ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.21, No.2, February 2010, pp.318−333
doi: 10.3724/SP.J.1001.2010.03738
© by *Institute of Software, the Chinese Academy of Sciences*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 面向参数化LTL的预测监控器构造技术*

赵常智+, 董 威, 隋 平, 齐治昌

(国防科学技术大学 计算机学院,湖南 长沙 410073)

## Construction Techniques of Anticipatory Monitors for Parameterized LTL

ZHAO Chang-Zhi+, DONG Wei, SUI Ping, QI Zhi-Chang

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: john9908009@gmil.com, http://www.nudt.edu.cn

**Zhao CZ, Dong W, Sui P, Qi ZQ. Construction techniques of anticipatory monitors for parameterized LTL. *Journal of Software*, 2010,21(2):318−333.** http://www.jos.org.cn/1000-9825/3738.htm

**Abstract**: This paper presents a method of constructing anticipatory monitors for $P_A$LTL (parameterized LTL (linear temporal logic)) based on automata theory. This paper on one hand investigates into the important concepts about the syntax, anticipatory semantics, valuation generation and binding of $P_A$LTL. It is assured that the binding and using are correct in syntax level. Then the concept of parameterized anticipatory monitor is presented consisting of the static part and the dynamic part. The static part is presented as parameterized Büchi automata, and the dynamic part is composed of the valuations of variables in the current state. While the system running, based on the static parameterized Büchi automata, the valuations of variables are dynamically generated and bound from the current state in an on-the-fly fashion, and the anticipatory monitor incrementally checks whether the current running system is satisfied with the given parameterized property. In this process, the parameterized monitor can precisely identify the minimal good/bad prefix of the monitored property.

**Key words**: runtime verification; software monitoring; anticipatory monitor; parameterized LTL (linear temporal logic); parameterized Büchi automata

**摘 要**: 介绍了一种基于自动机理论的参数化LTL(parameterized LTL(linear temporal logic),简称$P_A$LTL)公式运行时预测监控器构造方法.一方面研究$P_A$LTL公式的语法、预测语义、赋值提取以及赋值绑定等重要概念,从语法层面保证公式中参数化变量的正确绑定(binding)和使用(using);另一方面给出参数化预测监控器的概念.它由静态和动态两部分组成,静态部分由参数化Büchi自动机表示,动态部分为当前状态处的变量赋值.在系统运行过程中,预测监控器基于静态部分的参数化Büchi自动机,以on-the-fly的方式在当前状态处动态地提取和绑定变量赋值,递进地验证当前程序运行是否满足指定的参数化性质规约.在该过程中,参数化监控器能够精确地识别被验证性质的最小好/坏前缀.

**关键词**: 运行时验证;软件监控;预测监控器;参数化 LTL(linear temporal logic);参数化 Büchi 自动机

**中图法分类号**: TP311 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60673118, 60725206, 60970035, 90818024, 60803042 (国家自然科学基金)

Received 2009-06-15; Accepted 2009-09-11

© 中国科学院软件研究所 http://www.c-s-a.org.cn

运行时验证(runtime verification,简称 RV)是一种新兴的轻量级验证技术,在运行时验证中,监控器通常从系统需求中产生,通过观察程序的执行来检查程序运行是否满足系统需求,是传统的软件验证和确认技术(比如模型验证和测试)的有效补充.它不但可以有效地检测系统运行中的异常行为,而且也使得在检测到正确性背离时有效修复系统成为可能.

通常,运行时验证器可以在线检查系统的当前执行轨迹.在这种情况下,它被称为在线监控器,以一种递进的方式来检查系统运行,即不需要记录和保存系统的历史执行轨迹.另一方面,监控器也可以通过分析以往执行轨迹(有穷前缀)的记录来分析系统缺陷和异常.该监控过程被称为离线监控.在本文中,我们考虑在线监控以及在在线监控中如何尽可能早地发现软件异常.

伴随运行时验证技术以及软件技术本身的发展,人们希望通过运行时验证技术来验证更为复杂的系统属性,比如,对于程序中普遍存在的动态属性或那些很难通过静态技术精确验证的性质,这类需求往往要求特定类型的所有对象或数据结构的所有实例都满足或不满足某个特定的属性.在现实世界程序中,无论是顺序程序还是并发程序,这类需求都大量存在,例如:任何软件产品都需要满足的一个基本的性质是每一个被打开的文件都应该最终被关闭;在面向对象程序中,经常会要求其动态实例化后产生的所有对象都满足某个特定时序性质;在并发程序中,要求所有动态创建的进程都满足某个性质.对于这类性质,不难发现,仅仅采用命题符号 $P,Q$ 来描述原子命题已经不能满足需要了,需要引入参数化命题,比如 $p(x,y)$,其中,变量 $x$ 和 $y$ 的作用域分别为特定类型的对象集合;同时,为了采用统一的方式来刻画顺序程序和并发程序性质,还需要引入量词$\forall$,$\exists$来修饰相应变量.针对这样的验证需求,人们提出了参数化的性质规约逻辑,其中包括参数化的 LTL(linear temporal logic)规约.

文献[1]对参数化 LTL 的运行时验证技术进行了深入的研究,但由于其采用的监控语义(有穷轨迹语义)的限制,为了保证语义一致性,它从语法上限制参数化 LTL 规约中不能出现 Next(即 $X$)操作符,因此,文献[1]中的参数化 LTL 实际上为 free-$X$ 参数化 LTL.与所有基于有穷轨迹语义的运行时监控器一样,与其相对应的参数化监控器试图以一种推理和演算的方式来逐步说明被验证性质为什么最终为真或为假.虽然该类监控器有助于人们理解为什么被验证性质基于当前系统运行为真或为假,但是它不能最早发现系统异常.与之相对应的一类监控器为预测监控器,它们不但致力于发现系统异常,而且希望尽早地发现异常,为系统修复提供时间和空间.文献[2,3]中分别介绍了面向 LTL 和 TLTL(timed LTL)性质规约的预测监控器构造,文献[4]则扩展了上述工作,提出了面向线性逻辑的预测监控器构造的统一框架.文献[5]基于$\omega$正规语言.论述如何约减相应的 Büchi 自动机的规模以及如何把 Büchi 自动机变换为专用的、静态优化的非确定有穷状态机——二进制迁移树有穷状态机(binary transition tree finite state machine,简称 BTT-FSM).BTT-FSM 能够准确地在尽可能早的时刻识别$\omega$正规性质的背离.但当前对参数化性质规约的预测监控器构造问题研究还比较少,而参数化性质规约的运行时验证本身存在诸多的特殊性.本文面向参数化 LTL 性质规约研究参数化预测监控器的构造问题.

对于参数化 LTL,其分析对象是在程序运行过程中动态产生的,而且不同运行涉及的对象数目和标识也可能完全不同,因此很难预先静态确定分析对象的值域,需要在程序运行过程中动态地确定所有变量的当前取值,即动态提取变量赋值集合,同时在适当时刻绑定和使用这些变量赋值.因此,面向参数化 LTL 的预测监控器构造需要从静态构造和动态运行两个方面来刻画.在预测监控器的静态结构中不考虑具体的变量赋值,而仅留下变量赋值的占位符,同时刻画抽象状态间的迁移关系;而在预测监控器动态运行过程中,反复进行变量赋值集合的提取和绑定操作,把抽象状态转变为具体状态.

基于上述分析,参数化 LTL 的运行时验证技术与非参数化性质运行时验证之间的差异主要表现在以下几个方面:

- 在参数化性质的运行时验证过程中,需要考虑变量赋值集合的提取和绑定操作,即变量赋值集合的获取以及实例化过程.经过这两个操作的一次完整迭代,将获得一个实例化后的被监控性质.从完全实例化完成后的状态开始,其验证过程对应于一般的 LTL 公式的运行时验证.
- 在参数化性质的运行时验证过程中,由于其变量赋值集合的提取和绑定操作是基于程序无穷执行轨迹中的当前状态进行的,因此,在程序执行过程中可能存在变量赋值集合的提取和绑定的多次迭代过程,形

© 中国科学院软件研究所 http://www.c-s-a.org.cn

成多个被完全实例化的被监控性质,从而使得参数化 LTL 的运行时验证过程实际上对应了多个一般 LTL 公式的运行时验证.同时,这多个一般 LTL 公式的运行时验证始于不同的状态而并发进行.

需要强调的是,为了与传统的命题线性时序逻辑(proposition linear temporal logic,简称PLTL)相区别,本文把提出的参数化LTL记为$P_A$LTL.

与以往的有关研究[1,6,7]相比,本文的主要贡献包括以下几个方面:

- 扩展free-$X$参数化LTL为$P_A$LTL.
- 提出面向$P_A$LTL性质规约的预测监控器和参数化Büchi自动机的概念,使得相应的验证器能够尽可能早地识别$P_A$LTL性质规约的满足情况.
- 提出基于参数化 Büchi 自动机的预测监控器构造及运行机制.

本文第 1 节主要介绍预测语义及预测监控器提出的背景,面向LTL的预测语义定义及预测监控器构造的一般过程.第 2 节介绍$P_A$LTL的语法、预测语义及相关问题.第 3 节介绍面向$P_A$LTL的预测监控器的定义、构造和运行.最后一节总结全文并对下一步的工作进行说明.

## 1 预测语义及预测监控器

在线监控而言,监控对象在理想情况下一般是一个永不终止的系统,也就是说,往往需要检查一个无穷执行轨迹的正确性.但明显地,在系统运行过程,在任意时刻看到的都将是系统无穷执行轨迹的一个有穷前缀.因此,我们需要一种裁决机制:一方面通过考虑有穷执行轨迹来裁定以该有穷执行轨迹为前缀的所有无穷执行轨迹是否满足指定的正确性性质;另一方面,相应的运行时验证器要尽可能早地发现软件运行过程中的异常.在在线监控中,检测程序运行时错误固然重要,但同时,是否能够及时发现错误从而为错误修复提供充足的时间和空间则更为关键.对于前者,文献[8–15]提出了针对多种不同形式规约的运行时验证技术,但相同的是,它们都在系统的有穷行为轨迹上定义了相应的规约逻辑的有穷轨迹语义.以LTL公式为例,假定公式$\varphi\in LTL$,$u=w_0\dots w_n$为有穷行为轨迹,则比较典型的有穷轨迹语义定义分为以下几种:

- 弱语义:如果在轨迹 $u$ 的末尾,坏的事情还没有发生,那么$\varphi$成立.
- 强语义:只有在轨迹 $u$ 的范围内,$\varphi$被计算为真,$\varphi$才成立.
- 扩展语义:把有穷序列$u=w_0\dots w_n$扩展为无穷序列$u^\omega=w_0\dots w_nw_n\dots w_n\dots$,则$u\models\varphi$当且仅当$u^\omega\models\varphi$.

