CBR: 一种支持工作流过程语义验证的组件级化简方法

周建涛[1,2], 史美林[1], 叶新铭[2]
(1. 清华大学 计算机科学与技术系, 北京 100084; 2. 内蒙古大学计算机学院, 内蒙古呼和浩特 010021)

摘 要: 目前的工作流过程验证方法多数侧重控制流特性验证, 与数据或资源信息相结合的验证尚未得到很好发展. 然而, 业务过程的目标实现依赖于控制流、数据流和资源三维基本观点的协作. 保证过程的目标实现需要验证这三维观点协作的正确性, 称为语义验证. 该文侧重讨论大型、灵活的过程语义验证问题. 首先, 形式化定义过程模型, 综合体现三维基本观点, 表达完整的过程语义, 并使用基于组件的思想, 具有一定可扩展性和灵活性. 然后, 通过探讨组件间的协作逻辑, 提出基于组件的化简方法(Component Based Reduction, CBR)和化简规则, 为过程语义验证提供有力支持.

关键词: 工作流; 验证; 化简; 组件; Petri 网
中图分类号: TP311; TP391 文献标识码: A 文章编号: 0372-2112 (2005) 06-1060-06

CBR: A Component-Based Reduction Method for Semantics Verification of Workflow Processes

ZHOU Jian-tao[1,2], SHI Mei-lin[1], YE Xin-ming[2]
(1. *Department of Computer Science, Tsinghua University, Beijing* 100084, *China*;
2. *College of Computer Science, Inner Mongolia University, Huhhot, Neimenggu* 010021, *China*)

Abstract: Verification methods of workflow processes always focus on the control flow, while only a few developments have been done on the verification combining control flow with data and resource information. However achievements of business goals rely on cooperation of three basic dimensions of workflow, called control flow, data flow and resource. Semantics verification is to ensure cooperation correctness of the three dimensions. This paper endeavors semantics verification of large-scale and flexible workflow processes. It firstly presents a formal process model, not only integrating the basic three dimensions for semantics expression, but also using concept of component to facilitate scalability and flexibility of processes. By focusing collaborative logic between components, it then discusses component-based reduction technique and rules, strongly supporting semantics verification of processes.

Key words: workflow; verification; reduction; Petri net

1 过程验证的必要性、层次和方法

工作流技术作为计算机支持的协同工作(Computer Supported Cooperative Work, CSCW)领域的一项重要应用, 在企业过程管理中发挥了重大作用. 要实现工作流的管理功能, 需要对业务过程进行建模, 结果称为过程定义[1]. 过程建模是一项非常复杂且容易出错的工作. 首先, 跨领域和跨企业的业务过程规模较大, 其间的结构连接和交互通信比较复杂; 其次, 企业对过程灵活性、动态性和自适应性的要求逐渐增加[2]. 这些都为过程的抽象和描述带来了一定困难, 因而导致大型、具有灵活性的过程定义中经常会出现差错或存在与过程目标偏离的情况. 仅凭设计者的经验避免差错或操作者的反复试验来发现差错的方法存在效率低、代价高、开发周期长等缺点. 为了保证过程的正确执行和方便管理, 形式化过程验证成为工作流建模的重要环节.

过程验证可分为三个层次. 最基本的是语法检查, 目的是确定过程定义是否符合描述语言或模型的语法约束和规则, 即是否具有语法正确性, 这是工作流得以进一步执行和分析的基础. 进一步的验证是控制流验证, 也称结构验证, 目的是考察过程的结构正确性, 检测死锁、无同步、无终止活动或无初始活动等结构冲突[3~9]. 这方面的验证发展了一些有用的方法, 包括: 基于 Petri 网的方法[3,4], 基于化简技术的方法[5~7], 基于逻辑的方法[8]和基于矩阵计算的方法[9]等.

结构正确性只反映了过程定义正确性的一个侧面. 通过研究工作流过程的元模型[1,10]可以看出, 过程完整的语义描述包括三维基本观点: 控制流、数据流和资源. 控制流观点反映组成过程的活动(或子过程)之间的连接逻辑, 即"活动-活动"关系. 数据流观点侧重描述各个活动操作的数据对象, 即"活动-数据"关系. 资源观点涉及人力资源和设备资源, 侧重反映"活动-角色(或执行者)"之间的关系. 虽然一个工作流过

收稿日期: 2004-08-03; 修回日期: 2004-11-09
基金项目: 国家自然科学基金(No. 60073011); 国家高技术研究发展 863 计划项目(No. 2001AA113150)

程以控制流作为协同和调度的核心, 但是只有结合数据流和资源的共同作用才能实现其业务目标[1]. 因此, 仅用结构验证方法保证控制流方面的正确性尚不足以保证过程的目标实现, 我们需要验证控制流、数据流和资源这三维基本观点协作的正确性, 称为语义正确性. 保证语义正确性的验证称为语义验证.

语义验证研究还有待发展[11]. 一个基本问题是目前的过程建模技术在描述数据流和资源方面落后于语义验证的需求. 现有验证方法中, 多数模型侧重描述过程的控制流[3,5~9], 仅有少数同时描述数据流或资源[4,12], 而且, 或者使用非形式化模型, 或者将数据流、资源的描述与控制流相分离. 这种分离描述的方法在过程中同时呈现两条或多条"流", 不仅增加了处理复杂度, 更重要的是无法体现数据传递与资源分配沿控制流进行的本质. 我们提出的形式化模型将针对这一点进行改进, 综合描述过程的三维信息, 为语义验证提供模型基础.

