基于时态逻辑的工作流分析*)

周从华[1] 陶志红[2] 陈 钟[3] 王立福[3]

(江苏大学计算机科学与通信工程学院 江苏镇江 212013)[1]

(北京大学软件与微电子学院 北京 100871)[2] (北京大学信息科学技术学院 北京 100871)[3]

摘 要 工作流的属性规约语言需要具有高表达力以及基于状态和事件的推理能力。本文提出了一种时态逻辑规约语言 E-CTL*,该语言集成了状态和事件,能够精确和直觉地表示验证的属性。工作流的畅通性验证无论在时间上还是空间上代价都是非常高的,状态空间爆炸是验证的主要困难所在。利用 E-CTL* 描述畅通性,可以使用存在的符号化模型检测工具验证畅通性,在一定程度上克服了状态爆炸问题。同时模型检测技术给出失效路径的优点可以引导我们纠正工作流的错误。工作流的变动需要具有正确性。从时态逻辑的角度讨论了变动正确性问题,得出了保持变动正确性的一般特征。

关键词 工作流,时态逻辑,过程变动,同步网

Temporal Logic-based Workflow Analysis

ZHOU Cong-Hua[1] TAO Zhi-Hong[2] CHEN Zhong[3] WANG Li-Fu[3]

(School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013)[1]

(School of Software and Microelectronics, Peking University, Beijing 100871)[2]

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)[3]

Abstract Specification languages for workflow need to combine practical algorithmic efficiency with high expressive power and the ability to reason about both states and events. We address this question by defining a new temporal logic E-CTL* which integrates both state-based and event-based properties and allows us to represent properties precisely and naturally. The verification of throughness of workflow logic is very expensive both in time and space. The state explosion is the main difficulty. Describing the throughness by E-CTL* allows us to use symbolic model checkers to verify the throughness. Therefore the state explosion can be overcame efficiently. When the throughness is not satisfied by the workflow, model checking can give a counterexample which can guide us to correct the workflow. The change of workflow needs to keep throughness. We discuss the correctness of change based on temporal logic E-CTL* and obtain the character of correct change.

Keywords Workflow, Temporal logic, Process change, Synchronization net

1 引言

工作流逻辑[1]的正确性对于业务处理过程是非常重要的。一个在定义过程中包含错误定义和描述的工作流,在实施过程中会产生非常严重的后果。而由于工作流逻辑的复杂性,基于 Petri 网的工作流逻辑的正确性验证无论在时间上还是空间上代价都是非常高的,状态爆炸问题是验证的主要困难所在。典型的基于 Petri 网的工作流逻辑具有 50～1000 库所和变迁,如果 Petri 网是 k 界的,那么可能会有个 k^{1000} 状态。目前存在的工具还无法处理这么多的状态。模型检测[2~4]是验证反应式系统强有力的技术,在模型检测中用时态逻辑描述属性规约,用有限状态机描述系统行为,其主要任务是判定系统模型是否满足规约。目前,存在许多强大的模型检测器,如 NuSMV[5,6],SPIN[7],其中 NuSMV 能够处理具有 10^{20} 个状态的系统,因此使用模型检测器来验证工作流的正确性是一种可行的方法。模型检测一般聚集于有限状态模型和分时时态逻辑 CTL[8]或者线性时态逻辑 LTL[9]。对于工作流,传统的时态逻辑规约表达能力有限,我们必须有能力表达更加复杂的属性以及基于状态和事件的推理能力。

文本提出了一种新的时态逻辑规约语言 E-CTL*。E-CTL* 是时态逻辑 CTL* 的自然扩展,同时集成了状态和事件。这种集成不仅可以描述系统状态的变化而且可以体现事件之间的依赖关系。文中利用 E-CTL* 描述了工作流逻辑的畅通性、变迁引发规则和一般的正确性需求。由于 E-CTL* 算法复杂性不高于 CTL*,这使得可以使用存在的模型检测工具验证 E-CTL* 属性,在一定程度上克服了状态爆炸问题。

企业中存在着大量的业务流程,根据环境和客户的需要及时快速地调整业务流程,已经成为企业经营能力的体现。文[10]中对引起流程变动的原因进行了分类概括,从中可以看到流程变动的必要性。工作流技术的出现旨在为业务流程提供自动化支持,因此研究实际操作中存在滥些流程变动和它们符合的性质显得尤为重要。一般来说,工作流管理包含两个关键的阶段,即过程模型的建立和过程实例的执行,而过程变动则相应地发生在两个层次过程模型和过程实例。过程模型的变动将引发相关过程实例的变动,涉及到已有过程实例的处理,以及变动前后过程模型之间的关系等问题,这是解

*)国家自然科学基金助项目(No. 60573046)。**周从华** 博士,讲师,主要研究领域为模型检测、软件可靠性、工作流。

决过程变动的关键。而过程实例的变动影响到的只是当前一个实例，大多与异常处理相关联。虽然用户可以根据需要进行过程修改，但不能没有约束地随意变动，这就需要在变动中保持过程原有的性质(如畅通性)，即需要考虑工作流的变动正确性问题。虽然许多工作流项目就此问题提出了解决方案，但因为工作流模型本身的限制，使得处理过程变动的方法在表达能力和复杂性等方面存在问题。本文从时态逻辑的角度分析了各类基本过程变动，得出了不影响畅通性的变动的特征。