**定义 1**. 对于逻辑 $L$,如果不存在公式$\varphi\in L$ 以及公式$\varphi$的一个模型 $M$,使得 $M\models\varphi$并且 $M\models\neg\varphi$,那么称逻辑 $L$ 是一致的.

不难发现,上述任意一种有穷轨迹语义定义都存在局限性,都能非常容易地找到一个实例公式来说明其有穷轨迹语义存在着不一致.例如,考虑公式$\varphi=Xp$和$\varphi'=X\neg p$,$u=a_0$是一个仅包含 1 个元素的有穷状态序列,其中,$a_0=\{p\}$.如果采用LTL公式的强语义有穷轨迹语义,我们可以得到$u\not\models\varphi$并且$u\not\models\varphi'$,因为当前状态后不存在后继状态来满足这两个公式.同样地,如果采用LTL公式的弱语义有穷轨迹语义,可以得到$u\models\varphi$并且$u\models\varphi'$,因为直到轨迹 $u$ 的末尾仍未出现使得$\varphi$和$\varphi'$为假的事件.对于测试和验证,特别是基于运行时验证结论进行行为调整和错误修复,在系统运行的任意时刻,准确地知道被监控性质是否确实为真、为假或者仅仅不确定是至关重要的.

运行时验证的一个显著特征是基于当前观察到的有穷轨迹来判定当前程序执行是否满足指定的性质.也就是说,仅仅基于当前执行的一个有穷前缀来裁定整个程序执行的正确性.因此,需要研究有穷前缀与无穷轨迹之间的相互关系.当前的许多面向有穷轨迹的运行时验证和推理系统都是基于文献[5,16,17]中的研究成果.其中提出了好/坏前缀、最小好/坏前缀以及 informative 前缀的概念,下面分别加以介绍:

**定义 2**. 一个有穷字$u\in\Sigma^*$被称为语言$L\subseteq\Sigma^\omega$的好前缀当且仅当对于每一个无穷字$w\in\Sigma^\omega$,$u.w\in L$.同样地,一个有穷字$u\in\Sigma^*$被称为语言$L\subseteq\Sigma^\omega$的坏前缀当且仅当对于每一个无穷字$w\in\Sigma^\omega$,$u.w\notin L$.如果$u\in\Sigma^*$是语言$L$的好(坏)前缀,并且不存在任何其他$u'\in\Sigma^*$,$u'$为$L$的好(坏)前缀,使得$u'$为$u$的真前缀,那么称$u\in\Sigma^*$为语言$L$的最小好(坏)前缀.

与好/坏前缀的概念相对应的是informative前缀.直观上讲,一个有穷字$u\in\Sigma^*$是语言$L$的informative前缀,是指有穷字$u$具体说明了为什么以$u$为前缀的所有无穷状态序列满足或违背指定的正确性质.也就是说,基于

© 中国科学院软件研究所 http://www.c-s-a.org.cn

informative前缀,存在一个推理或演算过程具体展示了被验证性质为真或为假的过程.如果任何以informative前缀$u$为前缀的无穷执行轨迹都满足指定的正确性质$\varphi$,那么$u$被称为性质$\varphi$的informative好前缀,否则被称为informative坏前缀.比如,对于公式$\varphi=Gp$,$u=\{\neg p\}$,那么$u$是公式$\varphi$的informative坏前缀.因为一旦出现$\neg p$,就具体说明了为什么$Gp$为假.而对于公式$\psi=G(p\vee(Xq\wedge X\neg q))$,$u$就不是公式$\psi$的informative坏前缀,因为如果一个序列希望展示公式$\psi$为假的过程,则该序列必须至少包含两个状态,并且在前一个状态处$\neg p$成立.但明显地,$u$为公式$\psi$的最小坏前缀.

综上所述,当前的运行时验证技术,根据其识别的前缀的类型,一般可以分为两大类,分别识别被验证性质的最小好/坏前缀和 informative 好/坏前缀.对于模型检验(model checking)技术来说,比较关心 informative 好/坏前缀,因为在模型检验中,一旦发现性质违背就会给出反例,而基于 informative 坏前缀的反例路径充分地说明了为什么被验证性质被违背;而对于运行时验证技术来说,它更关心在系统运行时尽可能早地发现异常,从而为程序执行状态调控提供时间和空间.可以看出,最小好/坏前缀是 informative 好/坏前缀的前缀.也就是说,最小好坏前缀的长度小于等于 informative 好/坏前缀.

由此可见,运行时验证器需要考虑的两个关键问题是语义的一致性和验证的预测性.基于同样的认识,文献[18]中提出了预测监控器的概念,同时提出预测监控器必须满足两个准则:公平性和预测性.

- 公平性:对于任意有穷执行轨迹$\pi$,如果存在一条无穷后继$\sigma$使得$\pi\sigma$满足(或不满足)被监控性质,该有穷轨迹$\pi$就不能被该监控器裁定为假(或真).
- 预测性:对于任意有穷执行轨迹$\pi$,一旦对于$\pi$的任意一条无穷后继$\sigma$使得$\pi\sigma$都满足(或都不满足)被监控性质,那么有穷执行轨迹$\pi$将立刻被该监控器裁定为真(或假).

直观上讲,公平性即为语义一致性,预测性即要求运行时验证器识别被监控性质的最小好/坏前缀,尽早地发现被监控性质成立或者不成立.要达到上述目标,就需要在运行时验证过程中不能仅仅考虑当前已观察到的有穷状态序列,而且应该前瞻未来,判定从监控器的当前状态出发,是否仅存在到达监控器接收状态的路径.明显地,对于基于自动机的运行时验证器来说,该过程对应于从监控器中任意一个状态出发的判空问题.

### 1.1 LTL的预测语义

本文提出基于参数化 LTL 性质规约的预测监控器构造技术,其直接基础为 LTL 的预测监控器构造技术.下面简述其基本思想.

假定$\Sigma$是字母表,$\Sigma^*$是$\Sigma$上的有穷字集合,$\Sigma^\omega$是$\Sigma$上的无穷字集合.

LTL的语法:假定$AP$为原子命题集合,有穷字母表$\Sigma=2^{AP}$,$a$表示$\Sigma$中的一个元素,则字母表$\Sigma$上的命题线性时序逻辑公式定义为

$$\varphi ::= \text{true} \mid a \mid \neg\varphi_1 \mid \varphi_1\ op\ \varphi_2 \mid \varphi_1 \cup \varphi_2 \mid \varphi_1\ R\ \varphi_2 \mid X\varphi.$$

这里,$op\in\{\wedge,\vee\}$表示二元布尔操作符.除了上述基本操作符以外,为了描述简洁,还可引入其他几个导出操作符,比如 $G$ 和 $F$.

LTL的语义:假定$\varphi\in$LTL,$i\in$N表示序列中的一个位置,$w=a_0a_1\ldots\in\Sigma^\omega$为无穷状态序列,那么LTL公式的语义被递归定义如下:

$$w,i\models\text{true}.$$
$$w,i\models p\in AP \text{ iff } p\in a_i.$$
$$w,i\models\neg\varphi \text{ iff } w,i\not\models\varphi.$$
$$w,i\models\varphi_1\ op\ \varphi_2 \text{ iff } (w,i\models\varphi_1\ op\ \ w,i\models\varphi_2).$$
$$w,i\models\varphi_1\cup\varphi_2 \text{ iff } \exists k\geq i:((w,k\models\varphi_2\wedge\forall l:(i\leq l<k\rightarrow w,l\models\varphi_1)).$$
$$w,i\models\varphi_1\ R\ \varphi_2 \text{ iff } \forall i>0:((w,i\not\models\varphi_2)\rightarrow\exists l:(0<l<i\wedge w,l\models\varphi_1)).$$
$$w,i\models X\varphi \text{ iff } w,i+1\models\varphi.$$

$w\vDash\varphi$当且仅当 $w,0\vDash\varphi$.

LTL的预测语义:假定$\varphi\in$LTL,$i\in$N表示序列中的第$i$个位置,$\pi=a_0a_1\ldots a_n\in\Sigma^*$为有穷状态序列,那么LTL公式的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

预测语义定义如下:

$$[\pi \models \varphi]_{\mathrm{LTL}} = \begin{cases} \text{true}, & \text{if } \forall \sigma \in \Sigma^{\varpi} : \pi\sigma \models \varphi \\ \text{false}, & \text{if } \forall \sigma \in \Sigma^{\varpi} : \pi\sigma \not\models \varphi \\ ?, & \text{otherwise} \end{cases}.$$

容易看出,LTL预测语义的定义满足上面提到的两个准则:公平性和预测性.由于对于任意前缀$\pi\in\Sigma^*$,假定$\sigma$和$\sigma'$是它的两个无穷后继序列,并且$\pi\sigma\models\varphi$和$\pi\sigma'\models\neg\varphi$同时成立,那么$[\pi\models\varphi]_{\mathrm{LTL}}$的语义被计算为?.一旦仅有使得性质满足或不满足的后继序列存在时,$[\pi\models\varphi]_{LTL}$的语义才被定义为相应的true和false.

### 1.2 LTL预测监控器

面向LTL的预测监控器构造[2−4]以Büchi自动机为基础.本文中,$\mathrm{P_A}$LTL公式的预测监控器构造扩展一般的Büchi自动机,形成相应的参数化Büchi自动机,并以此为基础构造面向$\mathrm{P_A}$LTL的预测监控器.下面分别简单介绍面向LTL的预测监控器构造过程.