另外, 由于现代企业对业务过程复杂性和规模性有较高要求, 因此过程模型应具有一定灵活性和重用性, 以保证最大限度地利用已有建模结果, 避免每个过程都"从头做起"的建模方法. 基于组件的模型非常适合解决这个问题[13]. 组件的概念来源于软件工程领域, 是具有一定表示形式和协作方法的实体, 它可以按相对灵活的粒度表示过程片断, 有助于快速集成和建立大型系统[14]. 本文提出的形式化模型为工作流过程组件提供具体的描述方法, 支持大型、具有灵活性的过程定义.

大型过程定义的形式化验证还面临两个突出的问题. 首先是状态爆炸问题. 事实上, 即使是一个中等规模的系统, 构造其完整的可达状态空间也非常困难. 因此, 基于化简技术的验证方法[5~7,15,16]是一种较好的选择. 这类方法使用基于特性保持(property-preserving)原则的化简规则, 逐步缩小模型规模. 然后, 在小规模的模型中, 检测模型特性, 得出验证结论. 其次是验证效率问题. 与以上描述的避免"从头做起"的建模问题类似, 验证也希望能够利用积累的验证结果. 在基于组件的大型过程定义中, 组件的内部正确性是可以重用的结果, 此时, 组件与过程其他部分之间的连接情况成为验证的重点. 综合考虑这两个问题, 我们将讨论基于组件的化简方法, 通过探讨组件间在控制流、数据流和资源三维基本观点上的协作逻辑, 提出特性保持的化简规则, 支持过程的语义验证.

2　面向语义验证的过程模型

2.1　过程语义的模型描述

本节定义描述控制流、数据流和资源三维基本观点的工作流过程模型, 并解释其动态语义.

基本 Petri 网是一个五元组(P, T, F, W, M_0)[15], 其中: P 是库所的有限集合; T 是变迁的有限集合, 且 $P\cap T=\varnothing$, $P\cup T\neq\varnothing$; $F\subseteq(P\times T)\cup(T\times P)$ 是弧集合; $W: F\to\{1, 2, 3\cdots\}$ 是权函数; $M_0: P\to\{0, 1, 2, 3\cdots\}$ 是托肯在库所中的初始分布, 称为初始标识. 一般地, 在工作流建模中[17], 变迁表示过程中的活动, 网系统的标识表示工作流系统的状态, 弧表示控制流.

为了更细致地模拟工作流过程, 首先将上述模型做如下扩展: (1) 由于工作流活动可以分为原子活动与子过程, 以及一些起控制作用的内部活动, 因此将变迁集合 T 相应地划分为原子变迁集合 T_a、子网变迁集合 T_p 和内部变迁集合 T_τ. (2) 为了增强过程的流控能力, 允许使用禁止弧集合 F°.

接着, 为了描述数据和资源, 继续做如下扩展: (3) 将数据和资源的类型映射为托肯类型集合 C, 其中 Cd 为数据类型, Cr 为资源类型. 相应地, 网标识由描述托肯分布的正整数向量扩展为描述数据和资源分布的集合向量. (4) 弧上的权函数 W 扩展为约束数据和资源流动的弧表达函数 Exp. 该函数是托肯的数量($N=\{1, 2, \cdots\}$)、类型(C)、以及描述数值、比较和逻辑计算的条件表达式(Con)组成的三维向量. 一个条件表达式 $con\in Con$ 的结果取值为布尔类型, 即: Type(con) = {True, False}. 扩展后的模型定义如下:

定义 1　(三维工作流网, 3DWFN).　设 Σ, K, Ψ 分别表示工作流过程中的活动名、数据对象名和资源名的集合, 则其上一个三维工作流网(Three-Dimension Workflow Net)是一个 7 元组 $3DWFN=\langle P, T, F, C; Lab, Exp, M_0\rangle$:

$P\cap T=\varnothing$; $P\cup T\neq\varnothing$, 且 $\exists\, i, 0\in P$, $\bullet i=\varnothing$, $i^\bullet=\varnothing$;

$T=T_a\cup T_p\cup T_\tau$;

$F\subseteq(P\times T)\cup(T\times P)$, 且 $F^\circ\subseteq F$;

$C=Cd\cup Cr\cup\{\bullet\}$, '$\bullet$' 为缺省值;

$Lab: T\to\Sigma$, $Cd\to K$, $Cr\to\Psi$ 是标记函数;

$Exp: F\to N\times C\times Con$ 是弧表达函数;

$M_0: P\to\{\varnothing, \mu C\}$, μC 是 C 上的多集.

Petri 网的动态行为表现为托肯的迁移. 在基本 Petri 网描述的工作流过程中, 托肯的迁移只表示控制权(或称执行权)在活动之间的交接. 下面, 将通过解释 3DWFN 的动态语义, 说明数据和资源沿控制流进行传递、分配的过程.