2 Petri 网系统

定义 2.1 三元组 $N=(P,T;F)$称为有向网的充分必要条件是：

(1)$P\cap T=\emptyset$；

(2)$P\cup T\neq\emptyset$；

(3)$F\subseteq P\times T\cup T\times P$；

(4)$dom(F)\cup cod(F)=P\cup T$，其中 $dom(F)=\{x\mid \exists y:(x,y)\in F\}$，$cod(F)=\{y\mid \exists x:(x,y)\in F\}$，它们分别为 F 的定义域和值域。

其中 P 和 T 分别称为 N 的库所集和变迁集，F 为流关系。设 $x\in P\cup T$ 为 N 的任一元素，$\cdot x=\{y\mid (y,x)\in F\}$称为 x 的输入集，$x\cdot=\{y\mid (x,y)\in F\}$称为 x 的输出集。为方便起见，我们定义下述符号：$\mathcal{M}_0=\{0,1,2,\cdots\}$，$\mathcal{N}=\{1,2,3,\cdots\}$，$\omega$ 表示无穷。

定义 2.2 (1)$K:P\to\mathcal{N}\cup\{\omega\}$称为 N 的容量函数。

(2)对给定的容量函数 K，$M:P\to\mathcal{M}_0$ 称为 N 的一个标识的条件是：$\forall_p\in P:M(p)\leqslant K(p)$。

(3)$W:P\to\mathcal{N}$称为 N 上的权函数，对$(x,y)\in T$，$W(x,y)=W((x,y))$称为(x,y)的权。

定义 2.3 六元组$\sum=(P,T;F,K,W,M_0)$构成 P/T 网的条件是：

(1)$N=(P,T;F)$构成有向网，称为$\sum$的基网。

(2)K,W,M_0 依次为 N 上的容量函数、权函数和初始标识。M_0 称为$\sum$的初始标识。

定义 2.4(变迁发生条件) t 在 M 标识有发生权的条件是：$\forall p\in\cdot t:M(p)\geqslant W(p,t)\wedge\forall p\in t\cdot:M(p)+W(p,t)\leqslant K(p)$。

t 在 M 下有发生权记作 $M[t>$，也就是说 t 在 M 下可引发。

定义 2.5(变迁发生后果) 若 $M[t>$，则 t 在 M 下可以引发，将标识 M 改变为 M 的后继 M'，M' 的定义是对 $\forall p\in P$，

$$M'(p)=\begin{cases}M(p)-W(p,t) & :p\in\cdot t-t\cdot\\ M(p)+W(t,p) & :p\in t\cdot-\cdot t\\ M(p)-W(p,t)+W(t,t) & :p\in\cdot t\cap t\cdot\\ M(p) & :p\notin\cdot t\cup t\cdot\end{cases}\tag{1}$$

3 时态逻辑 E-CTL*

Kripke 结构[3]描述了系统状态之间的变化，但是不能刻画引起状态变化的事件的发生。我们采用文[11]中定义的标记 Kripke 结构(LKS)来描述系统行为。该模型组合了状态和事件，既体现了状态之间的变化，又描述了事件之间的关联。

定义 3.1(标记 Kripke 结构) 设 AP 表示一个原子命题集合，一个基于 AP 的标记 Kripke 结构 $\kappa=(S,s_0,\Delta,R,L)$，是如下的一个五元组且满足下面条件：

(1)S 是一个有限状态集合；

(2)$s_0\in S$ 是初始状态；

(3)Δ 是一个事件集合；

(4)$R\subseteq S\times\Delta\times S$ 是一个完全的转移关系，也就是说对于每个状态 $s\in S$ 存在事件 $\delta\in\Delta$，一个状态 $s'\in S$ 使得 $R(s,\delta,s')$成立；

(5)$L:S\to 2^{AP}$ 是一个标记函数，用于标记该状态下取真值的原子命题集。

在标记 Kripke 结构中一条从状态 s 出发的路径是一个由 s 开始的状态和事件交替出现的无限序列 $s_0,\delta_0,s_1,\delta_1\cdots$，使得 $s=s_0$ 且对于每个 i，$R(s_i,\delta_i,s_{i+1})$成立。我们用 π^i 来表示从状态 s_i 开始的路径，即 $\pi^i=s_i,\delta_i,\cdots$，用 $\pi(i)$表示状态 s_i，$\pi(i,E)$表示事件 δ_i。

E-CTL* 公式由路径量词和时态算子组成。路径量词用来描述标记 Kripke 结构中的分支结构，有两种路径量词：A(指对所有的路径)，E(指对某条路径)。时态算子用来描述路径的属性。有五个时态算子 X、F、G、U、R。在 E-CTL* 中有两种类型的公式：状态公式、路径公式。设 AP 是某个原子命题集，状态公式定义如下：