对于LTL性质规约,其基于Büchi自动机的预测监控器构造过程是一个标准化的过程[2].对于每一个公式$\varphi\in$LTL,构造非确定Büchi自动机 $A^{\varphi}=(\Sigma,Q^{\varphi},Q_0^{\varphi},\delta^{\varphi},F^{\varphi})$ 和 $A^{\neg\varphi}=(\Sigma,Q^{\neg\varphi},Q_0^{\neg\varphi},\delta^{\neg\varphi},F^{\neg\varphi})$,使得$L(A^{\varphi})=L(\varphi)$以及$L(A^{\neg\varphi})=L(\neg\varphi)$.对于任意的$u\in\Sigma^*$,假定 $\delta(Q_0^{\varphi},u)=\{q_1,...,q_l\},\delta(Q_0^{\neg\varphi},u)=\{p_1,...,p_n\}$,于是有$[\pi\models\varphi]_{\mathrm{LTL}}\neq$false当且仅当$\exists q\in\{q_1,\ldots,q_l\}$使得$L(A^{\varphi}(q))\neq\varnothing$,$[\pi\models\varphi]_{LTL}\neq$true当且仅当$\exists p\in\{p_1,\ldots,p_n\}$使得$L(A^{\neg\varphi}(p))\neq\varnothing$.这里,$A^{\varphi}(q)$和$A^{\neg\varphi}(p)$分别为把非确定Büchi自动机$A^{\varphi}$和$A^{\neg\varphi}$的初始状态集合设置为$\{q\}$和$\{p\}$后对应的Büchi自动机.根据自动机$A^{\varphi}$和$A^{\neg\varphi}$,定义函数$M^{\varphi}:Q^{\varphi}\to B$以及函数$M^{\neg\varphi}:Q^{\neg\varphi}\to B$,这两个函数分别根据自动机$A^{\varphi}(q)$以及$A^{\neg\varphi}(p)$是否接受语言为空,赋予状态$q$,$p$相应的取值.容易知道Büchi自动机判空问题是可判定的,从而定义$S^{\varphi}=\{q\in Q^{\varphi}|M^{\varphi}(q)=\text{true}\}$,并且$S^{\neg\varphi}=\{q\in Q^{\neg\varphi}|M^{\neg\varphi}(q)=\text{true}\}$.利用$S^{\varphi}$和$S^{\neg\varphi}$定义两个非确定有穷自动机 $\widehat{A^{\varphi}}=(\Sigma,Q^{\varphi},Q_0^{\varphi},\delta^{\varphi},S^{\varphi})$ 和 $\widehat{A^{\neg\varphi}}=(\Sigma,Q^{\neg\varphi},Q_0^{\neg\varphi},\delta^{\neg\varphi},S^{\neg\varphi})$.明显地, $\widehat{A^{\varphi}}$ 和 $\widehat{A^{\neg\varphi}}$ 接受的有穷轨迹$u\in\Sigma^*$分别对应于$[\pi\models\varphi]_{LTL}\neq$false和$[\pi\models\varphi]_{LTL}\neq$true.

综上所述,如果$u\in\Sigma^*$为有穷状态序列,$\varphi\in LTL$为待验证的性质规约, $\widehat{A^{\varphi}}=(\Sigma,Q^{\varphi},Q_0^{\varphi},\delta^{\varphi},S^{\varphi})$ 和 $\widehat{A^{\neg\varphi}}=(\Sigma,Q^{\neg\varphi},Q_0^{\neg\varphi},\delta^{\neg\varphi},S^{\neg\varphi})$ 分别为如上所述的非确定有穷自动机,那么有如下命题成立:

$$[u\models\varphi]_{LTL}=\begin{cases}\text{true}, & \text{if } u\notin L(\widehat{A^{\neg\varphi}}) \\ \text{false}, & \text{if } u\notin L(\widehat{A^{\varphi}}) \\ ?, & \text{if } u\notin L(\widehat{A^{\neg\varphi}}) \text{ and } u\notin L(\widehat{A^{\varphi}})\end{cases}.$$

上面的命题给出了基于给定的有穷轨迹 $u$ 计算性质规约预测语义的一个简单过程:同时把 $u$ 输入自动机 $\widehat{A^{\varphi}}$ 和 $\widehat{A^{\neg\varphi}}$,根据各自的接受情况,给定$[\pi\models\varphi]_{LTL}$相应的取值,即给定$\varphi\in LTL$相应的预测语义.

## 2 参数化的 LTL

在实际应用中,一方面LTL规约描述的是面向特定对象的某些性质,也就是说,这些性质往往涉及某个或某些预先静态确定的程序对象,而在现实程序中,某些对象的作用域往往是伴随程序执行的不同而动态变化;另一方面,面向LTL的预测监控器仅仅关注基于事件系统的验证,也就是说,在每个状态最多只有 1 个命题实例成立,比如对于顺序程序的执行.而对于某些系统,在每个状态却可能有同一命题的多个实例成立,比如对于某些并发程序的执行.明显地,对于这些系统和性质的刻画,仅仅LTL规约已经不能满足需要,而是需要扩展LTL规约中基本单元——命题的概念,引入变量参数和量词,使得公式中的命题由给定维数的构造子以及相应数目的变量参数组成.这时,程序中的事件为实例化后的参数命题,而程序中的状态由一组实例化后的参数命题组成.在一个状态中,构造子的参数值来自某些固定的对象域中的值.从而在语法上,我们必须引入新的操作符,实现具体程序状态与参数化命题的连接.该操作符将由两部分组成.一部分为特殊参数化命题,其中的每个变量必须由相应的量词限定,称为存在谓词.当存在谓词作用在某个状态上时,它负责从当前状态上提取赋值集合,即存在谓词

© 中国科学院软件研究所 http://www.c-s-a.org.cn

实际上为一个从状态到变量赋值集合的映射.另一部分基于前一部分提取的赋值集合,初始化相应变量,同时使用这些变量赋值在后续状态序列上验证实例化后的公式是否成立,而不产生新的赋值.这两个部分中,前者负责发现验证性质“感兴趣”的事件,并从当前状态提取变量赋值集合,后者负责绑定赋值同时在后续状态序列上验证实例化后公式[1,6,7].

**定义 3(命题、参数化命题、完全实例化命题以及存在谓词)**. 假定$PN$为命题名集合,$p\in PN^{(n)}$表示一个$n\in$N维构造子,那么给定值域$D$以及变量集合$V$,则命题集合$P$、参数化命题集合$P_1$、完全实例化命题集合$P_2$以及存在谓词集合$P_3$分别为

$$P=\cup_{n\in\mathrm{N}}\cup_{p\in PN^{(n)}}\{p(v_1,...,v_n)\mid v_k\in D\cup V,1\le k\le n\},$$

$$P_1=\cup_{n\in\mathrm{N}}\cup_{p\in PN^{(n)}}\{p(v_1,...,v_n)\mid \exists k:1\le k\le n,v_k\in V\},$$

$$P_2=\cup_{n\in\mathrm{N}}\cup_{p\in PN^{(n)}}\{p(v_1,...,v_n)\mid \forall k:1\le k\le n,v_k\in D\},$$

$$P_3=\cup_{n\in\mathrm{N}}\cup_{p\in PN^{(n)}}\{Q_1x_1..Q_mx_m:p(u_1,...,u_n)\mid \forall i:x_i\in V,\exists j:u_j=x_i,Q_i\in\{\forall,\exists\}\}.$$

从上述定义可以看出,$P$=$P_1$∪$P_2$,参数化命题集合$P_1$中的每个元素至少包含 1 个变量参数,而完全实例化命题每个位置上的变量都被$D$中的一个元素实例化了,从而不存在变量参数.比如,$p(x,4)$为参数化命题,而$p(3,4)$为完全实例化命题.其中,$x$为变量.存在谓词是指一个参数化的命题,其中某些变量被量词修饰(存在或全称量词),并且被量词修饰的任何变量都必须在后边的参数化命题中出现.不失一般性,通常假定被量词修饰的变量名与参数化命题中的自由变量名不冲突.

为了处理$\mathrm{P_A}$LTL中的赋值集合提取和绑定操作,我们在$\mathrm{P_A}$LTL中引入一个特殊的操作符号>>,称为bridge操作符.该操作符的一般形式为$Q_1x_1\ldots Q_mx_m{:}p(u_1,\ldots,u_n)>>\psi$.Bridge操作符的左边为一个存在谓词,通过该存在谓词,从当前状态提取变量赋值集合$Val$.在bridge操作符的右边为另外一个时序公式,通过绑定$Val$成为实例化LTL公式.该操作符允许嵌套,虽然中间的某个子公式中可能存在自由变元,但最终的$\mathrm{P_A}$LTL公式必须为句子.也就是说,$\mathrm{P_A}$LTL公式中所有的参数化变量必须受量词修饰,从而保证在经过一系列赋值提取和绑定操作之后, $\mathrm{P_A}$LTL公式中的所有参数化命题都将被完全实例化.需要注意的是,上述参数化命题实例化的过程不是一次性完成的,而需要基于系统运行过程中的多个状态递进实现,从而形成某个完全实例化公式的赋值链.

### 2.1 语法及语义

**定义 4(参数化LTL的语法($\mathbf{P_A}$LTL))**. 定义在变量集合$V$和值域$D$上的参数化LTL公式集合$\mathrm{P_A}$LTL被递归定义如下:

$$\begin{aligned}
&P_ALTL=LTL(\varnothing).\\
&LTL(V'\subset V)=tt\mid ff\\
&\qquad\mid p(u_1,...,u_n)\in PN^{(n)},u_1,...,u_n\in V'\cup D\\
&\qquad\mid Q_1x_1..Q_nx_x:p(u_1,...,u_m)>>LTL(V'')\\
&\qquad\quad \forall i:x_i\in V,\exists j:x_i=u_j,Q_i\in\{\forall,\exists\},\\
&\qquad\quad V''=V'\cup\{x_1,...,x_m\},\forall k:u_k\in V''\cup D\\
&\qquad\mid LTL(V')\wedge LTL(V')\\
&\qquad\mid LTL(V')\vee LTL(V')\\
&\qquad\mid \neg LTL(V')\\
&\qquad\mid X\ LTL(V')\\
&\qquad\mid LTL(V')\bigcup LTL(V')\\
&\qquad\mid LTL(V')\ R\ LTL(V').
\end{aligned}$$

在上面的定义中,$V''\in V$表示LTL公式中出现的自由变量集合.不失一般性,要求自由出现变量名与量化变量名之间不冲突.在bridge操作符定义中,要求每个被量化变量必须为后继参数化命题中的某个变量参数,最终的$\mathrm{P_A}$LTL公式中不允许存在自由变量.因此,上述定义有$P_ALTL$=$LTL(\varnothing)$.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

