ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.9, September 2007, pp.2153−2161
DOI: 10.1360/jos182153

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

工作流时序约束模型分析与验证方法*

王 远+, 范玉顺

(清华大学 自动化系,北京 100084)

A Method of Time Constraint Workflow Model Analysis and Verification

WANG Yuan+, FAN Yu-Shun

(Department of Automation, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62789635 ext 1070, Fax: +86-10-62789650, E-mail: yuan-wang02@mails.tsinghua.edu.cn

Wang Y, Fan YS. A method of time constraint workflow model analysis and verification. *Journal of Software*, 2007,18(9):2153−2161. http://www.jos.org.cn/1000-9825/18/2153.htm

Abstract: In order to describe the temporal aspect of workflow model and verify the temporal consistency, a method based on temporal logic and model checking for modeling and verifying time constraint workflows is presented. By this method, the first order logic is used to model workflow including its basic temporal information, temporal logic is used to model the time constraints, and model checking is used to analyze and verify the temporal consistency. The method can be used to verify any time constraints that can be described by temporal logic. Also the method can offer a counterexample of workflow instance to the time constraint which can not pass the verification. Finally, the method is validated through a case study.

Key words: workflow; time constraints; verification; temporal logic

摘 要: 为了解决工作流时间建模与时序一致性验证问题,以时序逻辑和模型检查为基础,提出了一种工作流时间建模与时序一致性验证方法.该方法用一阶逻辑描述工作流模型及其时间信息,用时序逻辑描述工作流的时序约束,用模型检查算法对时序约束进行验证与分析.该方法不是针对某一种时序约束提出来的,而是能够验证任何用时序逻辑描述的工作流时序约束.该方法还能够对未通过验证的时序约束提供工作流运行实例作为反例,帮助用户定位模型的问题.以一个工作流时间建模和时序一致性验证的实例证实了所提出方法的有效性.

关键词: 工作流;时序约束;验证;时序逻辑

中图法分类号: TP311 **文献标识码**: A

1 问题定义

工作流管理是实现企业过程集成、提高企业运行效率和柔性的一种全面的支撑技术.随着工作流技术的发展,涌现了各种各样的工作流管理系统产品[1].然而近年来,动态多变的业务环境和激烈的市场竞争使企业中工作流系统对时间管理的需求日益迫切,可以说,时间管理问题已成为困扰工作流技术应用实施的一个重大难题.在工作流时间管理中,时间信息的建模、验证和分析是最基础、最核心的问题[2].工作流模型中的时间信息主要

* Supported by the National Natural Science Foundation of China under Grant No.60274046 (国家自然科学基金)
Received 2006-03-28; Accepted 2006-08-16

由各种时序约束(temporal constraints)来表示.时序约束表示工作流中活动执行的时间层约束,常根据活动本身的执行情况、企业业务策略以及法律法规来确定,根据时序约束的产生方式,工作流时序约束分为隐式约束和显式约束[3].隐式时序约束是由工作流控制结构与活动延迟而导出,如一个活动必须在其所有的前序活动执行完毕才可启动.显式时序约束常由过程设计者指定,如事件之间的时序关系、事件与某个日期集的绑定等.工作流的时序一致性指的是时序约束与工作流模型的一致性,以及工作流实例执行中的时序约束满足性,具体定义如下:

定义 1. 一个时序约束与某一给定的工作流模型是一致的,当且仅当基于工作流模型语义与工作流活动的执行延迟,该时序约束是满足的(在工作流模型正常执行过程中,描述该时序约束的表达式一直成立).

定义 2. 一个时序约束集与某一给定的工作流模型是一致的,当且仅当基于工作流模型语义和工作流活动的执行延迟,该集合所包含的所有时序约束是满足的.

到目前为止,工作流时间建模方法主要有基于工作流图的方法[3,4]和基于Petri网[5−7]的方法两种.其中,基于工作流图的方法是在工作流图的活动上加上时间信息形成赋时工作流图.基于Petri网的方法是在工作流网上对变迁或库所加入时间信息形成时间工作流网.在以上两种工作流时间信息建模方法的基础上,研究人员提出了一些时间信息计算方法和时序一致性验证方法.其中,在赋时工作流图中主要通过计算活动的最早/最晚完成时间(或活动之间的最长/最短时间距离)来验证工作流的时序一致性;基于Petri网的方法使用Petri网的相关技术来分析时间约束,检验工作流模型在特定时序约束下的可执行性.