(1) 变迁 $t\in T$ 的前置条件: 对 t 的任意一个前置库所 $s\in\bullet t$ 来说, 如果(s, t)是非禁止弧, 即: $(s, t)\in F/F^\circ$, 那么 $Pre(s, t)=Exp(s, t)$ 表示 s 到 t 的点火条件, $Pre(t)=\{s\in\bullet t\mid Exp(s, t)\}$ 表示这一类前置条件的集合; 而当(s, t)是禁止弧时, 即: $(s, t)\in F^\circ$, $Prev(s, t)=Exp(s, t)$ 表示 s 到 t 的控制条件.

(2) 变迁 $t\in T$ 的后置条件: 对 t 的任意一个后置库所 $s\in t^\bullet$ 来说, $Post(t, s)=Exp(t, s)$ 表示 s 对 t 的后置条件, $Post(t)=\{s\in t^\bullet\mid Exp(t, s)\}$ 表示这些后置条件的集合.

(3) 变迁 t 在标识 M 下是使能(enable)的, 对 t 的任意一个前置库所 $s\in\bullet t$ 来说, 当且仅当:

① $(s, t)\in F/F^\circ$: $M(s)\geqslant(Pre(s, t)\ \Sigma N\times C)\ \wedge$ Type(con) = True,

② $(s, t)\in F^\circ$: $M(s)<(Prev(s, t)\ \Sigma N\times C)\ \vee$ Type(con) = False,

其中, $Pre(s, t)\ \Sigma N\times C$ 表示 $Pre(s, t)$ 中描述托肯数量和类型的部分, con 表示其中的条件表达式.

(4) 变迁 t 点火(fire)后产生新标识 $M'=M-(Pre(t)\ \Sigma N\times C)+(Post(t)\ \Sigma N\times C)$, 记为 $Mt>M'$ 或 $M\xrightarrow{t}M'$.

(5) M' 称为是从 M 可达的, 当存在一个点火序列 $\sigma=t_1t_2\cdots t_n$, 使 $M\xrightarrow{t_1}M_1\xrightarrow{t_2}\cdots M_{n-1}\xrightarrow{t_n}M'$, 记为 $M\xrightarrow{\sigma}M'$; 并将从 M 可达的所有标识集合记为 $[M]$.

如上, 使能的变迁按点火条件中托肯数量和类型的要求进行点火, 消耗前置库所中的数据和资源, 并根据相应的后置条件向后置库所中添加数据和资源. 这样, 3DWFN 的执行就有效地描述了工作流过程中数据和资源沿控制流的流动.

2.2 过程组件的模型描述

对工作流过程的其他部分而言, 组件是功能和表示形式相对独立的过程片断, 可以忽略其内部实现, 只考虑输入、输出端口与其他部分的交互[16, 18].

定义 2 (工作流过程组件, 3DWFC). 设 $\Gamma=\langle P^{\Gamma}, T^{\Gamma}, F^{\Gamma}, C; Lab, Exp, M_0^{\Gamma}\rangle$ 是一个 3DWFN 网, 若: $\exists I, O\subseteq P^{\Gamma}\wedge I\cap O=\varnothing\wedge\forall p\in I\cup O: {}^{\bullet}T^{\Gamma}\cap O=\varnothing\wedge T^{\Gamma\bullet}\cap I=\varnothing$, 则称 Γ 为一个工作流过程组件 3DWFC (Three Dimension Workflow Component), 表示为 $\Gamma=(P^c, T^c, F^c, I, O, M_0^c)$. 其中, P^c 称为内部库所, I, O 分别称为输入、输出端口, $P^c\cup I\cup O=P^{\Gamma}$. 若 Γ 是另一个 3DWFN 网 PN 的组件, 则表示为 $\Gamma\in Cmpt(PN)$.

以图 1 为例, 3DWFC 可表示为一个矩形, 如组件 A 和 B. 输入端口为内有指向组件内部箭头的库所, 如端口 1、2、4、5; 输出端口为内有指向组件外部的箭头的库所, 如端口 3、6、7. 另外, 下文中, 除明确描述的个数外, 为图示简化起见且不失一般性, 用并行的 2 个同向端口表示可能的 n ($\geqslant 2$) 个端口, 并行的 2 个箭头表示可能的 n($\geqslant 2$) 条输入/输出弧, 如图 1 中端口 1、2 及它们的输入弧. 并且, 将同一组件的输入(或输出)端口彼此称为同侧端口, 如组件 A 的输入端口 1、2、4; 同理, 输入和输出端口之间互为异侧端口, 如组件 A 的输入端口 1、2、4 和输出端口 3.

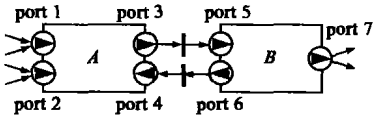

图 1 组件模型图例

3 CBR: 面向语义验证的组件级化简方法

3.1 CBR 方法和前提

由于 Petri 网的基本化简技术包括库所融合、变迁融合和子网融合, 工作流过程组件是 3DWFN 中的子网, 因此本节讨论的化简方法基于子网融合技术. 该方法称为 CBR (Component Based Reduction): 一个表示工作流过程的 3DWFN 网 $PN=\langle P, T, F, C; Lab, Exp, M_0\rangle$, 有组件 $\Gamma=(P^c, T^c, F^c, I, O, M_0^c)$, 且 $\Gamma\in Cmpt(PN)$, 在组件具有内部正确性的前提下, 将 Γ 化简为更小规模的子网 Γ', 得到 $PN'=(P', T', F', C; Lab, Exp, M'0)$, 记为: $PN\xrightarrow{R(\Gamma)}PN'$.