**定义 5(赋值的绑定)**. 假定$\rho:U\to D$为一个赋值, $\hat{\rho}$ 为$\rho$在$P_A$LTL命题和公式$\varphi\in LTL(V')$上的推广,$U,V',V''\subset V$:

$$\hat{\rho}:\ LTL(V')\to LTL(V'').$$

$$\hat{\rho}(p(v_1,...,v_m))=p(u_1,...,u_m),$$

$$u_i=\begin{cases}\rho(v_i), & \text{if } v_i\in V \text{ and } \rho(v_i) \text{ defined}\\ v_i, & \text{otherwise}\end{cases}.$$

$$\hat{\rho}(tt)=tt,\ \hat{\rho}(ff)=ff.$$

$$\hat{\rho}(\neg\varphi)=\neg\hat{\rho}(\varphi).$$

$$\hat{\rho}(Q_1x_1...Q_mx_m:p(u_1,...,u_n)>>\psi)=Q_1x_1...Q_mx_m:\hat{\rho}(p(u_1,...,u_n))>>\hat{\rho}(\psi).$$

$$\hat{\rho}(\varphi\triangleright\psi)=\hat{\rho}(\varphi)\triangleright\hat{\rho}(\psi),\ \ \triangleright\in\{R,\bigcup,\wedge,\vee\}.$$

$$\hat{\rho}(X\varphi)=X\hat{\rho}(\varphi).$$

$\rho_\varnothing$表示空赋值,即不存在为任何变量的赋值.注意,在上述定义中,之所以存在对被bridge操作符的绑定操作,是因为bridge操作符可以嵌套且内层子公式中允许出现自由变量.

为了定义$P_A$LTL的语义,下面引入文献[1]中的几个相关定义.

**定义 6(可匹配参数化命题)**. 两个具有相同命题名的参数化命题是可匹配的,当且仅当存在对参数化命题中所有变量的赋值,使得经过绑定操作后,两个完全实例化命题完全相同,其定义如下:

$$\downarrow: P\times P\to B$$

$$p(u_1,...,u_n)\downarrow p(v_1,...,v_n)(u_i,v_i\in V\cup D,1\le i\le n):\leftrightarrow$$

$$\exists\sigma_1,\sigma_2:V\to D \text{ such that } \widehat{\sigma_1}(p(u_1,...,u_n))=\widehat{\sigma_2}(p(v_1,...,v_n)).$$

**定义 7(赋值的精化)**. 操作符◊表示赋值的精化,通过该操作符实现赋值的扩展.即利用一个部分赋值来完善另一个部分赋值,从而使更多的变量被赋值,其形式化定义如下:

$$\Diamond:(V\to D)\times(V\to D)\to(V\to D),$$

$$\sigma_2\Diamond\sigma_1(x)=\begin{cases}\sigma_1(x), & \text{if } x\in V \text{ defined in } \sigma_1\\ \sigma_2(x), & \text{otherwise}\end{cases}.$$

**定理 1(赋值的复合操作)**. 操作符•表示赋值的复合操作,假定$\sigma_1$和$\sigma_2$为赋值,那么有

$$\widehat{\sigma_1}\bullet\widehat{\sigma_2}\equiv\widehat{(\sigma_1\Diamond\sigma_2)}.$$

**定义 8(基于状态的单变量取值集合提取)**. 给定一个部分实例化的参数化命题$p(u_1,\ldots,u_n)\in P$以及状态$a\in 2^{P_2}$,那么变量$x\in\{u_1,\ldots,u_n\}$的所有取值集合为

$$vals:P\times 2^{P_2}\times V\to 2^D,$$

$$vals(p(u_1,...,u_n),a,x)=\{d\in D\mid\exists p(d_1,...,d_n)\in a:\widehat{\{x/d\}}(p(u_1,...,u_n))\downarrow p(d_1,...,d_n)\}.$$

$P_A$LTL公式与LTL性质规约最本质的区别在于$P_A$LTL公式中引入了bridge操作符.首先,该操作符左边的存在谓词基于当前状态提取所有可能的赋值集合.根据存在谓词前量词的组合关系,假定$\sigma$为赋值,那么最终的赋值集合形如$2^{2^\sigma}$.然后,把相应的赋值集合作用于 bridge 操作符右边的公式,得到实例化公式的析取范式.其中,合取部分对应于内层赋值集合,而析取部分对应于外层集合.同时,$P_A$LTL中的bridge操作符允许嵌套使用,最终的$P_A$LTL公式中不允许出现自由变量.但对于其中任何一个中间操作符,如果出现在另一个bridge操作符的作用范围中,则可能包含自由变量,而这些自由变量的取值由外层的bridge操作符负责提取.

**定义 9(基于状态的赋值集合提取)**. 该操作过程可以用函数$spec:(V\to D)\times 2^{P_2}\times LTL(V)\to 2^{2^{V\to D}}$表示.不失一般性,这里的赋值集合提取操作面向任意一个bridge操作符,其中,$\rho:V\to D$表示从外边传入的已经提取的部分赋值集合.如果当前bridge操作符为待验证$P_A$LTL公式的最外层bridge操作符,那么$\rho=\rho_\varnothing$为空赋值;否则,$\rho$为当前子公式$\varphi\in LTL(V)$中包含的自由变量集合$V$的部分赋值.其形式化定义如下:

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$\Theta = valid(vals_{x1} \oplus_1 (...(vals_{xk} \oplus_k \{\{\sigma_\varnothing\}\})...)), \text{ or } \{\{\sigma_\varnothing\}\} \text{ if } k=0,$$

$$vals_{xi} = vals(\hat{\rho}(p(u_1,...,u_m)), w[j], x_i),$$

$$\oplus_i : 2^D \times 2^{2^{V\to D}} \to 2^{2^{V\to D}},$$

$$\Omega \oplus_i \Theta' = \begin{cases} \cup_{\theta'\in\Theta'}\{\cup_{\sigma'\in\theta'}\{\{x_i/d\}\Diamond\sigma'\} \mid d\in\Omega\}, \text{ if } Q_i = \exists \\ \otimes_{d\in\Omega}\{\{\{\{x_i/d\}\Diamond\sigma' \mid \sigma'\in\theta'\} \mid \theta'\in\Theta'\}\}, \text{ if } Q_i = \forall \end{cases},$$

$$\otimes\{\Omega_1,...,\Omega_n\} = \Omega_1\Delta...\Delta\Omega_n,$$

$$S\Delta T = \{s\cup t \mid s\in S, t\in T\}.$$

$$valid(\Theta) = \{\{\beta \mid \beta\in\theta, \widehat{(\beta\Diamond\sigma)}(p(d_1,...,d_m))\in w[j]\} \mid \theta\in\Theta\}.$$

由于赋值提取过程实际上仅仅与 bridge 操作符左边的存在谓词有关,从而 spec 函数可以简写为

$$\Theta = spec(\rho, w[j], Q_1x_1...Q_nx_n : p(u_1,...,u_m)).$$

例 1:我们通过一个实例来展示变量赋值提取和绑定的过程.假定

$$\varphi = \forall x\forall y : p(x,y)\to q(x,y),$$

$$w = \{p(1,1), p(1,2), p(2,1), p(2,3), q(1,1), q(1,2), q(2,1), q(2,3)\}...$$

这里,$\varphi$为待验证的$P_A$LTL性质,$w$为相应地无穷执行轨迹.公式$\varphi$左边的存在谓词为$\forall x\forall y{:}p(x,y)$,它负责从当前状态提取变量赋值集合.根据定义 9 中赋值集合提取操作,具体操作过程是,首先依次提取各量化变量取值集合:$vals_y=vals(p(x,y),w[1],y)=\{1,2,3\}$,$vals_x=vals(p(x,y),w[1],x)=\{1,2\}$,然后根据相应变量前的量词类型从右向左依次计算如下:

- 由于变量$y$前的量词类型为$\forall$,故$vals_y\oplus_2\{\{\sigma_\varnothing\}\}=\{\{\{y/1\},\{y/2\},\{y/3\}\}\}$;
- 变量 $x$ 前量词为$\forall$,故

$$valsx\oplus_1\{\{\{y/1\},\{y/2\},\{y/3\}\}\}=\{\{\{x/1,y/1\},\{x/1,y/2\},\{x/1,y/3\},\{x/2,y/1\}.\{x/2,y/2\},\{x/2,y/3\}\}\}.$$

- 从而有,

$$valid(\{\{\{x/1,y/1\},\{x/1,y/2\},\{x/1,y/3\},\{x/2,y/1\}.\{x/2,y/2\},\{x/2,y/3\}\}\})=\{\{\{x/1,y/1\},\{x/1,y/2\},\{x/2,y/1\},\{x/2,y/3\}\}\}.$$

最终的赋值集合为 $\Theta=\{\{\{x/1,y/1\},\{x/1,y/2\},\{x/2,y/1\},\{x/2,y/3\}\}\}$.

从而,$w\models\varphi\Leftrightarrow\vee_{\theta\in\Theta}\wedge_{\beta\in\theta}(w,\beta)\models q(x,y)$.通过绑定操作,参数化命题$q(x,y)$被实例化为$q(1,1)\wedge q(1,2)\wedge q(2,1)\wedge q(2,3)$,由于$q(1,1),q(1,2),q(2,1),q(2,3)$都包含在状态$w[1]$中,从而有$w\models\varphi$.