- 如果 $p\in AP$，那么 p 是状态公式。
- 如果 f 和 g 是状态公式，那么$\neg f$、$f\vee g$、$f\wedge g$ 是状态公式。
- 如果 f 是状态公式，那么 Ef、Af 是状态公式。

除了上述三条规则，还需三条额外的规则来描述路径公式：

- 如果 $\delta\in\Delta$，那么 δ 是路径公式。
- 如果 f 是状态公式，那么 f 是路径公式。
- 如果 f 和 g 是路径公式，那么$\neg f$、$f\vee g$、$f\wedge g$、Xf、Ff、Gf、fUg、fRg 是路径公式。

E-CTL* 是由上述规则定义的状态公式的集合。我们在标记 Kripke 结构上定义 E-CTL* 的语义。如果 f 是状态公式，记号 $\kappa,s\models f$ 表示 f 在 κ 中的状态 s 处成立。类似地，如果 f 是路径公式，$\kappa,\pi\models f$ 表示 f 在路径 π 上成立。关系$\models$定义如下(f_1、f_2 是状态公式，g_1、g_2 是路径公式)：

- $\kappa,s\models p$ 当且仅当 $p\in L(s)$；
- $\kappa,s\models\neg f_1$ 当且仅当 $\kappa,s\not\models f_1$；
- $\kappa,s\models f_1\vee f_2$ 当且仅当 $\kappa,s\models f_1$ 或者 $\kappa,s\models f_2$；
- $\kappa,s\models f_1\wedge f_2$ 当且仅当 $\kappa,s\models f_1$ 且 $\kappa,s\models f_2$；
- $\kappa,s\models Eg_1$ 当且仅当存在一条从 s 出发的路径 π 使得 $\kappa,\pi\models g1$；
- $\kappa,s\models Ag_1$ 当且仅当对每条从出发的路径满足，$\kappa,\pi\models g1$；
- $\kappa,\pi\models\delta$ 当且仅当 $\pi(0,E)=\delta$；
- $\kappa,\pi\models f_1$ 当且仅当 $\kappa,\pi(0)\models f_1$；
- $\kappa,\pi\models\neg g_1$ 当且仅当 $\kappa,\pi\not\models g_1$；
- $\kappa,\pi\models g_1\vee g_2$ 当且仅当 $\kappa,\pi\models g_1$ 或者 $\kappa,\pi\models g_2$；
- $\kappa,\pi\models g_1\wedge g_2$ 当且仅当 $\kappa,\pi\models g_1$ 且 $\kappa,\pi\models g_2$；
- $\kappa,\pi\models Xg_1$ 当且仅当 $\kappa,\pi^1\models g_1$；
- $\kappa,\pi\models Fg_1$ 当且仅当存在 $k\geqslant 0$ 使得 $\kappa,\pi^k\models g_1$；
- $\kappa,\pi\models Gg_1$ 当且仅当对任一 $i\geqslant 0$，$\kappa,\pi^i\models g_1$；
- $\kappa,\pi\models g_1Ug_2$ 当且仅当存在 $k\geqslant 0$ 使得 $\kappa,\pi^k\models g_2$ 且对

于任一 $0\leqslant j<k$，$\kappa,\pi^j\models g_1$；

• $\kappa,\pi\models g_1Rg_2$ 当且仅当对任一 $j\geqslant 0$，如果对每个 $i<j$，$\kappa,\pi^i\not\models g_1$ 那么 $\kappa,\pi^j\models g_2$。

我们称 E-CTL* 公式 f 在标记 Kripke 结构 κ 中是有效的当且仅当 $\kappa,s_0\models f$。E-CTL* 和 CTL* 相比有如下几个优点：(1)E-CTL* 是基于状态和事件的；(2)E-CTL* 描述属性更加直观和紧凑；(3)E-CTL* 公式有效性判定的复杂性不高于 CTL* 公式。图 1(a)是进入其他国家的一般流程。首先要获得护照(passport)和领事馆(consulate)的签证(visa)，海关检查过后就可以进入别国领土。需要验证的属性是：只有在领事馆拿到签证的人才可以进入该国。基于状态的形式化描述为$\neg(\neg$stamp U incontry)。这种形式化描述有个缺点：持有伪造的签证的人也可以进入该国。如图 1(b) 的流程也满足这个属性，但是不满足如下属性：任何一个国家都不允许伪造(forger)签证的人进入该国。E-CTL* 公式：$\neg(\neg$get visa U incountry)就可以避免这个问题。

