i,a_i,s_{i+1})\in\Delta$的状态序列$\rho=s_0s_1s_2\ldots s_n$.

Büchi 自动机的一个无限路径是:对所有 $0\leqslant i$ 存在 a_i,使得$(s_i,a_i,s_{i+1})\in\Delta$的状态序列$\rho=s_0s_1s_2\ldots$.字母表$\Sigma$上的一个无限字符串$\omega=a_0a_1a_2\ldots$对 Büchi 自动机来说是可接受的,当且仅当存在一个无限路径$\rho=s_0s_1s_2\ldots$满足$s_0\in Q_0$,$(s_i,a_i,s_{i+1})\in\Delta$且$inf(\rho)\cap F\neq\varnothing$,其中的$inf(\rho)$表示在路径$\rho$中出现无限多次的状态的集合.

根据 ETL 模型检验框架,在转换之前需先将 ETL 公式φ转换为否定形式$\neg\varphi$,而且该时态逻辑公式中只能包含$\cup$,R,O 这 3 种时态算子以及布尔操作符$\wedge$,$\vee$,$\neg$,那么其他时态算子根据算子之间的关系可以进行重写、化简:

- $\Diamond\varphi$=True$\cup\varphi$;
- $\Box\varphi$=FalseRφ=$\neg\Diamond\neg\varphi$;
- $\varphi_1\mathrm{R}\varphi_2=\neg(\neg\varphi_1\cup\neg\varphi_2)$;
- True=$p\vee\neg p$;
- False=$\neg$True;
- $\varphi_1\wedge\varphi_2=\neg(\neg\varphi_1\vee\neg\varphi_2)$;
- $\varphi_1\vee\varphi_2=\neg(\neg\varphi_1\wedge\neg\varphi_2)$;
- $\neg(\varphi_1\cup\varphi_2)=(\neg\varphi_1)\mathrm{R}(\neg\varphi_2)$;
- $\neg(\varphi_1\mathrm{R}\varphi_2)=(\neg\varphi_1)\cup(\neg\varphi_2)$;
- $\neg\Box\varphi=\Box\neg\varphi$
- $\varphi_1\rightarrow\varphi_2=(\neg\varphi_1)\vee\varphi_2$;
- $\varphi_1\leftrightarrow\varphi_2=((\neg\varphi_1)\vee\varphi_2)\wedge((\neg\varphi_1)\vee\varphi_2)$;
- $\neg(\varphi_1\wedge\varphi_2)=(\neg\varphi_1)\vee(\neg\varphi_2)$;
- $\neg(\varphi_1\vee\varphi_2)=(\neg\varphi_1)\wedge(\neg\varphi_2)$.

假设某机器总是会周期性地发生故障,当机器发生故障时,工作人员将立即维修该机器.这条性质用 ETL 公式可以表示为$\varphi=\Box(Failure\rightarrow \mathrm{O}Repair)$,表示不论何时,当 *Failure* 事件发生时,系统的下一个事件将是 *Repair*.该 ETL 公式φ的否定形式为$\neg\varphi=\neg\Box(Failure\rightarrow \mathrm{O}Repair)$.

根据重写、化简规则$\neg\varphi=\neg\Box(Failure\rightarrow \mathrm{O}Repair)$可以重写为

$$\begin{aligned}\neg\Box(Failure\rightarrow \mathrm{O}Repair)&=\neg\Box(\neg Failure\vee \mathrm{O}Repair)\\&=\neg(\mathrm{False}\mathrm{R}(\neg Failure\vee \mathrm{O}Repair))\\&=(\neg\mathrm{False})\cup\neg(\neg Failure\vee \mathrm{O}Repair)\\&=\mathrm{True}\cup(Failure\wedge\neg \mathrm{O}Repair)\\&=\mathrm{True}\cup(Failure\wedge \mathrm{O}\neg Repair).\end{aligned}$$

根据 Giannakopoulou 等人提出的 LTL 公式转换为 Büchi 自动机的高效方法[14],可以得到 ETL 公式$\neg\Box(Failure\rightarrow \mathrm{O}Repair)$的 Büchi 自动机如图 6 所示.