$P_A$LTL标准语义计算与LTL标准语义不同,因为其语义不但与当前状态有关,而且需要考虑由前面计算累计的赋值.根据上述$P_A$LTL公式的变量赋值提取操作和绑定操作,可定义$P_A$LTL标准语义如下:

**定义 10($P_A$LTL的标准语义,即无穷轨迹语义)**. 给定无穷非空路径 $w=w[0]w[1]...\in(2^{P_2})^w$,$j$为路径$w$中的第$j$个位置,累积赋值$\sigma{:}V\to D$以及公式$\varphi\in LTL(V)$,那么$P_A$LTL公式$\varphi$的标准语义在公式$\varphi$的结构上被递归定义如下:

$$(w,j,\sigma)\models tt,\ (w,j,\sigma)\not\models ff$$

$$\models p(u_1,...,u_n) \text{ iff } \hat{\sigma}(p(u_1,...,u_n))\in w[j]$$

$$\models\neg\varphi \text{ iff } (w,j.\sigma)\not\models\varphi$$

$$\models\varphi\wedge\psi \text{ iff } (w,j,\sigma)\models\varphi\wedge(w,j,\sigma)\models\psi$$

$$\models\varphi\vee\psi \text{ iff } (w,j,\sigma)\models\varphi\vee(w,j,\sigma)\models\psi$$

$$\models\varphi\cup\psi \text{ iff } \exists k(j\le k)((w,k,\sigma)\models\psi\wedge\forall l(j\le l<k)\to(w,l,\sigma)\models\varphi)$$

$$\models\varphi R\psi \text{ iff } \forall i>0:((w,i,\sigma\not\models\varphi)\to\exists l:(0<l<i\wedge(w,l,\sigma)\models\psi))$$

$$\models X\varphi \text{ iff } (w,j+1,\sigma)\models\varphi.$$

$$(w,j,\sigma)\models Q_1x_1...Q_kx_k : p(u_1,...,u_n) >> \psi, \text{ iff } \exists\theta\in\Theta \text{ such that } \forall\beta\in\theta:(w,j,\beta\Diamond\sigma)\models\psi$$

$$\equiv\vee_{\theta\in\Theta}\wedge_{\beta\in\theta}(w,j.\beta\Diamond\sigma)\models\psi,$$

$$\Theta = spec(\sigma, w[j], Q_1x_1...Q_nx_n : p(u_1,...,u_n)).$$

当$j$=0 时,$(w,j,\sigma)\models\varphi$可简写为$(w,\sigma)\models\varphi$.

**命题 1**. 对于所有的非空无穷路径 $w\in(2^{P_2})^\varpi$, $\varphi\in LTL(V)$以及赋值$\sigma{:}V\to D$,有下面的式子成立:

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$(w,\sigma)\models\varphi \Leftrightarrow (w,\sigma_{\varnothing})\models\hat{\sigma}(\varphi).$$

**定义 11($P_A$LTL预测语义)**. 假定$\varphi\in P_A$LTL,$\pi=a_0a_1\ldots a_n\in\Sigma^*$为有穷状态序列,$\sigma$为一个变量赋值,那么$P_A$LTL公式$\varphi$的预测语义定义如下:

$$[(\pi,\sigma)\models\varphi]_{P_A LTL}=\begin{cases}\text{true}, & \text{if } \forall\beta\in\Sigma^{\varpi}:(\pi\beta,\sigma)\models\varphi\\ \text{false}, & \text{if } \forall\beta\in\Sigma^{\varpi}:(\pi\beta,\sigma)\not\models\varphi.\\ ?, & \text{otherwise}\end{cases}$$

## 3 面向$P_A$LTL的参数化预测监控器

第 1.2 节介绍的面向LTL的预测监控器构造过程为$P_A$LTL预测监控器构造提供了有益借鉴,但同时, $P_A$LTL预测监控器又具有自身的特点,主要体现在面向$P_A$LTL的运行时验证需要基于程序的具体运行在适当时刻基于当前状态进行变量赋值的提取和绑定操作,因此变量赋值无法在系统实际运行前静态确定,而是在运行过程中动态进行变量赋值集合的提取和绑定操作.由此可见,面向$P_A$LTL的预测监控器无论在构造还是运行阶段都有不同于LTL预测监控器的方面,而且这种差别是本质的.对于面向$P_A$LTL的预测监控器构造,我们引入参数化Büchi自动机的概念,从$P_A$LTL公式向参数化Büchi自动机的转化以参数化的交错(alternating)Büchi自动机为中间环节.对于$P_A$LTL预测监控器的运行,基于当前状态累计的变量赋值集合,如果迁移边上标注为存在谓词(包含量词及被量化变量的参数化命题),那么将进一步基于当前状态提取新的变量赋值集合$\Theta$,形如$2^{2^{V\to D}}$,然后确定迁移边是否使能.在迁移发生的情况下,把相应的赋值集合传递到下一状态;如果迁移边上标注为不包含量词的参数化命题,那么将仅基于当前状态上的赋值集合确定迁移的使能情况.同时,在迁移发生的情况下,把相应的赋值集合传递到下一状态.$P_A$LTL语法保证,如果$P_A$LTL的预测监控器的某条迁移边标注为不包含量词的参数化命题,那么该参数化命题中包含的变量取值已经从前面的赋值提取操作中获取.

给定$\varphi\in P_A$LTL,本文介绍的预测监控器的构造过程如下:

(1) 对于$P_A$LTL公式$\varphi$,计算其否定形式$\neg\varphi$;

(2) 基于公式$\varphi$和$\neg\varphi$,构造相应的参数化交错 Büchi 自动机 $A_a^{\varphi}$ 和 $A_a^{\neg\varphi}$;

(3) 把参数化交错 Büchi 自动机 $A_a^{\varphi}$ 和 $A_a^{\neg\varphi}$ 分别转化为非确定参数化 Büchi 自动机 $A_n^{\varphi}$ 和 $A_n^{\neg\varphi}$;

(4) 基于参数化 Büchi 自动机 $A_n^{\varphi}$ 和 $A_n^{\neg\varphi}$,假定 $q$ 为 $A_n^{\varphi}$ (或 $A_n^{\neg\varphi}$)中任意一个状态,判定是否 $L(A_n^{\varphi}(q))=\varnothing$ (或 $L(A_n^{\neg\varphi}(q))=\varnothing$);

(5) 定义 $F^{\varphi}=\{q\in Q^{\varphi}\mid L(A_n^{\varphi}(q))\neq\varnothing\}$ 和 $F^{\varphi}=\{q\in Q^{\neg\varphi}\mid L(A_n^{\neg\varphi}(q))\neq\varnothing\}$,从而基于参数化Büchi自动机 $A_n^{\varphi}$ 和 $A_n^{\neg\varphi}$,构造相应的参数化有穷状态机 $\widehat{A^{\varphi}}$ 和 $\widehat{A^{\neg\varphi}}$,使得$F^{\varphi}$和$F^{\neg\varphi}$分别为有穷状态机 $\widehat{A^{\varphi}}$ 和 $\widehat{A^{\neg\varphi}}$ 的接收状态集合.

从而明显地有 $[(\pi,\sigma)\models\varphi]_{P_A LTL}=\begin{cases}\text{true}, & \text{if } \pi\notin L(\widehat{A^{\varphi}})\\ \text{false}, & \text{if } \pi\notin L(\widehat{A^{\neg\varphi}})\\ ?, & \text{if } \pi\notin L(\widehat{A^{\varphi}}) \text{ and } \pi\notin L(\widehat{A^{\neg\varphi}})\end{cases}$.

在一般的一阶逻辑框架中,当取非操作作用在量词上时,通常是取非操作穿过这些量词,同时翻转量词,即∀变∃,∃变∀.在时序公式中,时序操作符将出现类似的行为,即$F/G$,$R/U$互换.$P_A$LTL公式中除了包含上述操作符以外,还引入了bridge操作符.bridge操作符左边的存在谓词用于枚举当前状态处所有可能的变量赋值,这些变量赋值实际上是在当前状态处变量所能代表的对象集合.因此,bridge操作符左边不能为否定形式的参数化命题,否定只能出现在bridge操作符的右边.

**定义 12($P_A$LTL公式的肯定形式)**. 给定一个公式$\varphi\in P_A$LTL,通过反复应用如下的重写规格把否定符号推到命题符号的前面,可以得到$P_A$LTL公式的肯定形式$\varphi^+$.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$\neg\neg\varphi \Rightarrow \varphi.$$
$$\neg(\varphi \vee \psi) \Rightarrow \neg\varphi \wedge \neg\psi.$$
$$\neg(\varphi \wedge \psi) \Rightarrow \neg\varphi \vee \neg\psi.$$
$$\neg(F\varphi) \Rightarrow G\neg\varphi.$$
$$\neg(G\varphi) \Rightarrow F\neg\varphi.$$
$$\neg(\varphi \bigcup \psi) \Rightarrow \neg\varphi R \neg\psi.$$
$$\neg(\varphi\ R\ \psi) \Rightarrow \neg\varphi \bigcup \neg\psi.$$
$$\neg Q_1 x_x ... Q_n x_n : p(u_1,...,u_n) >> \psi \Rightarrow \overline{Q_1} x_1 ... \overline{Q_n} x_n : p(u_1,...,u_n) >> \neg\psi,$$
$$\overline{Q_i} = \begin{cases} \exists, & \text{iff } Q_i = \forall \\ \forall, & \text{iff } Q_i = \exists \end{cases}.$$

**定理 2($P_A$LTL公式及肯定形式之间的等价性)**. 假定$w$为任意一个无穷执行轨迹,$\beta$为当前变量赋值,$\varphi$为任意一个$P_A$LTL公式,那么有$(w,\beta)\models\neg\varphi \Leftrightarrow (w,\beta)\models(\neg\varphi)^+$.

证明略.

由于$P_A$LTL公式中的基本单元为参数化命题,参数化命题中的变量对应于系统中某些特定的对象或数据结构,而这些对象和数据结构的取值范围往往是无穷的,因此,静态考虑参数化命题中的变量取值,从而实现参数化命题的实例化面对的空间往往是无穷的.也就是说,像基于Büchi自动机的LTL预测监控器那样显式、静态地确定其所有状态是不可能的.因此,在面向$P_A$LTL的预测监控器的构造过程中必须通过抽象机制,用抽象状态来代表其所有实例化后的具体状态.而在系统运行过程中,结合具体的程序执行状态,实现监控器抽象状态的具体化.本文将从抽象静态迁移系统构造和动态运行两方面来刻画面向$P_A$LTL的预测监控器.

### 3.1 构 造

从$P_A$LTL公式向参数化非确定Büchi自动机的转化是一个比较复杂的过程,本文以参数化交错Büchi自动机为变换的中间环节,从而有效实现逻辑与组合数学的分离[19],使得变换过程直观、易于理解.