要想让该化简过程是特性保持的, 就必须要求 Γ' 与 Γ 具有相同的对外连接方式和行为.

定义 3 (协作逻辑) 组件 Γ 与 PN 的连接方式和行为构成了组件对外的协作逻辑, 其中:

Γ 与 PN 之间的连接方式称为静态逻辑 $SL=\langle F_I, F_o\rangle$, 其中 $F_I=\{(t, p)\mid p\in I, t\in {}^{\bullet}p\}$ 是 PN 与 Γ 的输入端口 I 之间的连接弧集合, $F_o=\{(p, t)\mid p\in O, t\in p^{\bullet}\}$ 是 Γ 的输出端口 O 与 PN 之间的连接弧集合.

Γ 在 PN 中的行为称为动态逻辑 $DL=\langle M_I, M_o\rangle$, 其中 $M_I=\{M(p)\mid M\in[M_0], p\in I\}$ 表示在标识 M 下 I 中的托肯分布, $M_o=\{M'(p)\mid M'\in[M], p\in O\}$ 表示在标识 M' 下 O 中的托肯分布, DL 反映了 I、O 之间托肯出现的因果关系, 记为 $M_I\xrightarrow{\Gamma}M_o$.

协作逻辑反映了组件 Γ 与过程 PN 的其他部分之间在控制流、数据流和资源三维观点上的连接情况. 其中, 静态逻辑反映了控制流连接, 动态逻辑关心托肯的分布和迁移, 反映了数据流和资源情况.

协作逻辑的直接表现形式是组件的端口连接和端口中托肯出现的因果关系. 这样, 在化简过程中, 只要 Γ' 与 Γ 具有相同的输入、输出端口, 就可以保持静态逻辑不变; 只要 Γ' 与 Γ 在输入、输出端口上, 具有相同的托肯出现因果关系, 就可以保持动态逻辑不变. 下一小节将基于不同的协作逻辑给出这样的组件级化简规则.

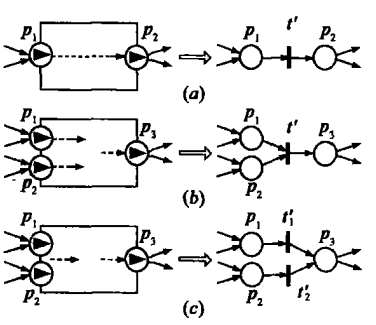

图 2 CR_R1-R3

3.2 CBR 的化简规则

CBR 的化简规则集称为 CRR-SV (Component based Reduction Rule for Semantics Verification), 由 10 条规则组成.

CR_R1: 单输入/单输出组件化简 如图 2(a) 所示, 若组件(唯一的输入端口出现托肯, 将使其唯一的输出端口也出现托肯:

$I=\{p_1\}$, $O=\{p_2\}$, ${M_0}^c=\varnothing\wedge M_I(p_1)\xrightarrow{\Gamma}M_o(p_2)$, 则可将 Γ 除端口以外的部分化简为一个变迁 t': $T'=T/T^c\cup\{t'\}$, $P'=P/P^c$, $F'=F/F^c\cup\{(p_1, t'), (t', p_2)\}$. 输入端口 p_1 到变迁 t' 的弧表达式设置方法如下: 若 p_1 到它的后置库所之间是"或"关系, 则 $\mathrm{Exp}(p_1, t')=\cup\mathrm{Exp}(p_1, p_1^{\bullet})$, 由"$\cup$"符号连接的弧表达式之间也是"或"关系; 若 p_1 到它的后置库所之间是"与"关系, 则 $\mathrm{Exp}(p_1, t')=\cap\mathrm{Exp}(p_1, p_1^{\bullet})$, "$\cap$"符号连接的弧表达式之间为"与"关系. 同理, 变迁 t' 到输出端口 p_2 的弧表达式设置也分这样两种情况, 为简化起见统一记为 $\mathrm{Exp}(t', p_2)=X\mathrm{Exp}({}^{\bullet}p_2, p_2)$.

CR_R2: "与"输入/单输出组件化简 如图 2(b) 所示, 若组件 I 所有的输入口端口都出现托肯, 将导致唯一的输出端口出现托肯, 即多个输入端口之间表现为"与"语义:

$I=\{p_1, p_2\}$, $O=\{p_3\}$, ${M_0}^c=\varnothing\wedge M_I(p_1, p_2)\xrightarrow{\Gamma}M_o(p_3)$, 可将 Γ 除端口以外的部分化简为变迁 t':

$$T'=T/T^c\cup\{t'\}, P'=P/P^c,$$
$$F'=F/F^c\cup\{(p_1, t'), (p_2, t'), (t', p_3)\},$$

化简后在输入端口之间仍然表现"与"结构特性, 弧表达式设置如下:

$$\mathrm{Exp}(p_k, t')=X\mathrm{Exp}(p_k, p_k^{\bullet}), (k=1, 2)$$
$$\mathrm{Exp}(t', p_3)=X\mathrm{Exp}({}^{\bullet}p_3, p_3).$$

CR_R3: "或"输入/单输出组件化简 如图 2(c), 若组件(的某一个或某几个输入端口出现托肯, 将使其唯一的输出端口出现托肯, 即多个输入端口之间表现"或"语义:

$I=\{p_1,p_2\}$, $O=\{p_3\}$, $M_0^c=\varnothing \wedge M_I(p_k)\xrightarrow{\Gamma}M_o(p_2)$,
其中 $k=1,2$, 则将 Γ 除端口以外的部分化简为与输入端口数等量的一组变迁, 每个变迁对应一个输入端口, 表现"或"语义:

$$T'=T/T^c\cup\{t'_1,t'_2\},\ P'=P/P^c,$$

$$F'=F/F^c\cup\{(p_1,t'_1),(p_2,t'_2),(t'_1,p_3),(t'_2,p_3)\},$$

弧表达式设置如下:

$$Exp(p_1,t'_1)=X\ Exp(p_1,p_1^{\bullet}),$$

$$Exp(p_2,t'_2)=X\ Exp(p_2,p_2^{\bullet}),$$

$$Exp(t'_1,p_3)=Exp(t'_2,p_3)=X\ Exp({}^{\bullet}p_3,p_3).$$

CR_R4: 单输入/"与"输出组件化简　如图 3(a), 若组件 I 唯一的输入端口出现托肯, 会导致所有输出端口出现托肯, 这种情况与 CR_R2 中的输入、输出端口之间的关系恰好相反, 可将 Γ 除端口以外的部分化简为变迁 t', 并将 Γ 所有的输出端口设置为 t' 的后置, 以体现其间的"与"语义. 该规则的形式描述和弧表达式设置可参照 CR_R2, 这里不再赘述.

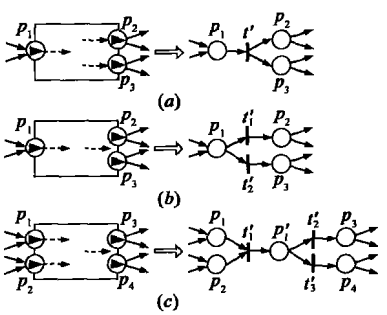

图 3　CR_R4-R6

CR_R5: 单输入/"或"输出组件化简　如图 3(b), 若组件(唯一的输入端口出现托肯, 将导致某一个或某几个输出端口出现托肯, 这种情况与 CR_R3 中的输入、输出端口之间的关系相反, 类似地, 可将 Γ 除端口以外的部分化简为与输出端口数等量的一组变迁, 每个变迁对应一个输出端口, 表现"或"语义. 该规则中弧表达式的设置如下:

$$Exp(p_1,t'_1)\cup Exp(p_1,t'_2)=X\ Exp(p_1,p_1^{\bullet}),$$

$$Exp(t'_1,p_2)=X\ Exp({}^{\bullet}p_2,p_2),$$

$$Exp(t'_2,p_3)=X\ Exp({}^{\bullet}p_3,p_3).$$

CR_R6: "与"输入/"或"输出组件化简　如图 3(c), 若组件(所有的输入端口都出现托肯, 将导致某一个或某几个输出端口出现托肯, 即在多个输入端口之间表现"与"语义, 在多个输出端口之间表现"或语义". 这种情况相当于 CR_R2 中输入端口和 CR_R5 中输出端口情况的结合. 其化简过程中, 结构方面的形式描述参见以上两规则, 弧表达式设置如下:

$$Exp(p_k,t'_1)=X\ Exp(p_k,p_k^{\bullet}),\ (k=1,2)$$

$$Exp(t'_1,p'_1)=Exp(p'_1,t'_2)\cup Exp(p'_1,t'_3),$$

$$Exp(t'_2,p_3)=X\ Exp({}^{\bullet}p_3,p_3),$$

$$Exp(t'_3,p_4)=X\ Exp({}^{\bullet}p_4,p_4).$$

CR_R7: "与"输入/"与"输出组件化简　如图 4(a), 若组件(所有输入端口出现托肯, 将导致所有输出端口出现托肯, 即输入端口之间、输出端口之间都存在"与"语义, 则可用一个变迁替换组件(, 并使该变迁"联结"所有的输入、输出端口. 这种情况相当于 CR_R2 中输入端口和 CR_R4 中输出端口情况的结合, 其结构方面的形式描述和弧表达式设置均可参照上述规则.

CR_R8: "或"输入/"与"输出组件化简　如图 4(b), 若组件(的某一个或某几个输入端口出现托肯, 将导致所有输出端口出现托肯, 即输入端口之间存在"或"语义, 输出端口之间存在"与"语义. 这种情况与 CR_R6 中描述的输入、输出端口的情况相反, 其化简过程的形式描述可参照上述规则.

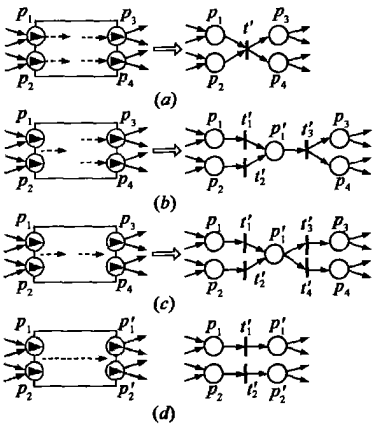

图 4　CR_R7-R10

CR_R9: "或"输入/"或"输出组件化简　若组件(的某一个或某几个输入端口出现托肯, 将使某一个或某几个输出端口也出现托肯, 则说明输入、输出端口之间都存在"或"语义:

$$I=\{p_1,p_2\},\ O=\{p_3,p_4\},\ M_0^c=\varnothing \wedge M_I(p_k)\xrightarrow{\Gamma}M_o(p_n),$$

其中 $k=1,2,\ n=1,2$.