经过对现有的时间建模、验证和分析方法的仔细研究,发现上述方法存在以下不足:具体的时间建模方法和时序一致性验证算法与所需验证的时间性质紧密相关.这些时间建模方法的共同特点就是针对一种时序约束,提出一种特殊的建模方法,在工作流模型中找到该约束的表达方式,并针对这种表达方式提出一种特殊的验证算法.这种模式不仅没有统一的方法框架,而且无法满足工作流时间模型多种性质的验证需要.比如,在建模方面,需要对多个活动之间的时间距离建模,现有的方法就都不适用.在时序一致性验证方面,对于两个活动之间的任意时间距离约束、多个活动之间的截止期限约束等,现有的方法也无法验证.另外,在指导工作流模型时间属性的设计方面,现有的方法无法满足需要.比如,要设计一个满足给定时间性质的工作流模型,现有的建模与验证方法无法提供有力的支持.针对上述方法中存在的问题,本文基于时序逻辑和模型检查(model checking)技术,提出了一种新的工作流时间建模与验证方法.该方法不是针对某一种时序约束提出来的,而是能验证任何可用时序逻辑描述的时序约束;如果在验证时发现时序一致性不满足,则该方法能够提供违反时序一致性的工作流运行实例作为反例,帮助用户定位模型的问题;同时,方法有大量比较成熟的模型检查工具的支持,在工程实践中易于实现和应用.

2 工作流时间建模与验证

本文使用一阶逻辑来对工作流及其基本的活动时间信息进行建模,使用时序逻辑描述工作流模型中的各种时序约束,使用模型检查算法对这些时序约束进行验证.

2.1 工作流时间建模方法

文献[8]给出了一种基于工作流管理联盟(workflow management coalition,简称WfMC)过程定义的图形化建模语言,利用两种对象实体进行过程建模:节点和有方向的连接弧.其中节点分为两种:任务节点和选择/汇合节点.任务节点用一个方框表示,选择/汇合节点(逻辑节点)用一个圆圈表示,用来表达“与分支”和“与连接”的逻辑结构.连接弧表示活动的前后逻辑控制约束.在此基础上,将活动标识K及其基本执行延迟d_k都标注在活动节点上就形成了一种赋时工作流图,下面给出用一阶逻辑来对赋时工作流图进行建模的方法.

要用一阶逻辑对工作流及其时间信息建模,必须用一阶逻辑描述工作流语义.在本文中,用于描述赋时工作流图的一阶逻辑系统在一个Kripke结构$M=(S,S_0,R,L)$上进行语义解释,其中,S表示工作流系统运行的状态集合,S_0表示工作流运行的初始状态,R表示随着时间的变化工作流系统状态的变迁关系,L为映射:$S\rightarrow 2^{AP}$,其中,AP为系统中所有原子命题的集合,$L(S)$表示在状态S下成立的原子命题的集合.为了描述工作流的执行过程和时间

信息,在一阶逻辑系统中定义了常量*Start*,*End*,*I*,*J*,*K*,…分别表示工作流的开始节点、结束节点以及工作流模型中的各项活动,系统中的变量定义如下:

(1) 向量$s[k]$,i=0,1,…,n+1,其中,n是工作流模型中除去开始和结束节点外的实际活动数量,$s[k]$的值域为{0,1},$s[k]$取1或0分别表示活动K处于执行或非执行状态.

(2) 全局时钟变量c,c的取值为整数,值域为[−1,N],N为一个大于工作流模型执行总时间的整数,在实际操作中可选 $N=\sum_{k=1}^{n} d_k$,其中n为工作流活动的总数量,c的变化表示系统的时钟运行.

(3) 局部时钟向量$lc[k]$,k=0,1,…,n+1,$lc[k]$的值域与变量c的值域相同,$lc[k]$的变化表示和活动K相关的局部时钟运行.

在上述变量定义的基础上还加入了带撇号的变量,带撇号的变量表示其处于时间轴上的下一个状态之中,如 $s[k]'$表示 $s[k]$在时间轴上下一个时刻的取值.现在可以用上述一阶逻辑系统来描述工作流随系统时钟运行的状态转换,本文采用对工作流中每个活动的状态转换分别进行描述的方法,整个工作流的状态是各个活动状态的集合.工作流模型中存在如图 1 所示的 6 种基本控制结构,活动位于不同的控制结构中,其状态转换的一阶逻辑描述也是不一样的,下面分别给出它们的描述方法.