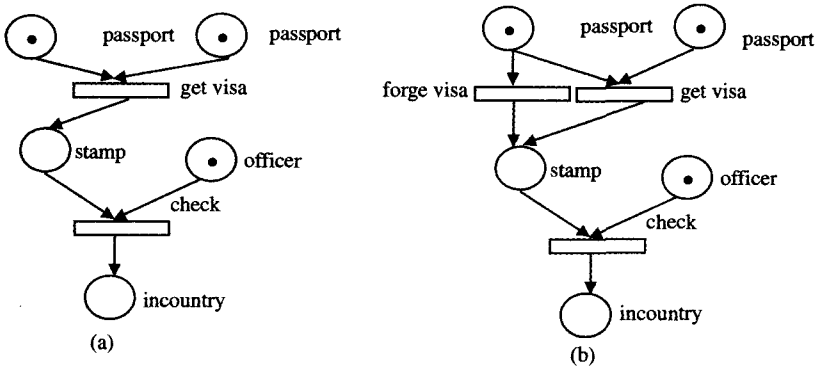

图 1　简单示例

4　工作流逻辑正确性验证

4.1　工作流的同步网模型

文[1]中提出的同步网是基于 Petri 网理论、用于解决工作流管理问题的建模工具。根据抽象层次的不同，同步网分为逻辑层、语义层以及执行层。逻辑层只关心过程中由哪些任务组成和这些任务之间的依赖关系，不关心任务的具体操作内容。语义层是在逻辑层之上使用案例的显性内容来消解冲突，为案例确定实际路径。而执行层运用了对偶的机制，在库所的位置实施管理规则，从而推动过程实例的执行。

定义 4.1(同步器)　库所 $p=(T_1,T_2,(a_1,a_2))$称为同步器，其中 $T_1=\{t_{11},\cdots,t_{1m_1}\}$，$T_2=\{t_{21},\cdots,t_{2m_2}\}$，$1\leqslant a_1\leqslant m_1$，$1\leqslant a_2\leqslant m_2$。

同步器是逻辑层的核心成分，它借助同步论中同步距离的概念，刻画了任务之间的同步关系，其示意如图 2。同步器根据$|T_1|$，$|T_2|$及 a_1,a_2 的取值不同体现为串行、并行、选择等多种同步关系，其组合可以实现多种工作流模式[12]。

定义 4.2　$WN=(P,T;F,K,W,M_{s_0})$称为工作流逻辑，如果

(1)$(P,T;F,K,W,M_0)$是 P/T 网，而且 $\forall p\in P$：$\cdot p\neq\varnothing\wedge p\cdot\neq\varnothing\Rightarrow p=(T_1,T_2,(a_1,a_2))$。

(2)令$<\cdot=\{(t,t')\mid t,t'\in T\wedge\exists p\in P:t\in\cdot P\wedge t'\in P\cdot\}$，则$<\cdot$满足：$<\cdot\cap<\cdot^2=\varnothing$且$(T,<\cdot)$是连通图。

(3)$\forall p\in P:(\cdot p=\varnothing\Rightarrow M_0(p)=1\wedge(\cdot p\neq\varnothing\Rightarrow M_0(p)=0)$。

工作流逻辑表现为可能存在冲突的 Petri 网系统。从上述定义不难看出，WN 是无环有限网，且 $\forall t\in T:\cdot t\neq\varnothing\wedge t\cdot\neq\varnothing$，所以必有库所 $p,p\cdot=\varnothing$。这种库所称为终点库所，终点库所不一定唯一。

定义 4.3　工作流逻辑具有畅通性，如果入口库所中的托肯经传递、复制及合并(同步器)能流动到一个终点库所，而且与冲突消解方案无关。

为了检验逻辑的畅通性，文[1]中采用图形化简的方法，提出了 8 条化简规则，并且证明这些规则不改变过程原来的畅通性。

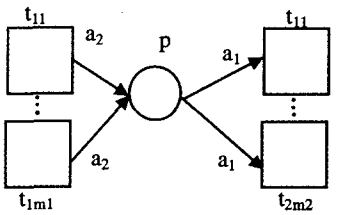

图 2　同步器

定理 4.4　工作流逻辑如果使用文[1]中 8 条规则能够化简为 1 个库所，则认为是畅通的。

定理 4.4 只是描述了一个充分条件，下面我们给出一个充分必要条件。为简单起见，我们假定 WN 只有一个终点库所 o。标识 M 称为终点标识，如果 M 满足 $M(o)=0$ 且对于其它任一 $p\in P,M(p)=0$。

定义 4.5(WN 的变迁规则)　$WN=(P,T;F,K,W,M_0)$的变迁规则是对 P/T 系统的变迁规则加上以下两点约束：

(1)每个变迁至多只能发生一次。

(2)同步器 $p=(T_1,T_2,(a_1,a_2))$只有在 $M(p)=a_1\times a_2$ 时才能同步授权给 T_2 中变迁发生权。

注意，当 T_2 中变迁从同步器获得发生权以后并不必同步发生。标记序列 $M_1,\cdots,M_n$ 称为标记路径当且仅当存在变迁序列 $t_1,\cdots,t_{n-1}$，使得对每个 $1\leqslant i<n,M_i[t_i>M_{i+1}$，且 t_i 的引发符合上述变迁规则。