如图 4(c) 所示, 可将组件除端口以外的部分化简为两组变迁和一个库所. 第一组变迁(图中为 $\{t'_1,t'_2\}$)与输入端口对应, 然后将每个输入端口流入的数据和资源在库所 p'_1(汇总, 再重新分配, 经过第二组变迁(图中为 $\{t'_3,t'_4\}$)流向输出端口. 本规则的其他形式描述部分参见 CR_R3, 与库所 p'_1 相连的弧表达式设置满足如下关系:

$$Exp(t'_1,p'_1)\cup Exp(t'_2,p'_1)=Exp(p'_1,t'_3)\cup Exp(p'_1,t'_4).$$

CR_R10: "或"输入对应"或"输出组件化简　本规则中, 组件 Γ 的输入、输出端口之间也都存在"或"语义, 但与 CR_R9 不同的是, 输入到输出端口之间的部分存在对应关系:

$$I=\{p_1,p_2\},\ O=\{p_3,p_4\},\ M_0^c=\varnothing \wedge M_I(p_k)\xrightarrow{\Gamma}M_o(p_k),$$

其中 $k=1,2$.

那么, 可将组件除端口以外的部分化简为一个变迁集合, 其中每个变迁对应一对输入、输出端口, 变迁之间彼此没有联系, 各通过相应的弧表达式将数据和资源从输入端口"传递"到输出端口, 如图 4(d) 所示.

综上, 各化简规则从考虑组件与外部的协作逻辑出发, 将组件化简为一个有较小规模的子网, 并通过弧表达式的设置和运算保证化简后与外部的协作逻辑不变.

3.3　化简规则应用举例

下面, 用电子政务中一个简化的例子说明上一小节给出的化简规则的使用方法. 例. 一个项目计划审批流程如图 5 所示, 图中非阴影部分是基于组件的流程表示, 阴影部分分别为各组件的细化描述. 组件 C_1 表示计划在项目组内起草并审查: 首先, 分析市场调研记录(t_{11}), 然后并行进行撰写调研结果报告(t_{12})和起草项目计划(t_{13}), 再交给项目经理审批(t_{14}), 得出通过与否(t_{15})的结果. 组件 C_2 表示在未获组内审查通过时的项目修改: 分析返回意见(t_{21})并修改计划(t_{22}), 再审批(t_{23}). 组件 C_3 表示组内审查通过后: 判断是否需要通知其他

相关部门(t_{31}),需要就通知(t_{32}),然后同时交给总裁批示(t_{31})并进行项目预算(t_{31}).组件以外的变迁含义如下:t_1:起草计划;t_2:计划备案;t_3,t_4:内部变迁;t_5:审查结果存档.

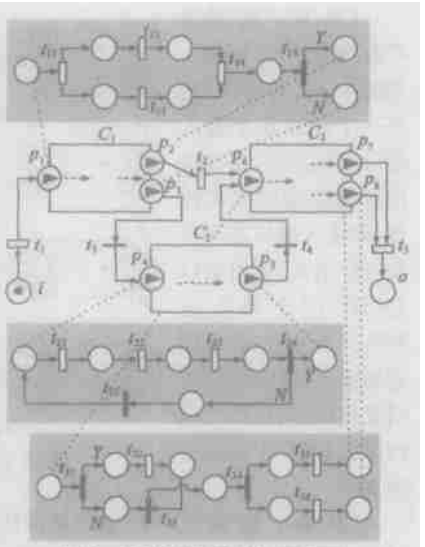

图 5 计划审批流程的 3DWFN

基于组件的流程中各库所(包括组件端口)含义如下:i:开始;p_1:开始起草计划;p_2:组内审查计划通过;p_3:组内审查未通过;p_4:准备修改计划;p_5:修改计划完成;p_6:准备进行上层审查;p_7:合理性审查完成;p_8:项目预算完成;o:结束.

尽管为了完整地描述整个流程,上面介绍了组件的内部含义,然而,在组件级化简中,组件是当作一个相对独立的部分被处理的,我们只关心它与外部的协作逻辑,这正是组件级化简方法的优点.化简后的结果如图 6 所示,虚线圈标明了使用的相应规则.

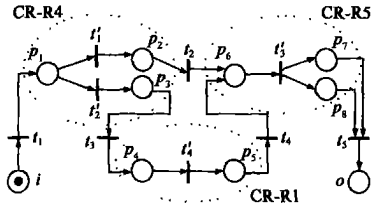

图 6 流程化简结果

需要说明的是,限于篇幅图中省略了各弧表达式的表示及其在化简前后运算的描述.实际中,这部分操作需严格按照化简规则中的描述执行.

如果对组件级化简的结果继续进行相应的原子级化简(其他文章中另有叙述),则最终很容易得到整个系统的验证结果.

4 分析和比较

通过对 CBR 方法中 CRR-SV 的定义及其举例的分析,得出以下一些结论.

定理 1 化简规则集 CRR-SV 在化简过程中,既不减少也不增加过程模型的任何特性.