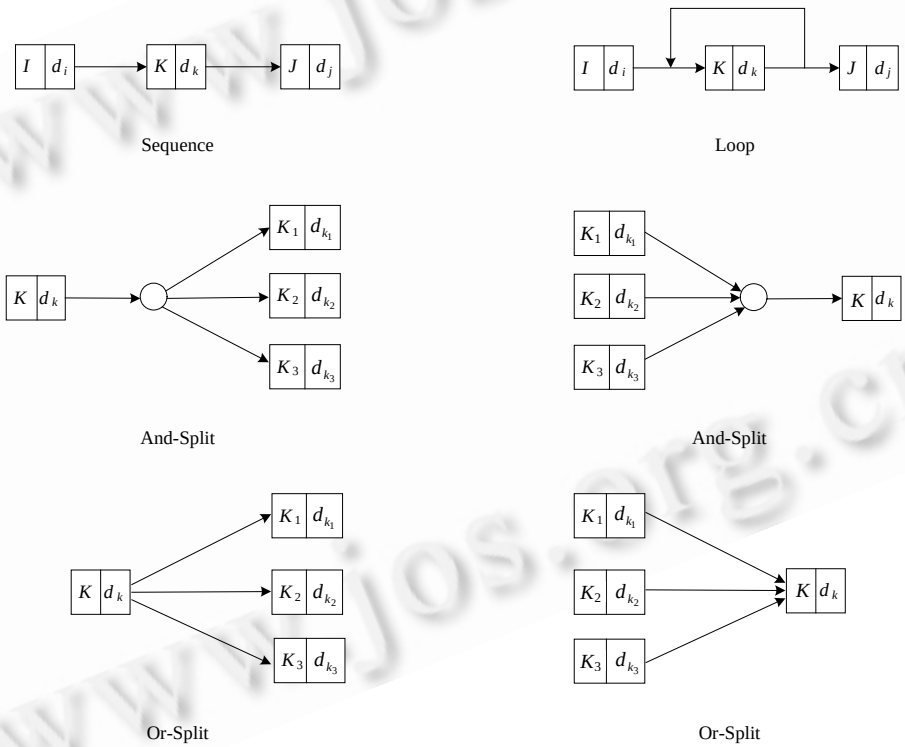

Fig.1 Control structure in workflow model

图 1 工作流模型的控制结构

顺序结构中的活动:顺序结构用来描述一系列固定顺序串行执行的活动.位于顺序结构中的活动 K 的状态转换由以下 5 个逻辑公式的并来表示(记作 $SE(k)$):

$$c'=c+1 \tag{1}$$

$$(lc[k]=-1)\wedge(lc[k]'=-1)\wedge(s[k]'=0) \tag{2}$$

$$(lc[k]\geq d_k)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=s[k]) \tag{3}$$

$$(0\leq lc[k]<d_k-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=s[k]) \tag{4}$$

$$(lc[k]=d_k-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=0)\wedge(lc[j]'=0)\wedge(s[j]'=1) \quad (5)$$

$$SE(k)=(1)\wedge(2)\vee(3)\vee(4)\vee(5) \quad (6)$$

下面对上述各逻辑公式的物理意义作简单说明.总的来说,上面的逻辑公式描述的是工作流系统时钟每向前运行一个计时粒度,位于顺序控制结构中的一个工作流活动的状态转换过程.式(1)表示系统时钟的运行;式(2)和式(3)分别描述在活动 K 没有执行之前和执行完毕之后这两段时间内,随着系统时钟的运行,活动 K 状态的转换,其中,为了验证时序一致性的需要,在活动 K 执行完毕之后,令活动 K 的局部时钟继续运行;式(4)描述了活动 K 执行过程中的状态转换,在 K 执行过程中局部时钟从 0 开始运行;式(5)描述了活动 K 执行结束时刻的状态转换,其中包括对顺序结构中后续活动 J 的激活.

与分支结构中的活动:与分支描述了在一个活动执行完后,工作流分解为几个并行执行的分支,允许多任务的并行执行.在与分支结构中,分支后面的并行活动$K_1,K_2,\ldots,K_w$的逻辑描述与顺序结构中的活动一样,分支前的活动K的逻辑描述由 5 个逻辑公式连接而成(记作$AS(k)$),其中,前 4 个逻辑公式与式(1)~式(4)相同,第 5 个逻辑公式为

$$(lc[k]=d_k-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=0)\wedge(lc[k_1]'=0)\wedge(s[k_1]'=1)\wedge\ldots\wedge(lc[k_w]'=0)\wedge(s[k_w]'=1) \quad (7)$$

$$AS(k)=(1)\wedge(2)\vee(3)\vee(4)\vee(7) \quad (8)$$

式(7)描述了分支前活动K执行结束时刻的状态转换,其中包括对分支后各个并行活动$K_1,K_2,\ldots,K_w$的激活,w为分支的数量.

与连接结构中的活动:与连接描述了多个并行的分支活动汇合成一个活动的控制结构,它要求所有汇合的分支都执行完毕后,汇合后的活动才能执行.在与连接结构中,连接前的并行活动$K_1,K_2,\ldots,K_w$的逻辑描述与顺序结构中的活动一样,汇合后的活动K的逻辑描述由 6 个逻辑公式连接而成(记作$AJ(k)$),其中,前 4 个逻辑公式与式(1)、式(2)、式(3)、式(5)相同,其余的 2 个逻辑公式为

$$(lc[k]=0)\wedge(lc[k_1]\geq d_{k_1}-1)\wedge\ldots\wedge(lc[k_w]\geq d_{k_w}-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=s[k]) \quad (9)$$

$$(0<lc[k]<d_k-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=s[k]) \quad (10)$$

$$AJ(k)=(1)\wedge(2)\vee(3)\vee(5)\vee(9)\vee(10) \quad (11)$$

式(9)描述了汇合的分支都执行完毕后,汇合后的活动 K 才能激活执行,其中,w 为分支的数量.式(10)描述了活动 K 执行过程中的状态转换.