定义 4.6　$WN=(P,T;F,K,W,M_0)$的可达标识集$[M_0>$是满足下列条件的最小集合：

(1)$M_0\in[M_0>$；

(2)若有 $M'\in[M_0>$且存在 $t\in T$ 满足 WN 变迁规则，使得 $M'[t>M$，则 $M\in[M_0>$。

事实上，我们可以把 WN 的可达标识集看成一个 LKS 结构，下面的定义给出了映射方法。

定义 4.7 设$[M_0>$是 $WN=(P,T;F,K,W,M_0)$的可达标识集，我们以下述方式构造五元组(S,s_0,Δ,L,R)：

(1)$s_0=M_0$ 且如果 $M\in[M_0>$，那么 $M\in S$。

(2)$\Delta=T\cup\{\tau,\theta\}$。

(3)若有 $M'\in[M_0>$且存在 $t\in T$ 满足 WN 变迁规则，使得 $M'[t>M$，则$(M',t,M)\in R$。若 $M'\in[M_0>$且 M'不是终点标识，同时不存在 $t\in T$ 在 M 下有发生权，则$(M',\tau,M')\in R$。若 $M'\in[M_0>$且 M'是终点标识，则$(M',\theta,M')\in R$。

(4)对每个 $M\in[M_0>$，$L(M)=\{a_i\mid a_i$，描述了这样的事实：在标识 M 下，在库所 p_i 中有 $M(p_i)$个托肯$\}$。

我们称 $WN=(P,T;F,K,W,M_0)$满足 E-CTL* 公式 α 当且仅当 WN 的可达标识集按上述方式得到的标记 Kripke 结构满足 α。

4.2 工作流逻辑畅通性 E-CTL* 描述

定理 4.8 设 $WN=(P,T;F,K,W,M_0)$为一工作流逻辑，具有畅通性当且仅当 WN 满足时态逻辑公式 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\},M(p)=0))$。

证明：

$(\Rightarrow)$。WN 可达标识集中的每一条从初始标识开始的标识路径都对应了入口库所的托肯传递、复制及合并的过程。由工作流逻辑的畅通性定义，入口库所的托肯经传递、复制及合并能流到一个终点库所。因此每一条从初始标识开始的标识路径都会到达终点标识。由定义 4.7，WN 可达标识集中的每一条从初始标识开始的标识路径都和其标记 Kripke 结构中某条路径相对应，即在标记 Kripke 结构中每一条从初始状态开始的路径都能到达终点标识对应的状态。由终点标识的定义，终点标识 M 满足 $M(o)=1$ 且对其它任一库所 p，$M(p)=0$。因此 WN 满足时态逻辑公式 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\},M(p)=0))$。

$(\Leftarrow)$。据 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\},M(p)=0))$语义解释，在 WN 的标记 Kripke 结构中从初始状态出发都能到达状态 M，M 满足 $M(o)=1$ 且对其它任一库所 p，$M(p)=0$，所以 M 就是终点标识。由定义 4.7，WN 标记 Kripke 结构中从初始状态出发的路径都对应了可达标识集中从初始标识开始的标识路径，因此入口库所的托肯经传递、复制及合并能流到一个终点库所。

4.3 工作流逻辑变迁规则 E-CTL* 描述

回顾一下，工作流逻辑网的变迁规则是对 P/T 系统的变迁规则加上以下两个约束：

(1)每个变迁至多只能发生一次；

(2)同步器 $p=(T_1,T_2,(a_1,a_2))$只有在 $M(p)=a_1\times a_2$ 时才能同步授权给 T_2 中变迁发生权。

对约束(1)，假设我们需要验证变迁 t 是否至多只能发生一次，即变迁 t 引发后将不能再引发，这个性质用 E-CTL* 可以描述为 $AG(t\rightarrow XG(\neg t))$。

对约束(2)，不失一般性，我们假设同步器 $p=(T_1,T_2,(a_1,a_2))$，且 $T_1=\{t_1,\cdots,t_n\}$中的变迁不会在工作流逻辑的其它地方出现。这时约束(2) 可以用 E-CTL* 描述为 $AG((t_1\vee\cdots\vee t_n)\rightarrow F(M(p)=a_1\times a_2))$。由于 T_2 中变迁发生权是同步授权的，这就使得 T_2 中变迁在满足 $M'(p)=a_1\times a_2$ 的标识 M'下才是可授权的。因此 T_1 变迁的引发必然会导致标识 $M(M$ 满足 $M(p)=a_1\times a_2)$的出现。这正是 $AG((t_1\vee\cdots\vee t_n)\rightarrow F(M(p)=a_1\times a_2))$的语义解释。