#### 3.1.1 从$P_A$LTL公式到参数化交错Büchi自动机

如上所述,对应于$P_A$LTL的参数化交错Büchi自动机与一般的对应于LTL公式的交错Büchi自动机不同,前者的每个状态都为抽象状态,由抽象的状态标识和表示变量赋值的符号组成.针对不同的系统执行,各个抽象状态将绑定不同的变量赋值,从而对应于不同的具体状态.前者的状态迁移边上不仅可以标注参数化命题(包括完全实例化和部分实例化命题),还可以标注存在谓词.如果迁移边上的标注为存在量词,那么它将负责动态地从当前状态上提取变量赋值集合,同时把赋值集合传递给后继状态.如果迁移边上标注的为部分实例化的参数化命题,那么$P_A$LTL语法将保证其中的变量赋值已被提取,从而基于该变量赋值判定迁移边的使能情况.因此,文献[1,6]指出,面向free-$X$参数化交错Büchi自动机可以分为静态和动态两部分,其中,静态部分只与公式的语法结构相关,刻画不同抽象状态之间的前驱后继关系;而动态部分则包括基于迁移边提取变量赋值,判定迁移使能情况,同时把变量赋值传递到其后继状态.下面将具体描述面向$P_A$LTL的参数化交错Büchi自动机的定义及构造过程.

**定义 13($P_A$LTL公式的闭包)**. 假定$P_A$LTL公式$\varphi$为肯定形式,$cl'(\varphi)$:$P_A$LTL$\rightarrow 2^{LTL(V)}$定义如下:

$$\varphi \in cl'(\varphi).$$
$$tt, ff \in cl'(\varphi).$$
$$\varphi_1 \vee \varphi_2 \in cl'(\varphi) \rightarrow \varphi_1, \varphi_2 \in cl'(\varphi).$$
$$\varphi_1 \wedge \varphi_2 \in cl'(\varphi) \rightarrow \varphi_1, \varphi_2 \in cl'(\varphi).$$
$$\varphi_1\ R\ \varphi_2 \in cl'(\varphi) \rightarrow \varphi_1, \varphi_2 \in cl'(\varphi).$$
$$\varphi_1 \bigcup \varphi_2 \in cl'(\varphi) \rightarrow \varphi_1, \varphi_2 \in cl'(\varphi).$$
$$X\varphi_1 \in cl'(\varphi) \rightarrow \varphi_1 \in cl'(\varphi).$$
$$Q_1 x_1 ... Q_n x_n : p(u_1,...,u_m) >> \psi \in cl'(\varphi) \rightarrow \psi \in cl'(\varphi).$$

© 中国科学院软件研究所 http://www.c-s-a.org.cn

**定义 14($P_A$LTL公式$\varphi$对应的参数化交错Büchi自动机$A_p(\varphi)$)**. $P_A$LTL公式$\varphi$对应的参数化Büchi自动机$A_p(\varphi)=\langle 2^{P_2},Q,q_0,\delta,F\rangle$是一个五元组,$S=cl'(\varphi)$:

- $2^{P_2}$为字母表.
- $Q=S\times[V\to D]$.
- $q_0=(\varphi,\sigma_\varnothing)\in Q$为唯一初始状态.
- $F=(\{tt\}\cup\{q\in S|q=\varphi R\psi\})\times[V\to D]$.
- $\delta:Q\times\Sigma\to B^+(S)$为迁移函数.

其中,迁移函数$\delta$定义如下(在文献[1]中定义的基础上增加了对 *Next*($X$)操作符的定义):

$$\delta((tt,\beta),a)=\{\varnothing\}.$$

$$\delta((ff,\beta),a)=\varnothing.$$

$$\delta((p(u_1,...,u_n),\beta),a)=\begin{cases}\{\{(tt,\beta)\}\}, & \hat{\beta}(p(u_1,...,u_n))\in a\\ \{\{(ff,\beta)\}\}, & \text{otherwise}\end{cases}.$$

$$\delta((\neg p(u_1,...,u_n),\beta),a)=\begin{cases}\{\{(tt,\beta)\}\}, & \hat{\beta}(p(u_1,...,u_n))\notin a\\ \{\{(ff,\beta)\}\}, & \text{otherwise}\end{cases}.$$

$$\delta((\varphi\vee\psi,\beta),a)=\delta((\varphi,\beta),a)\cup\delta((\psi,\beta),a).$$

$$\delta((\varphi\wedge\psi,\beta),a)=\delta((\varphi,\beta),a)\Delta\delta((\psi,\beta),a).$$

$$\delta((\varphi\cup\psi,\beta),a)=\delta((\psi,\beta),a)\cup(\{\{(\varphi\cup\psi,\beta)\}\}\Delta\delta((\varphi,\beta),a)).$$

$$\delta((\varphi R\psi,\beta),a)=(\delta((\varphi,\beta),a)\Delta\delta((\psi,\beta),a))\cup(\{\{(\varphi R\psi,\beta)\}\}\Delta\delta((\varphi,\beta),a)).$$

$$\delta((X\varphi,\beta),a)=\{\{(\varphi,\beta)\}\}.$$

$$\delta((Q_1x_1...Q_nx_n:p(u_1,...,u_n)>>\psi,\beta),a)=\begin{cases}\{\varnothing\}, & Q_1=\forall \text{ and } \Theta=\varnothing\\ \cup_{\theta\in\Theta}\{\otimes(\delta((\psi,\sigma\Diamond\beta),a)\mid\sigma\in\theta\}\}, & \end{cases},$$

$$\Theta=spec(\beta,a,Q_1x_1...Q_nx_n:p(u_1,...,u_n)).$$

由上述定义可以看出,$A_p(\varphi)$中的每个状态上的标识为$\varphi$的一个子公式.$A_p(\varphi)$的终止状态$F$与具体的绑定无关,因为一个状态是否为接受状态仅依赖于该状态上标注的子公式.给定无穷执行轨迹$w\in(2^{P_2})^\varpi$,其对应的无穷运行树为$r$,那么该无穷执行轨迹最终满足$P_A$LTL性质$\varphi$,当且仅当$r$的所有有穷分支都终止为$tt$,而所有无穷分支都无穷多次穿过某个终止状态.

对于迁移函数$\delta$的定义,由于$\Sigma=2^{P_2}$,故每个状态的后继状态可能为无穷多个.因此,基于理想的显式的$\delta$定义来构造相应的交错Büchi自动机是不现实的.下面给出$\delta$函数的一种可操作的定义,在这种定义中,将暂时不考虑输入符号,而仅考虑抽象状态间的前驱和后继关系.对于$A_p(\varphi)$中的每个状态,都把当前状态上标识的子公式分解为两个部分:第1部分是为了公式$\varphi$最终被满足,必须在当前状态处被满足的那部分子公式;第2部分为需要在未来某个时刻被满足的公式.如果第1部分被证明是不成立的,就没有必要继续计算下去;否则,需要继续计算第2部分.如果第2部分为空,那么整个公式最终为真当且仅当在当前状态处第1部分被满足.令$S=cl'(\varphi)$,具体的分解过程如下:

$$split:S\to 2^{S\times S}.$$

$$split(tt)=\{\{(t,\text{null})\}\}.$$

$$split(ff)=\{\{(ff,\text{null})\}\}.$$

$$split(p(u_1,...,u_n))=\{\{(p(u_1,...,u_n),\text{null})\}\}.$$

$$split(\neg p(u_1,...,u_n))=\{\{(\neg p(u_1,...,u_n),\text{null})\}\}.$$

$$split(\varphi\vee\psi)=split(\varphi)\cup split(\psi).$$

$$split(\varphi\wedge\psi)=split(\varphi)\Delta split(\psi).$$

$$split(\varphi\cup\psi)=split(\psi)\cup combine(split(\varphi),\varphi\cup\psi).$$

$$split(\varphi R\psi)=(split(\varphi)\Delta split(\psi))\cup combine(split(\psi),\varphi R\psi).$$

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$split(X\varphi)=\{\{(tt,\varphi)\}\}$,

$combine:2^{2^{S\times S}}\times S\to 2^{2^{S\times S}}$,

$combine(s,\varphi)=s\Delta\{\{(tt,\varphi)\}\}$.

$split(Q_1x_1...Q_nx_n:p(u_1,...,u_n)>>\psi)=\{\{(Q_1x_1...Q_nx_n:p(u_1,...,u_n)>>now_\varphi,next_\varphi)\mid(now_\varphi,next_\varphi)\in qs\}\mid qs\in split(\psi)$.

通过上述分解过程可以发现,公式$\varphi$的每个子公式都被分解成为一个析取范式的形式,其中,外层集合的各个元素之间表示析取关系,对应于参数化交错 Büchi 自动机的析取迁移;内层集合各元素间为合取关系,对应于参数化交错 Büchi 自动机的合取迁移.对于内层集合中的元素 $e$,其一般形式为 $S\times S$,前者为必须在当前状态被满足的部分子公式,后者为必须在未来某个状态被满足的部分子公式.对于参数化命题,$tt$,$ff$,其后继状态即第 2 部分都为 null,这就意味这类公式只需要在当前状态成立即可.

假定存在公式$\varphi_1,\varphi_i,\varphi_j\in cl'(\varphi)$并且$split(\varphi_1)=\{...,\{...,(\varphi_i,\varphi_j),...\},...\}$,那么在相应的参数化交错Büchi自动机中,$(\varphi_j,[V\to D])$将为$(\varphi_1,[V\to D])$的后继,同时,$\varphi_i$为两者之间迁移边上的标识.通过上述过程获得的状态迁移系统称为参数化较交错Büchi自动机的静态结构.

**定义 15**. $P_A$LTL公式$\varphi$对应的参数化交错Büchi自动机的静态结构 $A_a^{static}(\varphi)=\langle Q,q_0,\delta,F_a\rangle$ 是一个四元组,$S=cl'(\varphi)$:

- $Q=S$ 为抽象状态集合.
- $q_0=\varphi\in Q$为唯一初始状态.
- $F_a=\{tt\}\cup\{q\in Q\mid q=\varphi\ R\ \psi\}$).
- $\delta:Q\to B^+(Q\times Q)$为迁移函数.

其中,$\delta(Q)=split(Q)$.

例 2:假定需要验证某类无线接收设备满足如下性质:对于任意处于工作状态的某类无线接收设备 $x$,直到它向外发出停止工作信号 $z$ 之前,如果存在信号 $y$ 从其他无线发射设备发出,那么在下一时刻,$x$ 一定会接收到 $y$.

假定 $p(x)$表示接收设备 $x$ 处于工作状态,$q(y)$表示信号 $y$ 从其他设备发出,$s(x,z)$表示设备 $x$ 发出停止工作信号$z$,$r(x,y)$表示设备$x$接受到信号$y$,那么,相应的$P_A$LTL公式为$\forall x:p(x)\to((\exists y:q(y)\to Xr(x,y))\cup\exists z:s(x,z))$,与其对应的交错 Büchi 自动机的抽象状态集合及迁移关系如图 1 所示.