证明 设 3DWFN 为 $PN=\langle P, T, F, C; Lab, Exp, M_0\rangle$,$\Gamma=(P^c, T^c, F^c, I, O, M_0{}^c)\in Cmpt(PN)$,$PN\xrightarrow{R(\Gamma)}PN'$:

(1) $\forall p\in P^c: M_0(p)=\varnothing$,即:CRR-SV 不化简含初始网标识的部分,因而不减少那些与托肯相关的特性,例如活性、有界性等.

(2) 组件(的内部正确性是化简的必要条件,因此,被化简的是无语义冲突的部分,不减少原过程可能含有的冲突.

(3) 替换组件(的部分与(具有相同的协作逻辑,因而不为过程增加新的特性:

①组件的端口 I、O 在化简中保持不变,保证了静态逻辑不变;

②$\bigcup_{p\in It\in p} X_t\cdot Exp(t, p)$ 和 $\bigcup_{p\in Ot\in p} X_t\cdot Exp(p, t)$ 在化简中保持不变,保证了动态逻辑不变.

定理 1 说明了 CBR 方法中各化简规则在语义验证中的可用性.定理 2 将说明化简规则集的完备性,以及各规则的划分依据.

定理 2 化简规则集 CRR-SV 是从协作逻辑出发的组件级化简的完备集合.

证明 设 3DWFN 为 $PN=\langle P, T, F, C; Lab, Exp, M_0\rangle$,$\Gamma\in Cmpt(PN)$,$\Gamma=(P^c, T^c, F^c, I, O, M_0{}^c)$.

首先,考虑组件异侧端口之间的对应逻辑 $IL=\{LL, JL\}$,LL 表示异侧端口之间存在固定映射,即某一输入端口出现(或不出现)托肯,某一相应输出端口也一定出现(或不出现)托肯,JL 表示异侧端口之间不存在固定映射,在这种情况下:(1) 由于 $|I|\geqslant 1$,$|O|\geqslant 1$,因此设单端口属性 $PSL=\{IS, OS\}$ 描述单一的输入/输出端口,多端口属性 $PML=\{IM, OM\}$ 描述有多个输入/输出端口.

(2) 同侧端口之间存在"与"、"或"语义,因此设输入端口之间的逻辑为 $GIL=\{IAND, IOR\}$,输出端口之间的逻辑为 $GOL=\{OAND, OOR\}$.

将协作逻辑表示为 CL,则:$|CL|=|LL|+|JL|=|LL|+|(IS, OS)|+|PML|\times|GIL|\times|GOL|=10$.即:从协作逻辑出发有 10 种情况,可以描述为 10 条规则,因此,CRR-SV 在此意义上是完备的.

定理 3 在 CBR 方法中,组件结构越复杂,化简的收敛速度越快.

证明 设 3DWFN 为 $PN=\langle P, T, F, C; Lab, Exp, M_0\rangle$,$\Gamma\in Cmpt(PN)$,$\Gamma=(P^c, T^c, F^c, I, O, M_0{}^c)$.并且,$PN\xrightarrow{R(\Gamma)}PN'$,$PN'=(P', T', F', C; Lab, Exp, M_0)$.收敛速度可以用结点数减少/增加比来衡量:$v=\dfrac{|P^c|+|T^c|}{|P-P^c-P'|+|T-T^c-T'|}$.在 CRR-SV 各规则中,$P-P^c-P'$ 和 $T-T^c-T'$ 分别为代替组件的那些库所和变迁.$\exists k\in N$,$k\geqslant|P-P^c-P'|+|T-T^c-T'|$,因此:$v=\dfrac{|P^c|+|T^c|}{k}$,$|P^c|+|T^c|$ 和 v 成正比,表示组件结构越复杂,化简的收敛速度越快.

定理 3 说明了化简规则的效率,这一点从上节的举例中可见一斑.如果有丰富可用的过程组件库,过程模拟和验证的速度都将得以大幅提高.

CBR 方法可与上文引用文献中的形式化验证方法从以下 3 个角度进行比较:(1) 是否面向工作流过程:因为面向不同的应用领域,验证方法可能选择不同的"过程模型",表达不同的"形式语义";(2) 是否基于化简技术:基于化简技术的方法不构造全局的状态空间,具有较好的"可扩展性";(3) 是否基于组件重用:基于组件的方法也具有较好的"可扩展性".比较结果如表 1 所示.

5 结论

过程验证技术是工作流系统中重要但复杂的研究课题之一. 文中侧重研究了大型、灵活过程的语义验证问题: (1) 首先提出描述过程的 3DWFN 模型, 作为语义验证的基础. 该模型不仅能够完整地描述控制流、数据流和资源三维观点表现的过程语义, 而且是基于组件的模型, 在描述大型、灵活的过程方面具有一定优势; (2) 接着提出一种支持过程语义验证的化简方法 CBR, 并具体描述了化简规则集 CRR-SV, 具有如下特点: 第一, 该方法以组件为单元进行化简; 第二, 提出的语义级化简规则, 在化简前后, 不仅考虑结构连接(静态逻辑) 的变化, 更重要的是考虑语义衔接(动态逻辑) 的变化.