4.4 一般性质需求的 E-CTL* 描述

这一小节我们从时序作用范围和使用条件等方面利用 E-CTL* 精确地描述工作流逻辑的性质需求。表 1 总结了一般性质的 E-CTL* 描述。

5 过程逻辑变动分析

过程发生变动的情况各异。为降低问题的复杂性，本文将过程变动看作基本变动的组合。通过对这些基本变动操作进行研究，从而获得过程变动的性质。

定义 5.1 过程逻辑变动具有正确性，如果该变动能够保持工作流逻辑的畅通性。

表 1 一些性质的 E-CTL* 描述

名称	E-CTL* 描述	说明
不变式	$AG\alpha$	刻画工作流逻辑生命周期中不变的状态
无死锁	$AG\neg\tau$	在从可达标识集到标记 Kripke 结构的转换过层中，对于除了终点标识的死锁状态 s，我们增加了一个转换关系 $R(s,\tau,s)$。
没有发生	$G(t)$	刻画在工作流网逻辑中变迁 t 不会引发
最终发生	Ft	刻画在工作流网逻辑中变迁 t 会被引发
Until	$\alpha U\beta$	某变迁引发使 α 为真直到另一变迁引发使 β 为真
互斥发生	$G(t_1\wedge t_2)$	变迁 t_1，t_2 不能同时引发
同时发生	$G(t_1\rightarrow t_2\wedge t_2\rightarrow t_1)$	变迁 t_1，t_2 同时引发
因果	$G(t_1\rightarrow Ft_2)$	变迁 t_1 的引发会导致变迁 t_2 的引发
单因多果	$G(t_1\rightarrow(Ft_2\wedge Ft_3))$	变迁 t_1 的引发会导致变迁 t_2，t_3 蹦引发
多因单果	$G((t_1\wedge t_2)\rightarrow Ft_3)$	变迁 t_1，t_2 的引发会导致变迁 t_3 的引发

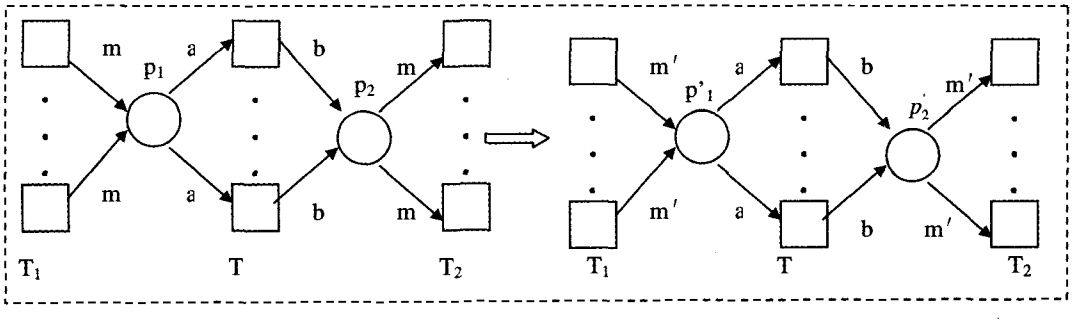

图 3 变动 1 改变同步关系

5.1 基本变动过程

变动 1 改变同步关系

图 3 表示通过同时改变相邻两个同步器的两端弧的加权值(由 m 变成了 m'),以改变中间活动集合的同步关系。这种改变只会引起 WN 的可达标识集的局部改变,即对于可达标识集中的标识路径:$M_1,\cdots,M_n$,其中 M_1 满足 $M_1(p_1)=a\times m, M_1(p_2)=0, M_n$,满足 $M_n(p_1)=0, M_n(p_2)=b\times m$,产生影响,得到一个新的标识路径:$M'_1,\cdots,M'_h$,其中 M'_1 满足 $M'_1(p'_1)=a\times m', M'_1(p'_2)=0, M'_h$,满足 $M'_h(p'_1)=0, M'_h(p'_2)=b\times m'$。因此改变相邻两个同步器的两端弧的加权值产生的影响可以简单地描述为用一段标识路径替换另一段标识路径而已,不会影响终点库所标识的可达性。由 E-CTL* 的语义解释,新的 WN 也会满足时态逻辑公式 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\}, M(p)=0))$。

变动 2　增加新活动

图 4 表示在两个同步器之间添加新活动(t_n),以增加可能的分支。这种改变方式得到的新的 WN 的可达标识集和改变之前的 WN 的可达标识集状态转换关系一样,因此新的 WN 也具有畅通性。

变动 3　增加新活动

图 5 表示通过改变已同步器和增加新的同步器,在两个活动之间插入新的活动(t_n)。这种改变对 WN 的可达标识集的影响是全部的。这种影响可以看成由下面 3 个过程组成:(1)对于可达标识集中的每个标识 M,将其改变为 M',这里 M' 满足 $\forall p\in P, M(p)=M'(p), M'(p_n)=0$,得到一个新的可达标识集。(2)对于新的可达标识集中的标识路径 $M_1,\cdots,M_n$,其中 M_1 满足 $M_1(p_1)=1, M_n$ 满足 $M_n(p_2)=1$。我们在 $M_1,\cdots,M_n$ 插入一个新的标识 M',假设插在 M_i 和 M_{i+1} 之间,那么 M' 满足 $M'(p_n)=1, \forall p\in P, M'(p)=M_i(p)$。(3) 对所有的 $m>j>i$,令 $M_j(p_n)=1$。不难发现,这种改变不会影响终点库所标识的可达性。由 E-CTL* 的语义解释,新的 WN 也会满足时态逻辑公式 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\}, M(p)=0))$。