或分支结构中的活动:或分支描述彼此之间具有相互排斥关系的分支活动,分支点之后的活动只能选择一个执行.在或分支结构中,分支后面活动的逻辑描述与顺序结构中的一样,分支前的活动 K 的逻辑描述也是由 5 个逻辑公式的连接表示(记作 $OS(k)$),其中,前 4 个逻辑公式与式(1)~式(4)相同,第 5 个逻辑公式为

$$((lc[k]=d_k-1)\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=0))\wedge((h[k_1]\wedge\neg h[k_2]\wedge\ldots\wedge\neg h[k_w])\vee\ldots\vee(\neg h[k_1]\wedge\neg h[k_2]\wedge\ldots\wedge h[k_w])) \quad (12)$$

$$OS(k)=(1)\wedge(2)\vee(3)\vee(4)\vee(12) \quad (13)$$

其中,$h[k_i]=(lc[k_i]'=lc[k_i]+1)\wedge(s[k_i]'=1)$.式(12)表示分支点后的活动只能选择一个执行.

或连接结构中的活动:或连接控制结构描述彼此之间互斥的分支活动的汇合.在或连接结构中,汇合前各活动的逻辑描述与顺序结构中相同.汇合后的活动 K 的逻辑描述由 6 个逻辑公式连接而成(记作 $OJ(k)$):

$$((lc[k]=0)\wedge((lc[k_1]\geq d_{k_1}-1)\vee\ldots\vee(lc[k_w]\geq d_{k_w}-1))\wedge(lc[k]'=lc[k]+1)\wedge(s[k]'=s[k]) \quad (14)$$

$$OJ(k)=(1)\wedge(2)\vee(3)\vee(5)\vee(10)\vee(14) \quad (15)$$

在一阶逻辑公式 $OJ(k)$中,式(1)~式(3)、式(5)、式(10)的物理意义和与连接结构中的一样.式(14)描述的是系统时钟从任意一个或连接分支执行过程中的最后一个计时粒度运行到汇合后活动 K 开始执行的过程中 K 的状态转换,即汇合的过程.

循环结构:循环结构用来描述需要多次执行的活动,在实际的执行过程中,循环的次数最终总是会被确定的.如果循环的次数不能确定(或者是无穷循环),那么循环的执行时间就有可能是单次循环时间的任意整数倍,这种情况下显然无法进行时序一致性验证,所以,本文假设循环的次数在进行时序一致性验证之前已经确定或其上限已经确定,这样循环就可展开成一个顺序结构,其中,活动的逻辑描述方法与顺序结构中的相同.

为了完整地描述工作流模型,还需要给出工作流系统的初始状态以及开始和结束节点的逻辑描述,这里,*Start* 和 *End* 分别是开始和结束节点对应的状态变量编号.

系统初态:

$$\left(s[k]=\begin{cases}1, & k=Start\\ 0, & k\neq Start\end{cases}\right)\wedge\left(lc[k]=\begin{cases}0, & k=Start\\ -1, & k\neq Start\end{cases}\right)\wedge(c=-1) \tag{16}$$

开始节点逻辑描述:

$$(c'=c+1)\wedge(s[Start]'=0)\wedge(s[k_f]'=1)\wedge(lc[k_f]'=lc[k_f]+1) \tag{17}$$

其中,K_f是开始节点后的第 1 个实际活动.

结束节点逻辑描述:

$$(c<N)\wedge(c'=c+1)\wedge(s[End]'=s[End])\wedge(lc[End]'=lc[End]+1) \tag{18}$$

2.2 工作流模型时序约束建模与验证方法

在第 2.1 节提出的工作流时间建模方法的基础上,可以方便地进行工作流时序约束的建模与验证.本文运用时序逻辑对时序约束进行建模,采用的时序逻辑系统是 CTL(computation tree logic).CTL 是在普通的一阶逻辑基础上增加了 2 个量词 *A*(for all computation paths),*E*(for some computation path)以及 4 个时序操作符 *X*(next), *F*(eventually),*G*(always),*U*(until)后形成的逻辑系统,对 CTL 详细的介绍见文献[9].CTL 具有很强的描述能力,不仅能够描述现有的一些时序约束类型,而且能够描述大量非常复杂和个性化的时序约束.CTL 的上述特点使得本文提出的方法可以描述并验证大量不同形式的时序约束,而不是像现有的方法那样只能验证一种或几种特定的时序约束.下面首先针对几种常见的时序约束给出其 CTL 的描述方法:

有限延迟约束:限制工作流模型所表示的过程延迟适用于过程所有的实例类,比如,限制整个工作流模型的执行时间t在某一区间$[t_{\min},t_{\max}]$内,在$t_{\min}=0$时,$t_{\max}$就是过程的截至期限.有限延迟约束在CTL中的描述如下:

$$AF((s[End]=1)\wedge(t_{\min}\le c\le t_{\max})) \tag{19}$$

式(19)的物理含义是:工作流模型所有的执行实例都会达到一个状态,在该状态下,工作流已经执行完毕并且系统时钟运行到区间$[t_{\min},t_{\max}]$内.

截止期限(或期限时间):限制实例执行中活动或过程的开始或结束时间.比如,限制某活动K必须在时间区间$[t_{s\min},t_{s\max}]$内开始,在区间$[t_{e\min},t_{e\max}]$内结束.期限时间在CTL中用下面的公式描述:

$$AF((lc[k]=0)\wedge(t_{s\min}\le c\le t_{s\max}))\wedge AF((lc[k]=d_k)\wedge(t_{e\min}\le c\le t_{e\max})) \tag{20}$$

时间距离约束:限制同一工作流模型中两个任务之间的时间距离,比如,某活动K执行结束的时间与活动J开始的时间间隔值必须在区间$[t_{\min},t_{\max}]$内.这一时间距离约束在CTL中用下面的公式描述:

$$AF((lc[j]=0)\wedge(t_{\min}\le lc[k]-d_k\le t_{\max})) \tag{21}$$

除了上述常见的时序约束之外,还可以在 CTL 中描述现有方法不能描述和验证的约束.比如,3 个活动 *I*,*J*,*K* 的结束时间之间的最大差值不能超过 *t*,在 CTL 中可以描述为下面的公式:

$$C(I,I,K)=C_1\wedge C_2 \tag{22}$$

其中,

$$C_1=AF((lc[k]=d_k)\wedge((lc[i]-d_i)^2<t^2)\wedge((lc[j]-d_j)^2<t^2)) \tag{23}$$

$$C_2=AF((lc[i]=d_i)\wedge((lc[j]-d_j)^2<t^2)) \tag{24}$$

上述的各种时序约束都属于最终属性(eventually properties),它表示满足这些时序约束的工作流系统执行状态最终一定会出现.CTL 中还允许用户设计并验证一些可能属性(possible properties),它表示满足这些时序约束的工作流系统执行状态可能会出现,这样的时序约束是现有方法所不能描述的.比如,活动 *K* 的开始时间是否有可能比活动 *J* 的结束时间早 5 个以上的时间单位,在 CTL 中用下面的公式描述:

$$EF((lc[k]\ge 0)\wedge((lc[j]=d_j-5)) \tag{25}$$

虽然这类约束大多都可以转换为最终属性类的约束进行处理,但是提供这样的约束描述与验证方法为工作流系统用户在约束设计和建模方面提供了很大的方便.在工作流一阶逻辑模型的基础上,还可以用 CTL 描述

与验证工作流模型其他性质,比如,可以用下面的命题描述工作流模型的逻辑正确性:

$$AF(s[End]=1) \tag{26}$$

可见,CTL 对工作流模型的时序约束以及其他性质有着很强的描述和验证能力,用户可以根据具体的业务需求,结合工作流系统中状态变量的变化,设计和验证各种需要的时序约束及其他模型性质.

上述用CTL描述的时序约束可以用模型检查方法进行验证,模型检查的主要思想是检测用状态迁移系统描述的有穷状态系统的行为是否满足用时序逻辑描述的系统性质,模型检查最初是20世纪80年代为了进行硬件设计的验证而提出来的,随着符号模型检查(symbolic model checking)算法的出现,模型检查逐渐应用到通信协议、控制系统和软件设计等领域的性质验证,也出现了SMV,SPIN,CWB等著名的模型检查工具[10−13].本文使用的工具是由Carnegie Mellon大学和Cadence实验室联合开发的模型检查工具SMV2.5.为了方便用户使用,模型检查工具中都有自己的系统建模语言,一般与第 2.1 节中使用的一阶逻辑在形式和语法上并不完全一样(SMV的建模语言参见文献[13]),这就需要将第 2.1 节中对活动的描述向SMV的语言上做一个映射,将用一阶逻辑描述的工作流模型映射成SMV能够接受的模型.由于一阶逻辑和SMV中的模型描述语言都是根据系统状态变量的变化来描述系统的状态迁移,它们之间很相似,所以这个映射比较简单,作者已经开发了相应软件自动完成这项工作.这里不再列出所有的映射规则,只以顺序结构中活动的映射为例来说明方法.顺序结构中活动K的描述可以映射成为SMV中的如下语言单位:

$next(c):=c+1$

if $((lc[k]=-1)$ $\{next(lc[k]):=-1;next(s[k]):=0;\}$

else if $((lc[k]>=0)$ & $(lc[k]<>(d_k-1)))$

$\{next(lc[k]):=lc[k]+1;next(s[k]):=s[k];\}$

else if $(lc[k]:=d_k-1)$

$\{next(lc[k]):=lc[k]+1;next(s[k]):=0;next(lc[j]=0;next(s[j]):=1)$

从这个映射可以看出,式(6)中“并”的逻辑关系通过 SMV 中不同的选择分支实现,式(2)~式(5)中“交”的逻辑关系通过 SMV 中 if 条件和条件满足后的语句之间的配合实现.在时序约束的描述方面,SMV 中使用的语言就是 CTL,所以,描述时序约束的 CTL 命题可以直接输入 SMV 进行验证.

从上述的时序约束建模和验证方法中可以看出,由于使用模型检查作为验证的手段,所以只要是能用 CTL 描述的时序约束都能进行验证,而 CTL 强大的描述能力使得本文的方法可以验证几乎所有可能产生的时序约束.同时,模型检查算法的特点使得在约束未通过验证时,能够将一个工作流实例作为反例提供给用户.

3 实 例

图 2 给出了一个用赋时工作流图描述的工作流模型,图中每个活动的左边是活动标识,右边是活动的执行延迟时间.表 1 中列出了在这个工作流模型中我们想要验证的时序约束及其相应的 CTL 描述.

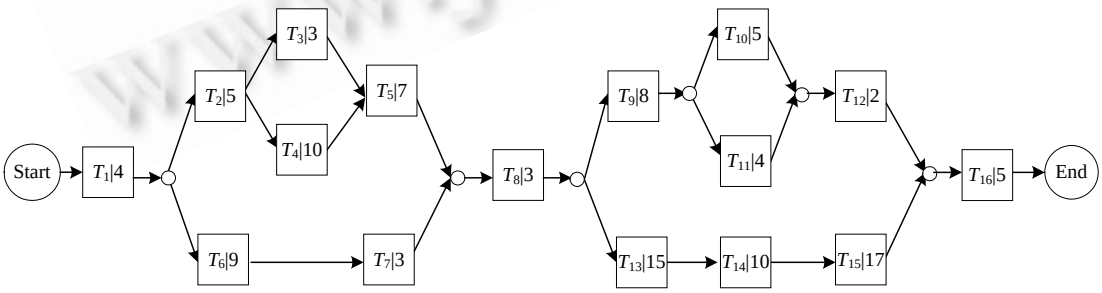

Fig.2 Example of time constraint workflow model

图 2 工作流时序约束模型实例

Table 1 Time constraints of workflow need to be verified

表 1 工作流中需要验证的时序约束

Time constraints	Description	CTL denotation
*Tc*1	The workflow will be finished in 80 time intervals	$AF((s[17]=1)\wedge(c\leq 80))$
*Tc*2	Task T_8 will start in 15 time intervals after the workflow starts	$AF((lc[8]=0)\wedge(c>15))$
*Tc*3	The time intervals from the end of task T_6 to the start of task T_{10} is longer than 20	$AF((lc[10]=0)\wedge(lc[6]-9>20))$
*Tc*4	The time intervals between the end of the task T_{10}, T_{11} and T_{15} is on longer than 10	$f_1\wedge f_2$, where in, $f_1=AF((lc[15]=17)\wedge((lc[10]-5)^2<100))\wedge((lc[11]-4)^2<100))$ $f_2=AF((lc[10]=5)\wedge((lc[11]-4)^2<100))$
*Tc*5	Task T_5 is possibly starts earlier than task T_7	$EF((lc[5]=0)\wedge(lc[7]<0))$
*Tc*6	The workflow model is correct in control logic	$AF((s[17]=1))$

首先对该工作流及其时间信息用一阶逻辑进行建模,然后将模型映射成 SMV 中可以接受的模型,并将时序约束输入 SMV.在实际输入时序约束时,需要将约束 *Tc*5 转换成如下命题进行验证:

$$AF((lc[7]=0)\wedge(lc[5]<0)) \tag{27}$$

式(27)描述的是*Tc*5 的逆命题"活动T_7总是早于T_5开始".如果式(27)验证通过,则*Tc*5 未通过验证;如果式(27)未通过验证,则SMV提供的反例就是*Tc*5 成立的一个实例.图 3 是SMV的主界面,在该界面中已经打开了模型文件,在主界面选择菜单Prop中的Verify all来验证所有的 6 个时序约束,验证的结果如图 4 所示.从图中可以看出,时序约束*Tc*1 和*Tc*2 通过验证,而*Tc*3 没有通过验证,SMV在遇到一个未通过验证的系统性质时就会停止验证,所以剩下的几个时序约束没有进行验证.对于未通过验证的性质,SMV会通过系统状态跟踪的形式给出一个反例.从图 4 状态跟踪结果来看,工作流系统有可能处于这样一种运行状态:当系统时钟运行到 30 个时间单位时,活动T_{10}开始执行,而这一时刻,活动T_6的局部时钟运行到 26 个时间单位,所以,T_6运行结束后所经历的时间单位是$lc[6]-9=17$.这种情况下,从T_6结束到T_{10}开始之间的时间间隔小于 20,所以*Tc*3 不满足.下面把*Tc*3 从时序约束集合中删除,重新对系统进行验证,验证的结果如图 5 所示.图中显示,*Tc*5 的逆命题未通过验证,所以*Tc*5 通过验证.对反例的状态跟踪给出了一种活动T_5早于T_7开始的情况,可以看出,这是在活动T_2后的或分支中选择活动T_3执行的结果.将not *Tc*5 从时序约束集合中删除,则剩下的时序约束全部通过验证.