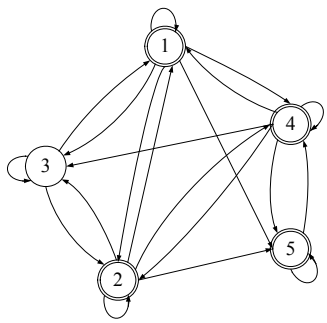


Fig.6 Büchi automaton for $\neg\Box(Failure\rightarrow \mathrm{O}Repair)$

图 6 公式$\neg\Box(Failure\rightarrow \mathrm{O}Repair)$对应的 Büchi 自动机

4 ETL 的实例分析

本节以滑雪场管理系统的仿真模型[19]为例验证本文所提 ETL 语言的可用性,该系统对应的事件图模型如图 7 所示.本文实验主要关注的问题是:面向事件图和 ETL 的模型检验方法能否验证事件图模型是否满足由本文所提出的 ETL 公式表示的性质.本文的实验环境为双向 2.53GHz 的四核 Xeon 处理器 E5540,8GB RAM,内核 2.6.18 的 Linux 操作系统.

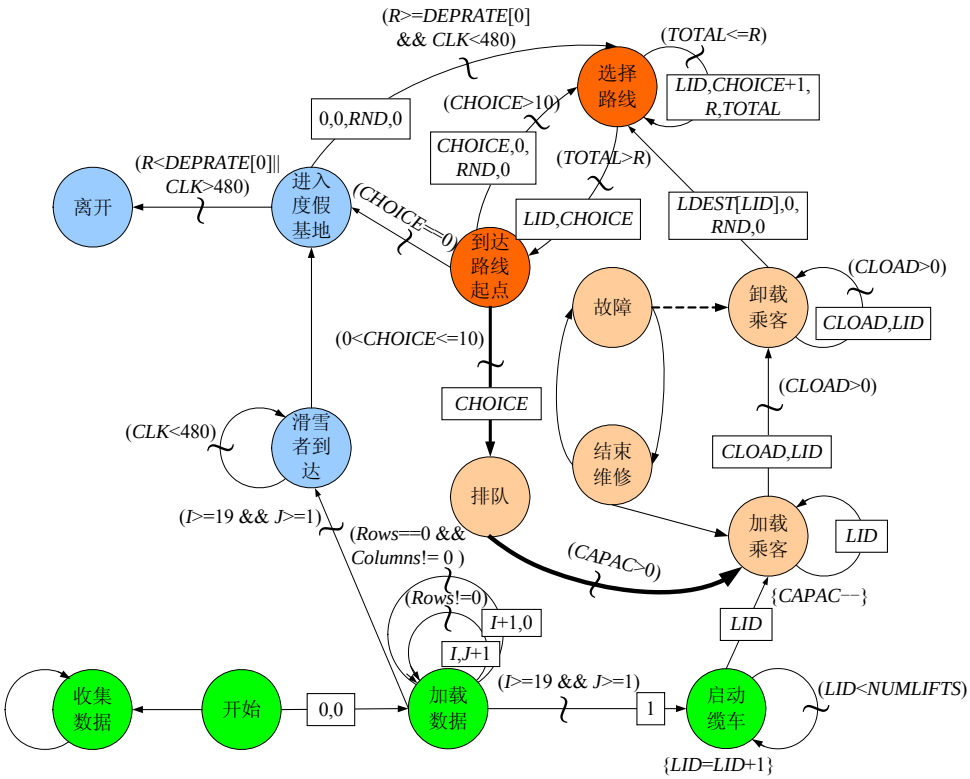


Fig.7 Event graph model of Ski base system

图 7 滑雪场系统的 EEG 模型

本文设计对模型是否满足如下性质进行检验:

(1) 对于所有等待缆车的滑雪者,如果缆车座位有空闲,那么必然会搭载顾客,其中,*CAPAC* 表示缆车的最大乘客量.

$$\Box(Queue \wedge CAPAC>0 \rightarrow \Diamond UnLoadLift).$$

(2) 缆车发生故障的情况下,不可能有滑雪者被运往目的地.

$$\Box(Failure \rightarrow \Diamond Failure \nabla UnloadLift).$$

根据第 3 节提出的面向事件图和 ETL 的模型检验方法,将事件图以及 ETL 描述的性质作为输入,分别由模型转换器和逻辑转换器转换为两个 Büchi 自动机,然后根据这两个自动机的同步积进行语言交集判空.首先,我们去掉图 7 模型中的一条事件调度边(如图 7 中的粗线所示),对模型进行了安全性验证,验证结果如图 8(a)所示.结果表明,该模型具有死锁状态.这是由于顾客完成路线选择后等待缆车,但是接下来却没有登缆车事件,因而出现了死锁,致使滑雪者处于无限等待状态.修改后模型的安全性以及上述性质(1)、性质(2)的验证结果如图 8(b)~图 8(d)所示.本文用 DiVinE 工具的建模语言和 LTL 性质公式对上述模型进行了验证,得到了同样的验证结果,即模型不存在死锁,并且满足上述两条性质.可以看出,对于仿真领域的用户而言,面向事件图和 ETL 的模型

检验方法能够降低模型检验的应用难度,使得检验过程更为直观、简便.