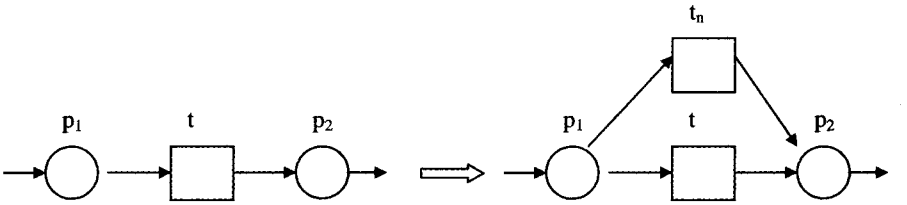

图 4　变动 2 增加新活动

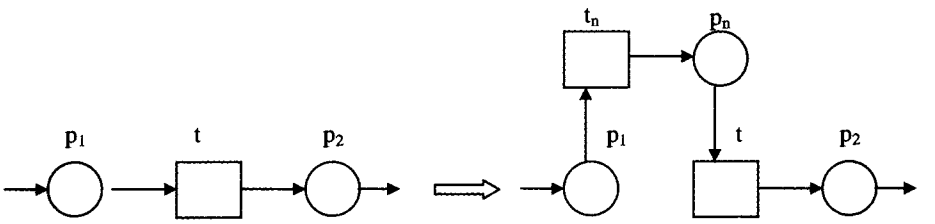

图 5　变动 3 增加新活动

变动 4　增加新活动

图 6 表示在两个同步器之间添加新活动(t_n),以增加可能的分支。这种改变方式得到的新的 WN 的可达标识集和改变之前的 WN 的可达标识集状态转换关系一样,因此新的 WN 也具有畅通性。

变动 5　移除旧活动

图 7 表示移除两个同步器中间的活动,这种改变方式得到新的 WN 的可达标识集可以看成由下面三个过程组成:(1)对可达标识集中的每一个标识 M,如果 $M(p_1)=0, M(p_2)=0$,那么令 $M(p)=0$,同时剔除 $M(p_1), M(p_2)$;(2)对可达标识集中的每一个标识 M,如果 $M(p_1)=1$ 或者 $M(p_2)=1$,那么令 $M(p)=1$,同时剔除 $M(p_1), M(p_2)$;(3)上述两个过程完成后,如果新的可达标识集中存在标识序列 $M_1,\cdots, M_n, M_{n+1}$ 使得 $M_1=M_n$,那么剔除 M_n,使得 M_{n+1} 成为 M_{n-1} 的后继。不难发现,这种改变不会影响终点库所标识的可达性。由 E-CTL* 的语义解释,新的 WN 也会满足时态逻辑公式 $AF(M(o)=1\wedge(\forall p\in P\backslash\{o\}, M(p)=0))$。