表 1　几种验证方法的比较

验证方法	过程模型	形式语义	可扩展性	组件间逻辑
基于系统状态可达性的、非组件方法	Petri 网[3], WFRS[4] 命题逻辑[8], 矩阵[9]	没有完整描述工作流过程的三维元信息	较差	无
基于系统状态可达性的、基于组件的模型检测方法	时间 Petri 网[18]	使用 CTL 等描述组件的接口之间的逻辑	中等	粘合逻辑
基于化简技术的非组件方法	Petri 网[5,6], 工作流图[7]	侧重描述工作流过程的控制流	中等	无
基于化简技术的组件级方法	时间 Petri 网[16]	描述 C2 系统的控制流和时间	良好	连接逻辑
CBR	3DWFN	完整描述工作流过程的控制流、数据流和资源	良好	协作逻辑

现有大多数工作流系统一般只提供对过程的语法检查, 一部分支持形式化的控制流验证. 本文的工作促进了现有形式化验证技术从"面向结构" 向"面向语义" 的转变.

语义验证检测过程三维基本观点协作的正确性, 保证过程的目标实现, 更符合企业对业务过程正确性的实际需求. 然而企业的实际需求是多样的, 他们可能要求在更严格或者更丰富的条件约束下实现业务目标, 例如时间条件的约束等, 因此, 如何在更多约束、完成更复杂目标的系统设计实现中, 扩展和使用这些原理方法是有待进一步研究的内容.

参考文献:

[1] Workflow Management Coalition. Terminology & Glossary [S]. WFMC-TC-1011, Feb. 1999.

[2] M Reichert, T Bauer, et al. Enterprise-wide and cross-enterprise workflow-management-challenges and research issues for adaptive workflows [A]. Proc. of the Informatik'99 Workshop of Enterprise-Wide and Cross-Enterprise Workflow-Management [C]. Paderborn, Germany, 1999. 56-64.

[3] W Aalst. Workflow verification: finding control-flow errors using petri-net-based techniques [M]. W Aalst, J Desel, A Oberweis (eds.): Business Process Management. Berlin: Springer-Verlag, 2000. 161-183.

[4] K Barkaoui, L Petrucci. Structural analysis of workflow nets with shared resource [A]. Workflow Management: Net-based Concepts, Models, Techniques and Tools [C]. Lisbon, Portugal, 1998. 82-95.

[5] W Aalst, A Hirnschall, et al. An alternative way to analyze workflow graphs [A]. A Banks Pidduck, J Mylopoulos, et al (Eds.). 14th International Conference of Advanced Information Systems Engineering [C]. Toronto, Canada, Springer Verlag, 2002. 535-552.

[6] 李建强, 范玉顺. 基于 Petri 网化简方法的工作流模型验证[J]. 信息与控制, 2001, 30(6): 492-497.
LI Jian-qiang, FAN Yu-shun. Research of petri net based workflow model reduction methods [J]. Information and Control, 2001, 30(6): 492-497. (Chinese)

[7] W Sadiq, M Orlowaka. Analyzing process models using graph reduction techniques [J]. Information Systems, 2000, 25(2), 117-134.

[8] Bi H, Zhao J. Applying propositional logic to workflow verification [J]. Information Technology and Management: Special Issue on Workflow and E-business, 2004, 5(3-4): 293-318.

[9] Y Choi, J Zhao. Matrix-based abstraction and verification for E-business processes [A]. Proceedings of the 1st Workshop on e-Business [C]. Barcelona, Spain, 2002. 154-165.

[10] Y Lei, MP Singh. A comparison of workflow metamodels [A]. Proceedings of the ER-97 Workshop of Behavioral Modeling and Design Transformations: Issues and Opportunities in Conceptial Modeling [C]. Los Angeles, USA, 1997.

[11] E Sivaraman, M Kamath. On the use of petri nets for business process modeling [A]. Proc. of the 11th Annual Industrial Engineering Research Conference [C]. Orlando, Florida, 2002.

[12] S Sadiq, M Orlowska, et al. Data flow and validation in workflow modelling [A]. K Schewe, H Williams (eds.). Conferences in Research and Practice in Information Technology [C]. Dunedin, New Zealand, 2004. 207-214.

[13] H Zhuge. Component-based workflow systems development [J]. Decision Support Systems, 2003, 35(4): 517-536.

[14] Z Stojanovic, A Dahanayake, H Sol. Integrated Component-Based Framework for Effective and Flexible Telematics Application Development [R]. Delft University of Technology, ISBN: 90-76412-13-8. 2000.

[15] T Murata. Petri nets: properties, analysis and applications [J]. Proceedings of the IEEE, 1989, 77(4): 541-580.

[16] Daniel Karlsson, Petru Eles, et al. Formal verification in a component-based reuse methodology [A]. Proceedings of the 15th international symposium on System Synthesis [C]. New York, USA: ACM Press, 2002. 156-161.

[17] W Aalst. The application of petri nets to workflow management [J]. The Journal of Circuits, Systems and Computers, 1998, 8: 21-66.

[18] J Wang, Y Deng, M Zhou. Compositional time petri nets and reduction rules [J]. IEEE Transactions on System, Man and Cybernetics, 2000. Part B, 30(4): 562-572.

作者简介:

周建涛　女, 满族, 1974 年 2 月出生于内蒙古呼和浩特市, 清华大学博士研究生, 主要研究领域为计算机网络、计算机支持的协同工作、工作流和形式描述技术等. E-mail: cszjtao@mail.imu.edu.cn