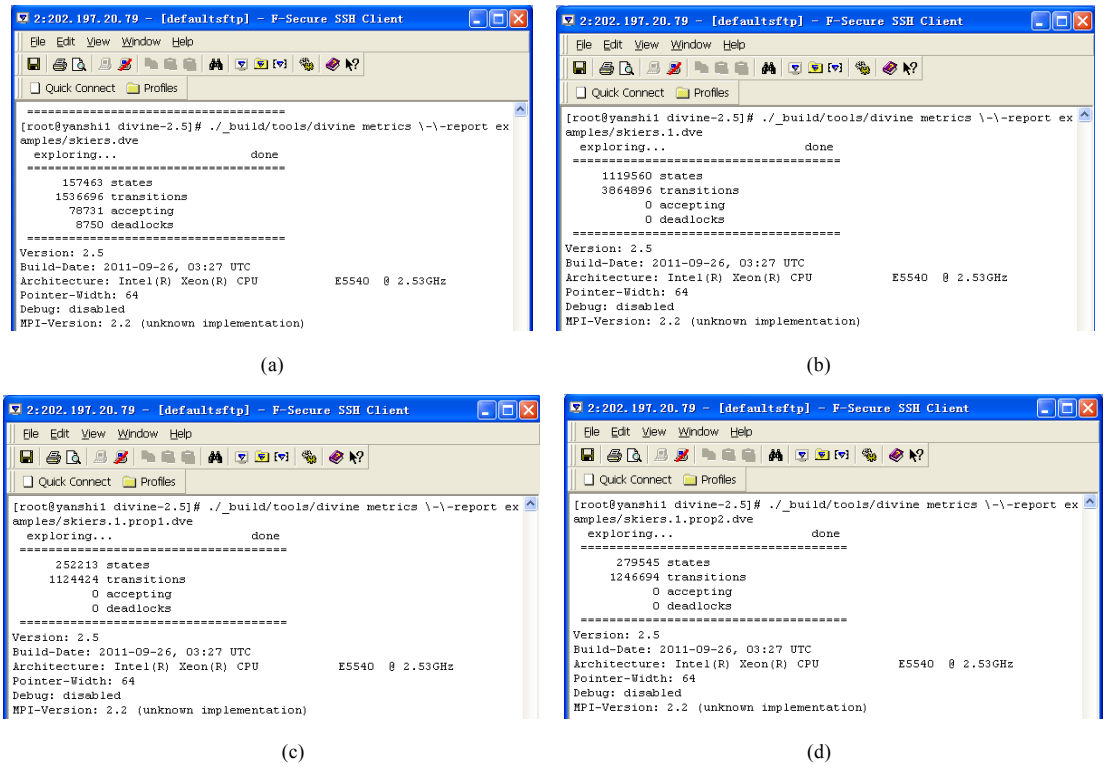


Fig.8 Verification results in DiVinE

图 8 DivinE 的验证结果

5 总 结

为了便于仿真人员对事件图模型进行形式化验证,本文首先提出了一种事件时态逻辑语言 ETL,它针对事件图的特点增加了事件取消操作、模型实例化、时间约束和同时事件优先级的时态逻辑表述.然后,给出了基于自动机理论的面向事件图和 ETL 的模型检验方法,使得仿真领域的用户能够较为直观、简便地对基于事件图的仿真模型进行形式化验证.实验结果验证了 ETL 的可用性以及该方法的有效性.

References:

[1] Clarke EM, Emerson EA, Sifakis J. Model checking: Algorithmic verification and debugging. Communications of the ACM, 2009, 52(11):75−84. [doi: 10.1145/1592761.1592781]

[2] Clarke EM. The birth of model checking. In: Grumberg O, Veith H, eds. Proc. of the 25 Years of Model Checking Festschrift. LNCS 5000, Heidelberg: Springer-Verlag, 2008. 1−26. [doi: 10.1007/978-3-540-69850-0_1]

[3] Zhang PC, Zhou Y, Li BX, Xu BW. Property sequence chart: Formal syntax and semantic. Journal of Computer Research and Development, 2008,45(2):318−328 (in Chinese with English abstract).