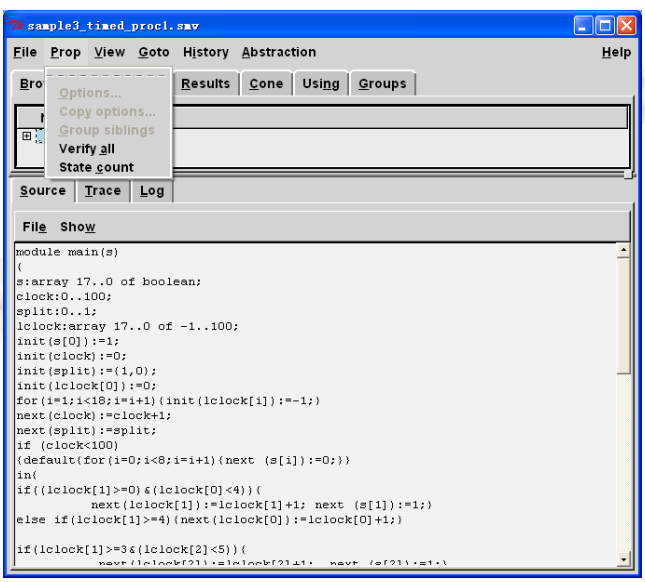

Fig.3 Main interface of SMV

图 3 SMV 主界面

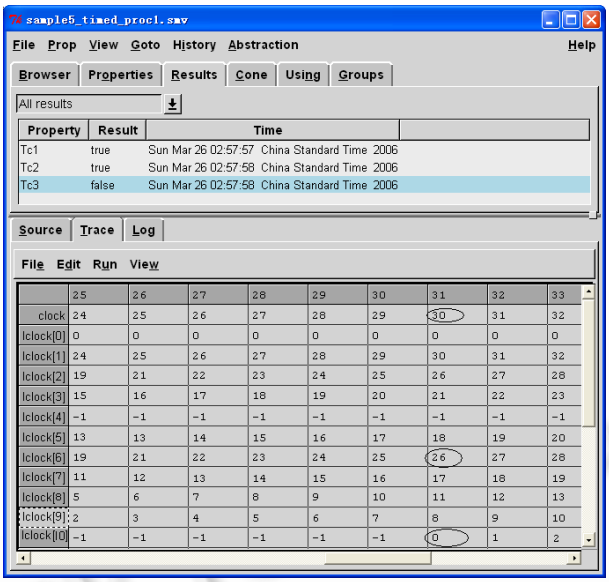

Fig.4 Result of verification and state trace of time constraint *Tc*3

图 4 验证结果以及对时序约束 *Tc*3 的状态跟踪

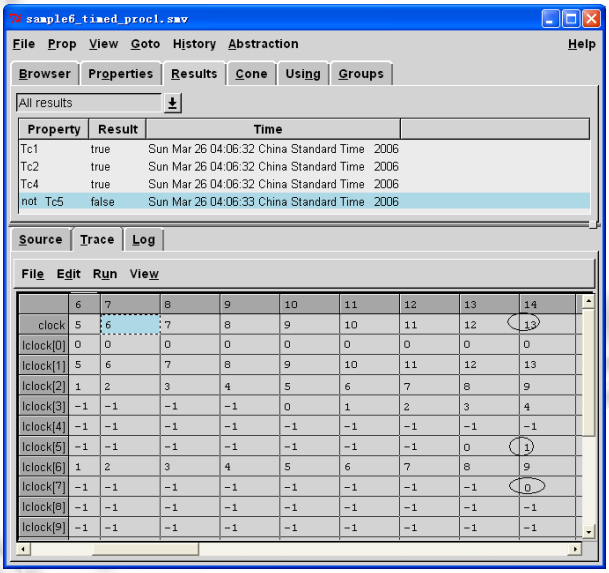

Fig.5 Result of verification and state trace of time constraint *Tc*5

图 5 验证结果以及对时序约束 *Tc*5 的状态跟踪

4 结束语

本文仔细研究了已有的工作流时间建模和验证方法,针对现有方法中存在的问题,以时序逻辑和模型检查作为理论基础,提出了一种新的工作流时间建模与验证方法.本文提出的方法用一阶逻辑描述工作流模型及其时间信息,用时序逻辑 CTL 描述工作流的时序约束,用模型检查算法对时序约束进行验证与分析.该方法不是针对一种时序约束提出来的,而是能够验证任何用 CTL 描述的时序约束.如果在验证的时候发现时序一致性不满足,该方法还能够提供违反时序一致性的工作流运行实例作为反例,帮助用户定位模型的问题.该方法的另外一个重要特点是,它不仅能够解决工作流时间建模与分析的问题,而且适用于工作流模型的逻辑正确性、性能等

一系列工作流模型验证与分析问题.同时,本文的方法有大量的比较成熟的模型检查工具的支持,在工程实践中易于实现和应用.本文提出的方法克服了以往研究成果中时间建模与验证方法无法统一以及验证能力不足的缺点,使得工作流研究人员可以采用统一的方法进行时间建模和时序一致性验证.在本文研究的基础上,可以在以下几个方面展开进一步的工作:在建模方面,本文只给出了工作流模型核心的控制流程和基本的时间信息的建模方法,由于工作流的语义十分丰富,所以对诸如组织、资源等语义的建模方法的研究是下一步的目标;在验证分析方面,如何有效地压缩模型的状态空间,使得本文的方法真正能够有效地解决大型工作流模型的时间验证与分析是一个重要的研究方向;在指导工作流模型的设计和综合方面,一阶逻辑模型的求精技术也是需要进一步研究的内容.

References:

[1] Fan YS. Fundamentals of Workflow Management Technology. Beijing: Tsinghua University Press/Springer-Verlag, 2001. 5−22 (in Chinese).

[2] Li HF, Fan YS. Overview on managing time in workflow systems. Journal of Software, 2002,13(8):1552−1558 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1552.pdf

[3] Eder J, Panagos E, Rabinovich M. Time constraints in workflow systems. In: Jarke M, Oberweis A, eds. Proc. of the 11th Conf. on Advanced Information Systems Engineering. Heidelberg: Springer-Verlag, 1999. 286−300.

[4] Eder J, Panagos E, Pozewaunig H, Rabinovich M. Time management in workflow systems. In: Abramowicz W, Orlowska ME, eds. Proc. of the 3rd Int'l Conf. on Business Information Systems. Berlin: Springer-Verlag, 1999. 265−280.

[5] Bowden FD. A brief survey and synthesis of the roles of time in Petri nets. Mathematical and Computer Modeling, 2001,31(3): 55−86.

[6] Serthomieu B, Diaz M. Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. on Software Engineering, 1991,17(3):259−273.

[7] Tsai J, Yang SJ. Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications. IEEE Trans. on Software Engineering, 1995,21(1):32−49.

[8] Sadiq W, Orlowska ME. Analyzing process models using graph reduction techniques. Information System, 2000,25(2):117−134.

[9] Edmund M, Clarke J, Orna G, Doron AP. Model Checking. Cambridge: MIT Press, 2001.

[10] Kifer M, Subrahmanian VS. Theory of generalized annotated logic programming and its applications. Journal of Logic Programming, 1992,12(4):1−33.

[11] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engineering, 1997,23(5):279−295.

[12] Cleavel R, Parrow J, Steffen B. The concurrency workbench: A semantics based verification tool for the verification of concurrent systems. ACM Trans. on Programming Languages and Systems, 1993,5(1):36−72.

[13] McMillan KL. The SMV language. 2001. http://www.cs.cmu.edu/~modelcheck/smv.html

附中文参考文献:

[1] 范玉顺.工作流管理技术基础.北京:清华大学出版社,2001.5−22.

[2] 李慧芳,范玉顺.工作流系统时间管理.软件学报,2002,13(8):1552−1558. http://www.jos.org.cn/1000-9825/13/1552.pdf

王远(1976－),男,山东文登人,博士生,主要研究领域为工作流管理技术,企业建模方法与优化分析技术.

范玉顺(1962－),男,博士,教授,博士生导师,主要研究领域为网络化制造,企业建模方法与优化分析技术,企业经营过程重组与工作流管理,系统集成与集成平台技术,面向对象与柔性软件系统技术,Petri 网建模与分析技术,车间管理与控制技术.