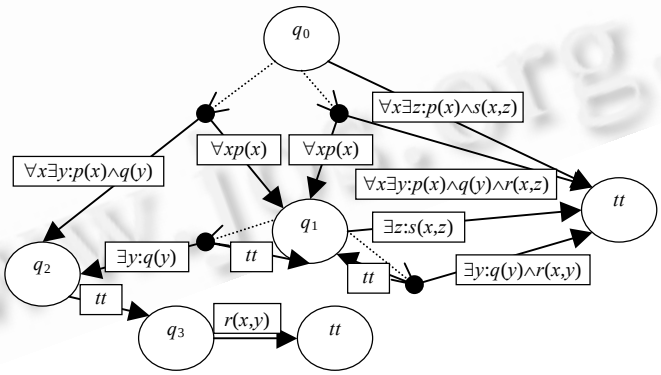


Fig.1 One example of parameterized alternating Büchi automata

图 1 参数化交错 Büchi 自动机示例

在图 1 中,状态$q_0,q_1,q_2$和$q_3$上标注的子公式依次为$\forall x:p(x)\to((\exists y:q(y)\to Xr(x,y))\cup\exists z:s(x,z))$,$(\exists y:q(y)\to Xr(x,y))\cup\exists z:s(x,z)$,$Xr(x,y)$以及$r(x,y)$.每条迁移边上标注的公式为上述 split 函数的第 1 部分,即需要在当前状态的满足的子公式.迁移边指向的状态为 split 函数的第 2 部分.图中的圆点表示迁移边的合取关系.比如,

$split(q_0)=\{\{(\forall x\exists y:p(x)\wedge q(y),q_2),(\forall x\ p(x),q_1)\},\{(\forall x\ p(x),q_1),(\forall x\exists y:p(x)\wedge q(y)\wedge r(x,y),tt)\},\{(\forall x\exists z:p(x)\wedge s(x,z),tt)\}\}$.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

3.1.2 从交错 Büchi 自动机到非确定 Büchi 自动机

预测监控器的一个本质特征是其每一个状态都标注符号 $\tau\in\{\text{true,false,?}\}$,用来表示从该状态出发是否还可以接受无穷状态序列,从而一旦预测监控器伴随系统有穷运行轨迹 $\pi$ 到达该状态,预测监控器就能尽早地判定对于任意无穷序列 $\sigma\in\Sigma^{\omega}$,是否 $\pi\sigma$ 满足被验证性质.Büchi自动机非空的判定问题不但是可以判定的,而且其时间复杂度为线性时间.因此,当前预测监控器的构造往往以Büchi自动机为基础.面向 $\mathrm{P_A}$LTL的预测监控器同样如此,对应于第 3.1.1 节参数化交错Büchi自动机静态结构的概念,我们将定义参数化Büchi自动机的静态结构,它同样暂不考虑输入符号,而仅考虑Büchi自动机的抽象状态及状态间的迁移关系.

基于参数化 Büchi 自动机的静态结构可进行 Büchi 自动机非空的判定.由于参数化 Büchi 自动机的静态状态代表了其运行时的任何具体状态,因此,从抽象状态出发的判空过程代表了对预测监控器所有可能执行情况的判定.也就是说,假定参数化 Büchi 自动机的静态结构为 $A_n^p$ ,$a$ 为任意抽象状态,那么一旦判定 $A_n^p(a)$ 为空,就说明从抽象状态 $a$ 的任何具体状态出发的任何具体执行都不会满足被监控的参数化性质.从而可以看出,基于预先构造的参数化Büchi自动机的静态结构而不考虑具体的变量绑定,对于 $\mathrm{P_A}$LTL预测监控器的构造不但是可行的,而且是十分有效的方式.

下面具体介绍参数化 Büchi 自动机静态结构的构造过程.

由文献[19]可知,任意一个交错Büchi自动机 $A=(\Sigma,S,s_0,\rho,F)$ 都存在一个非确定Büchi自动机 $A_n=(\Sigma,S_n,S_0,\rho_n,F_n)$,使得 $L_\omega(A)=L_\omega(A_n)$.$A_n$ 模拟 $A$ 的执行,当读入一个输入符号时,它猜测 $A$ 的运行树的所有下一级状态.同时,非确定Büchi自动机为了保证 $A$ 的运行树的每一个无穷分支都无穷次地越过某个接受状态,需要记录接受状态是否出现的信息.最终,非确定Büchi自动机 $A_n$ 把 $A$ 的运行时每层节点分为两个不同的集合:最近命中接受状态的分支和最近没有命中接受状态的分支.从参数化交错Büchi自动机静态结构构造参数化非确定Büchi自动机静态结构的过程同样如此.

**定义 16**. 给定参数化交错 Büchi 自动机的静态结构 $A_a^{static}(\varphi)=\langle Q,q_0,\delta,F_a\rangle$ ,与其相对应的参数化非确定 Büchi 自动机的静态结构 $A_n^{static}(\varphi)=\langle S_n,S_0,\rho_n,F_n\rangle$ 是一个四元组:

- $S_n=2^Q\times 2^Q$.
- $S_0=\{(\{q_0\},\varnothing)\}$.
- $F_n=\{\varnothing\}\times 2^Q$.
- 迁移函数 $\rho_n$ 定义如下:

如果 $U\neq\varnothing$,那么 $\rho_n((U,V))=\{(U',V')|$存在集合 $X,Y\subseteq 2^Q$,使得 $X\in\otimes_{t\in U}\{\{\{\psi_i|(\varphi_i,\psi_i)\in qs\}|qs\in\delta(t)\}\}$,

$$Y\in\otimes_{t\in V}\{\{\{\psi_i|(\varphi_i,\psi_i)\in qs\}|qs\in\delta(t)\}\},U'=X-F_a,V'=Y\cup(X\cap F_a)\};$$

如果 $U=\varnothing$,那么 $\rho_n((\varnothing,V))=\{(U',V')|$存在集合 $Y\subseteq 2^Q$,使得 $Y\in\otimes_{t\in V}\{\{\{\psi_i|(\varphi_i,\psi_i)\in qs\}|qs\in\delta(t)\}\}$,

$$U'=Y-F_a,V'=Y\cap F_a)\}.$$

相应地,对应于集合 $X,Y$,定义:

$$\overline{X}\in\otimes_{t\in U}\{\{\{(\langle t,\varphi_i\rangle,\psi_i)\,|\,(\varphi_i,\psi_i)\in qs\}\,|\,qs\in\delta(t)\}\},$$

$$\overline{Y}\in\otimes_{t\in V}\{\{\{(\langle t,\varphi_i\rangle,\psi_i)\,|\,(\varphi_i,\psi_i)\in qs\}\,|\,qs\in\delta(t)\}\}.$$

在运行时,如果预测监控器从 $(U,V)$ 迁移到 $(U',V')$,那么存在 $\overline{x}\in\overline{X},\overline{y}\in\overline{Y}$ ,对于任意的 $\varphi_i\in(U',V')$,$(U',V')\in S_n$ 都存在 $(\langle t_i,\varphi_i\rangle,\psi_i)\in\overline{x}\cup\overline{y}$ 与之相对应,从而定义 $\overline{x}\cup\overline{y}$ 为从状态 $(U,V)$ 迁移到状态 $(U',V')$ 的迁移约束.

基于上述参数化Büchi自动机的静态结构 $A_n$,对于任意状态 $S$,判定 $L(A_n(S))$ 是否为空,从而赋予该状态相应的标示true,false或?.同样的过程应用于被监控 $\mathrm{P_A}$LTL性质 $\varphi$ 的否定 $\neg\varphi$.

### 3.2 运 行

通过上述构造过程形成了面向 $\mathrm{P_A}$LTL公式的预测监控器的静态结构,也就是说,没有考虑监控器的输入问题.明显地,预测监控器的真正运行需要把输入和预测监控器静态结构的状态迁移结合起来.下面将讨论面向 $\mathrm{P_A}$LTL的预测监控器的运行过程,在预测监控器的运行过程中,将通过赋值集合提取和绑定操作,以on-the-fly的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

方式把具体变量赋值与抽象的静态预测监控器状态的状态关联起来,同时判定抽象迁移的使能情况.

对于$P_A$LTL公式$\varphi$,预测监控器的运行从初始状态$(\{(\rho_\varnothing,\varphi)\},\varnothing)$开始,其中,$\rho_\varnothing$表示空赋值.根据相应的非确定Büchi自动机的静态结构,对于任意抽象后继状态$(U',V')$,都存在从状态$(U,V)$迁移到状态$(U',V')$的迁移约束$\bar{x}\cup\bar{y}$与之相对应.即对于任意元素$\psi_k\in(U',V')$,存在$\langle t_k,\varphi_k\rangle$与之对应,如果$\varphi_k$包含存在谓词,那么基于存在谓词执行赋值提取操作$\Theta_k=spec(\sigma(t_k),w[j],\varphi_k)$,其中,$\sigma(t_k)$表示状态$t_k$处累积的变量赋值,$w[j]\in 2^{P_2}$为当前输入集合.如果$\Theta_k$不为空,那么生成新的集合$\overline{\Theta_k}=\{\{\beta\rangle\sigma(t_k)\mid\beta\in a\}\mid a\in\Theta_k\}$;如果$\varphi_k$为参数化命题,那么将执行绑定操作,判定是否$\widehat{\sigma(t_k)}(\varphi_k)\in w[j]$,如果$\widehat{\sigma(t_k)}(\varphi_k)\in w[j]$,那么生成集合$\overline{\Theta_k}=\{\{\sigma(t_k)\}\}$.只有当任意元素$\langle t_k,\varphi_k\rangle$都执行了相应的赋值提取和绑定操作,并且要么提取了非空赋值集合(赋值集合为空,同时$Q_1=\forall$除外),要么$\widehat{\sigma(t_k)}(\varphi_k)\in w[j]$,那么预测监控器将迁移到一个新的状态$S_r$.$S_r$为抽象状态$s$的运行时实体,是$s$在特定运行中的实例化,

$$S_r=(\{...,(\overline{\Theta_i},\psi_i),...\},\{...,(\overline{\Theta_j},\psi_j),...\}),0\le i\le n,0\le j\le m.$$

实际上,$S_r$代表了非确定Büchi自动机在当前输入下的所有可能运行.定义如下操作:

$$(\overline{\Theta_i},\psi_i)=\{\{(\eta,\psi_i)\mid\eta\in\theta\}\mid\theta\in\overline{\Theta_i},$$
$$\{...,(\overline{\Theta_i},\psi_i),...\}=\otimes\{(\overline{\Theta_i},\psi_i)\mid 0\le i\le n\},$$
$$\{...,(\overline{\Theta_j},\psi_j),...\}=\otimes\{(\overline{\Theta_j},\psi_j)\mid 0\le j\le m\}.$$

从上述定义容易看出,在预测监控器运行时,其静态结构中的任意抽象状态$s$对应于运行时状态集合$S_r$,我们称$S_r$为抽象状态$s$对应的运行时状态集.