[4] Savage EL, Schruben LW, Yücesan E. On the generality of event-graph models. INFORMS Journal on Computing, 2005,17(1):3−9. [doi: 10.1287/ijoc.1030.0053]

[5] Fujimoto RM. Parallel and Distributed Simulation Systems. New York: A Wiley-Interscience Publication, John Wiley & Sons, 2000. 41−47.

[6] Atlee JM, Gannon J. State-Based model checking of event-driven system requirements. IEEE Trans. on Software Engineering, 1993, 19(1):24−40. [doi: 10.1109/32.210305]

[7] Farn W. Formal verification of timed systems: A survey and perspective. Proc. of the IEEE, 2004,92(8):1283−1305. [doi: 10.1109/JPROC.2004.831197]

[8] Yang J, Aloysius KM, Farn W. Symbolic model checking for event-driven real-time systems. ACM Trans. on Programming Languages and Systems, 1997,19(2):386−412. [doi: 10.1145/244795.244803]

[9] Lamport L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Boston: Addison-Weslet, 2002. 15−73.

[10] Xia W, Yao YP, Mu XD. Parallel model checking for discrete event simulation models based on event graphs. Ruanjian Xuebao/Journal of Software, 2012,23(6):1429−1443 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4047.htm [doi: 10.3724/SP.J.1001.2012.04047]

[11] Wan L. The research of system, rules and checking of protocol based on TLA [Ph.D. Thesis]. Guiyang: Guizhou University, 2009 (in Chinese with English abstract).

[12] Buss AH. Basic event graph modeling. Simulation News Europe, 2001,31(1):1−6.

[13] Bellini P, Mattolini R, Nesi P. Temporal logics for real-time system specification. ACM Computing Surveys, 2000,32(1):12−42. [doi: 10.1145/349194.349197]

[14] Giannakopoulou D, Lerda F. From states to transitions: Improving translation of LTL formulae to Büchi automata. In: Proc. of the FORTE 2002. LNCS 2529, Heidelberg: Springer-Verlag, 2002. 308−326. [doi: 10.1007/3-540-36135-9_20]

[15] Baier C, Katoen JP. Principles of Model Checking. London: The MIT Press, 2008. 19−296.

[16] Kindler E, Vesper T. ESTL: A temporal logic for events and states. In: Desel J, Silva M, eds. Proc. of the ICATPN'98. LNCS 1420, Heidelberg: Springer-Verlag, 1998. 365−384. [doi: 10.1007/3-540-69108-1_20]

[17] Liu WW. Reasoning and symbolic model checking of extended temporal logics [Ph.D. Thesis]. Changsha: National University of Defense Technology, 2009 (in Chinese with English abstract).

[18] Dong W, Wang J, Qi ZC. An approach of model checking UML statecharts. Ruanjian Xuebao/Journal of Software, 2003,14(4): 750−756 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/750.htm

[19] Baumler J, Giaever H, Park K, Tan S, Yang J. Ski lift optimizer final report. 2009. 1−12. http://sybak.com/sites/skilift/final_report.pdf

附中文参考文献:

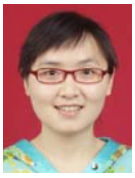
[3] 张鹏程,周宇,李必信,徐宝文.属性序列图:形式语法和语义.计算机研究与发展,2008,45(2):318−328.

[10] 夏薇,姚益平,慕晓冬.基于事件图的离散事件仿真模型并行检验方法.软件学报,2012,23(6):1429−1443. http://www.jos.org.cn/1000-9825/4047.htm [doi: 10.3724/SP.J.1001.2012.04047].

[11] 万良.基于行为时序逻辑 TLA 的系统、规则与协议检测的研究[博士学位论文].贵阳:贵州大学,2009.

[17] 刘万伟.扩展时序逻辑的推理及符号化模型检验技术[博士学位论文].长沙:国防科学技术大学,2009.

[18] 董威,王戟,齐治昌.UML Statecharts 的模型检验方法.软件学报,2003,14(4):750−756. http://www.jos.org.cn/1000-9825/14/750.htm



夏薇(1983－),女,河南信阳人,博士生,主要研究领域为仿真模型验证.
E-mail: weiwei32329@163.com

慕晓冬(1965－),男,博士,教授,博士生导师,主要研究领域为计算机仿真技术.
E-mail: mxd_402@163.com

姚益平(1963－),男,博士,教授,博士生导师,主要研究领域为并行与分布仿真,虚拟现实.
E-mail: ypyao1@tom.com