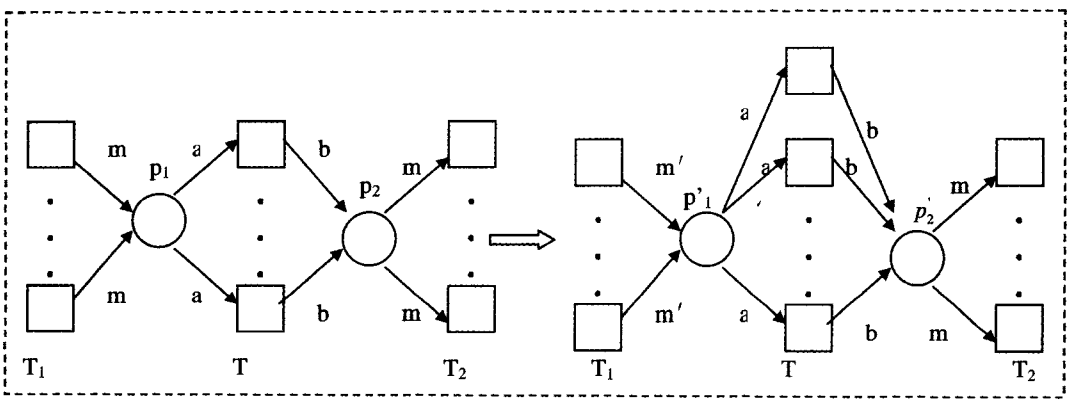

图 6　变动 4 增加新活动

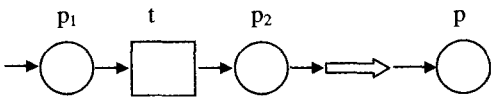

图 7　变动 5 移除旧活动

变动 6　移除旧活动

图 8 表示移除两个同步器中间的活动。这种改变方式得到的新的 WN 的可达标识集和改变之前的 WN 的可达标识集状态转换关系一样，因此新的 WN 也具有畅通性。

5.2　过程变动相关工作

目前工作流领域存在若干种处理变动的方法，其中绝大部分依赖于各自的工作流模型，从而受到模型表达能力的影响。文[13]提出了工作流变动的框架，针对工作流的若干方面(过程、信息、操作等)描述了变动的基本操作，并通过工作流结构和执行状态给出了变动的正确性约束。由于文中变动所依赖的模型类似于活动图，因此活动之间的关系刻画起来比较复杂，在定义修改操作时没有明确给出活动之间的关系。文[14]提出了一种工作流演化方法，采用特定 Petri 网作为描述工作流的模型，提出变动区域作为变动的关注点，这样变动就体现在子网之间的替换，并且通过工作流执行序列的合法性描述了变动的正确性。方法中的变动区域很直观，但是不能通过程序自动产生。文[15]运用变动区域的思想，明确区分了静态、动态和最小变动区域，并给出了自动产生变动区域的算法。但是算法的复杂度相比本文中给出的方法较高，因为本文中对变动的描述直接反映了变动区域。本文提出的方法建立在同步网基础之上，而同步网基于传统的 Petri 网理论，具有局部确定性、形式化的描述和分析方法；同时同步网的提出来源于实践，层次化的方法使得对过程的表达能力很强。因此，相比其他方法，本文的方法更简单，并且可以由程序自动完成。

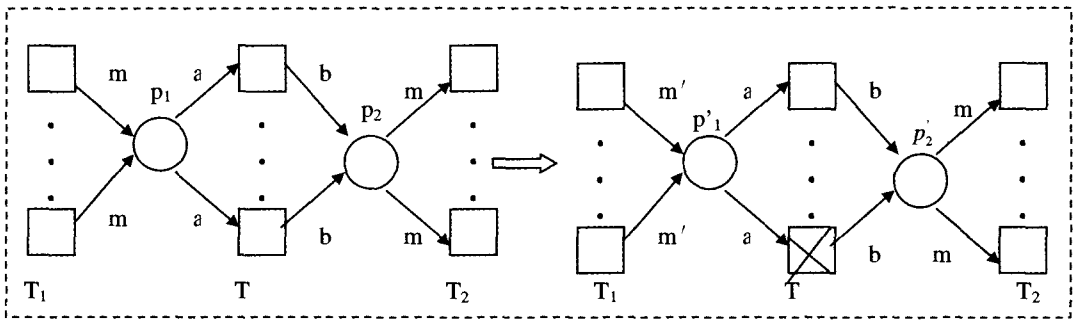

图 8　变动 6 移除旧活动

总结　工作流模型的分析是工作流技术中重要的研究课题之一。工作流逻辑的畅通性一般是通过显式构造其可达状态空间来达到验证的目的，因此状态爆炸问题是验证的主要困难。本文扩展了传统的时态逻辑 CTL*，提出了基于状态和事件的时态逻辑 E-CTL*。E-CTL* 能够非常紧凑和直觉地描述工作流逻辑的畅通性要求，同时验证复杂性不高于 CTL*，因此可以使用符号化的模型检测工具自动完成畅通性的验证。工作流的过程变动问题由来已久，在实际应用中用户不断提出变动的需求，如何使用形式化的方法刻画过程变动，并对过程变动进行有效的管理是很多学者研究的问题。以工作流的同步网模型为基础，分类描述了基本变动操作，同时证明了这些变动的性质不改变工作流逻辑的畅通性。本文中的方法还存在不足，如对数据部分的变动管理没有涉及等。下一步的工作包括：讨论过程中语义部分变动的管理，同时将过程变动的描述和管理加入已经实现的工作流管理系统中。

参 考 文 献

1　Yuan Chong Yi. Principals and Application of Petri Nets. Publishing House of Electronics Industry, 2005. 213～258

2　Lin Huiming, Zhang Wenhui. Model checking: theories, techniques, and applications. Acta Electronic Singa, 2002,30(12A): 1907～1912

3　Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proc. of Logic of Programs Workshop, Yorktown Heights, 1981

4　Clarke E M, Grumberg O, Peled D. Model Checking. MIT Press, 1999

5　McMillan K L. Symbolic model checking. Kluwer Academic Publishers, 1993

6　Burch J R, Clarke E M, McMillan K L. Symbolic model checking: 1020 states and beyond. Information and Computation, 1992,98:142～170

7　Holzmann G. The Model Checker Spin. IEEE Trans on Software Engineering, 1997, 23(5): 279～295

8　Ben-Ari M, Manna Z, Pnueli A. The temporal logic of branching time. Acta Information, 1983,20: 207～226

9　Pnueli A. A temporal logic of concurrent programs. Theoretical Computer Science,13: 45～60

10　van der Aalst W P M. Dealing with Workflow Change: Identification of Issues and Solutions. Comput Syst Sci & Eng, 2000, 5: 267～276

11　Chaki S, Clarke E M, Quaknine J, et al. Concurrent software verification with states, events, and deadlock. Formal Aspects of Computing, 2004,17(4):461～483

12　van der Aalst W M P, ter Hofstede A H M, Kiepuszewski B, et al. Workflow Patterns. BETA Working Paper Series, Eindhoven University of Technology, Eindhoven, 2000

13　Reichert M, Dadam P. ADEPTflex-Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems ,1998,10(2):93～129

14　Ellis C, Keddara K, Rozenberg G. Dynamic Change Within Workflow Systems. In: Proc. Conf. on Organizational Computing Systems, Milpitas, USA, August 1995

15　van der Aalst W M P. Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. Information Systems Frontiers, 2001, 3(3): 297～317