从而,$s$和$S_r$之间的对应关系可以形式化描述如下:

$$\Phi_{\psi_k}^{\langle t_k,\varphi_k\rangle}=\begin{cases}\{\varnothing\}, & \text{iff } \varphi_k=Q_1x_1...Q_nx_np(u_1...u_m),\ Q_1=\forall \text{ and } \Theta_k=\varnothing\\ \{\{(\psi_k,\beta\rangle\sigma(t_k))\mid\beta\in\theta\}\mid\theta\in\Theta_k\}, & \text{iff } \varphi_k=Q_1x_1...Q_nx_np(u_1...u_m) \text{ and } \Theta_k\ne\varnothing\\ \{\{(\psi_k,\beta)\}\}, & \text{iff } \varphi_k \text{ is parametric proposition}\end{cases},$$

$$S_r=\{(x\in\otimes\{\Phi_{\psi_i}^{\langle t_i,\varphi_i\rangle}\},y\in\otimes\{\Phi_{\psi_j}^{\langle t_j,\varphi_j\rangle}\})\}.$$

面向$P_A$LTL的预测监控器从初始状态出发,依次判定每个后继状态的迁移约束$\bar{x}\cup\bar{y}$.对于任意基于$(\langle t_i,\varphi_i\rangle,\psi_i)\in\bar{x}\cup\bar{y}$,如果$\varphi_i$为存在谓词,则执行赋值提取操作,如果提取的赋值集合为空,那么$\varphi_i$将从后继状态中消失;如果提取的赋值集合不为空,则把当前状态累积的赋值以及新提取的赋值一起传递给后继状态;如果迁移边上标记为参数化命题,那么基于当前状态上的累积赋值判定迁移边的使能情况,如果迁移边被触发,就把前一状态的累积赋值传递给后继状态.如果存在$(\langle t_i,\varphi_i\rangle,\psi_i)\in\bar{x}\cup\bar{y}$不满足上述 3 种情况,那么迁移不能发生.

## 4 结 论

本文针对现实世界程序中对参数化性质的验证需求,以文献[1,6,7]以及文献[2–4]的工作为基础,提出面向参数化 LTL 公式预测监控器的构造技术和过程.

目前,面向参数化 LTL 公式的监控器以被监控性质的有穷轨迹语义为基础,存在两个方面的局限性:

- 为了保证其有穷轨迹语义的一致性,实际上采用的形式化规约不能包含 $X$ 操作符.
- 基于有穷轨迹语义的运行时验证器,往往识别无穷运行轨迹的 informative 前缀,而不是尽可能早地报告系统运行时异常.

此外,当前的预测监控器构造技术还没有考虑性质规约中包含一阶特性的情况.但实际上,在现实的顺序和并发程序中,许多性质的描述需要用到逻辑的参数化特性.例如,描述的性质需要在特定类型数据结构的所有实例上都成立.特别是对于系统中的某些动态特性描述,比如对象的实例化、进程/线程的产生、文件的打开和关闭等.因此,研究面向参数化性质规约的预测监控器构造技术具有重要的理论和应用价值.

本文首先描述了$P_A$LTL语法,引入一个特殊的bridge操作符,使得从语法层面限定了赋值集合的提取和绑定操作,并包含了$X$操作符.同时给出了$P_A$LTL基于无穷轨迹模型的标准语义,具体描述了赋值集合的提取、绑定、传递等操作.以此为基础,提出$P_A$LTL基于有穷轨迹的预测语义定义.面向$P_A$LTL的预测监控器以文献[4]中给出

© 中国科学院软件研究所 http://www.c-s-a.org.cn

的预测监控器构造的动态过程为基础,描述了参数化的运行时监控器的概念.它由静态和动态两个部分组成,静态部分描述系统的状态及状态间的迁移关系,动态部分描述了在不同状态处需要绑定的变量赋值.其静态部分的构造过程包括:

- 从$P_A$LTL到参数化交错Büchi自动机的转化;
- 从参数化交错 Büchi 自动机到非确定 Büchi 自动机的转化.

运行时验证器的动态部分是基于 bridge 操作符的变量赋值提取、绑定以及传递过程.

通过上述过程构造的监控器最终能够识别一个无穷运行轨迹的最小好/坏前缀,比传统的验证器(识别 informative 前缀)更早地发现系统异常,为系统行为修复和调控提供尽可能充分的时间和空间.我们将进一步研究发现系统异常后的故障诊断、修复机制以及在预测监控器的基础上,通过前瞻被监控对象的行为模型,在异常未发生之前预言系统失效的可能,并采用相应的预防措施.

**References**:

[1] Stolz V. Temporal assertions for sequential and concurrent programs [Ph.D. Thesis]. Aachen: RWTH Aachen University, 2007.

[2] Bauer A, Leucher M, Schallhart C. Runtime verification for LTL and PTLTL. Technical Report, TUM-I0724, München: Technische Universität München, 2007.

[3] Bauer A, Leucker M. Monitoring of real-time properties. In: Kumar A, Seth A, eds. Proc. of the Foundations of Software Technology and Theoretical Computer Science (FSTCS). LNCS 2937, Heidelberg: Springer-Verlag, 2006. 260−272.

[4] Dong W, Leucker M, Schallhart C. Impartial anticipation in runtime verification. In: Kim M, Viswanathan M, eds. Proc. of the 6th Int'l Symp. on Automated Technology for Verification and Analysis (ATVA 2008). LNCS 5311, Heidelberg: Springer-Verlag, 2008. 386−396.

[5] d'Amorim M, Rosu G. Efficient monitoring of $\omega$-languages. In: Etessami K, Rajamani S, eds. Proc. of the 17th Int'l Conf. on Computer-Aided Verification (CAV 2005). LNCS 3576, Heidelberg: Springer-Verlag, 2005. 101−112.

[6] Stolz V, Bodden E. Temporal assertions using AspectJ. Electronic Notes in Theoretical Computer Science, 2005,144(4):109−124.

[7] Stolz V. Temporal assertions with parametrised propositions. Journal of Logic and Computation, 2008. doi: 10.1007/978-3-540-77395-5

[8] Havelund K, Rosu G. Monitoring programs using rewriting. In: Alexander P, Berry Y, eds. Proc. of the Int'l Conf. on Automated Software Engineering (ASE 2001). San Diego: ACM Press, 2001. 135−143.

[9] Havelun K, Rosu G. Synthesizing monitors for safety properties. In: Ball T, Bouajjani A, eds. Proc. of the Tools and Algorithms for Construction and Analysis of Systems (TACAS 2002). LNCS 2280, Berlin: Springer-Verlag, 2002. 342−356.

[10] Drusinsky D. The temporal rover and the ATG rover. In: Havelund K, Penix J, eds. Proc. of the SPIN Model Checking and Software Verification. LNCS 1885, Springer-Verlag, 2000. 323−330.

[11] Drusinsky D. Monitoring temporal rules combined with time series. In: Somenzi F, eds. Proc. of the 15th Int'l Conf. on Computer-Aided Verification (CAV 2003). LNCS 2725, Springer-Verlag, 2003. 114−118.

[12] Finkbeiner B, Sipma H. Checking finite traces using alternating automata. Electronic Notes in Theoretical Computer Science, 2001, 55(2):44−60.

[13] Giannakopoulou D, Havelund K. Automata-Based verification of temporal properties on running programs. In: Alexander P, Berry Y, eds. Proc. of the Int'l Conf. on Automated Software Engineering (ASE 2001). San Diego: ACM Press, 2001. 412−416.

[14] Barringer H, Goldberg A, Havelund K, Sen K. Eagle monitors by collecting facts and generating obligations. Technical Report, CSPP-25, Manchester: Department of Computer Science, University of Manchester, 2003. 201−222.

[15] Barringer H, Goldberg A, Havelund K, Sen K. Rule-Based runtime verification. In: Steffen B, Levi G, eds. Proc. of the 5th Int'l Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI 2004). LNCS 2937, Heidelberg: Springer-Verlag, 2004. 44−57.

[16] Gerth R, Peled D, Vardi M, Wolper P. Simple on-the-fly automatic verification of linear temporal logic. In: Dembinski P, Sredniawa M, eds. Proc. of the 15th IFIP/WG6.1 Symp. on Protocol Specification Testing and Verification (PSTV'95). Warsaw: Chapman & Hall, 1995. 3−18.

[17] Kupferman O, Vardi MY. Model checking of safety properties. In: Halbwachs N, Peled D, eds. Proc. of the 11th Int'l Conf. on Computer Aided Verification. LNCS 1633, Trento: Springer-Verlag, 1999. 172–183.

[18] Bauer A, Leucker M, Schallhart C. The good, the bad, and the ugly—But how ugly is ugly? In: Havelund K, Rosu G, eds. Proc. of the 7th Int'l Workshop on Runtime Verification. Vancouver: Springer-Verlag, 2007. 126–138.

[19] Vardi MY. An automata-theoretic approach to linear temporal logic. Logics for Concurrency: Structure Versus Automata. New York: Springer-Verlag, 1996. 238–266.

**赵常智**(1980－),男,河南西峡人,博士生,主要研究领域为程序分析与验证.

**董威**(1976－),男,博士,副教授,CCF 会员,主要研究领域为高可信软件工程,软件测试与验证.

**隋平**(1982－),男,硕士生,主要研究领域为程序的运行时验证.

**齐治昌**(1944－),男,教授,博士生导师,主要研究领域为软件工程.