工作流模式的π演算建模研究

黄贤明,李长云,梁爱南
HUANG Xian-ming,LI Chang-yun,LIANG Ai-nan

湖南工业大学 计算机系,湖南 株洲 412008
Department of Computer,Hunan University of Technology,Zhuzhou,Hunan 412008,China
E-mail:hxmshjs@21cn.com

HUANG Xian-ming,LI Chang-yun,LIANG Ai-nan.Modeling research for workflow patterns based on pi-calculus.Computer Engineering and Applications,2007,43(17):66-68.

Abstract: The π-calculus is a kind of mobile process algebra which can be used to model concurrent and dynamic systems.According to the syntax of π-calculus,a novel formal modeling method of workflow patterns has been proposed.The π-calculus has been regarded as a formal foundation for workflow,furthermore models workflow patterns by using the π-calculus.
Key words: π-calculus;ECA rule;workflow patterns

摘 要:π演算是一种移动进程代数,可用于对并发和动态变化的系统进行建模。根据π演算的语法,提出了一种新的用于工作流模式的形式化建模方法,以π演算作为工作流形式化的基础,并利用π演算对工作流模式进行了建模。
关键词:π演算;ECA规则;工作流模式
文章编号:1002-8331(2007)17-0066-03 **文献标识码**:A **中图分类号**:TP311

1 引言

为了实现工作流管理功能,必须将业务过程从现实世界中抽象出来,并用一种形式化方法对其进行描述。π演算(π-calculus)是Robin Milner提出的以进程间的移动通信为研究重点的并发理论,它是对CCS(Calculus of Communication System)的发展[1,2]。π演算是系统交互行为建模的理论基础,适合描述动态系统。它既提供了行为等价理论,又支持系统行为的分析。在工作流建模阶段使用π演算有助于清楚地描述工作流;而在模型建立后,则可利用π演算来推演系统的行为,同时验证模型的正确性,如发现系统行为不完整、死锁、缺少同步等;另外,π演算作为一种强大和成熟的形式化方法,π演算有支持其正确性验证和相关应用的工具,所以π演算自然地成为工作流过程建模的理想工具,例如,文[3,4]使用π演算对工作流进行了初步的形式化。

工作流模式(workflow patterns)是工作流建模的基本构造单元,它为工作流过程处理带来了极大的灵活性。本文着眼于对工作流过程建模中反复出现的模式即工作流模式进行π演算建模研究。

2 π演算

π演算的基本计算实体为名字和进程,进程之间的通信是通过传递名字来完成。由于π演算不但可以传递CCS中的变量和值等,还可以传递通道名,并且将这几种实体都统称为名字而不再作区分,这使得π演算具有了建立新通道的能力,因此π演算可以用来描述结构不断变化的并发系统。

π演算有几种不同的符号表示[1,2,5,6],下面介绍π演算的基本语法可由以下BNF范式给出:

$P::=M|P|P|vzP|!P$

$M::=0|\pi.P|M+M$

$\pi::=\bar{x}\langle y\rangle|x(z)|\tau|[x=y]\pi$

π演算中最简单的实体是名字,进程通过名字进行交互,并在交互中传递名字。名字的语法定义是标识符。上述π演算语法定义中,x是单个名字,$\bar{x}$与x互为对偶名字(Co-names),进程通过对偶名字进行交互。P是π演算中另一实体进程的语法定义,而M是π演算中"和"(summations)的语法定义,用以表达选择关系。π的语法给出进程能够执行的四种动作,称为前缀。进程通过执行这些动作进行演化。

π演算语法的形式语义运用约简关系和变迁关系定义,下面运用自然语言给出它的直观定义。

(1)0表示非活动进程,即不做任何工作的进程。

(2)前缀$\pi.P$表示具有π表示的单一行为能力,该能力执行后,执行进程P。

输出前缀$\bar{x}\langle y\rangle.p$表示通过名字$x$输出名字$y$,然后执行进程$P$。

输入前缀$x(z).p$表示通过名字x输入名字,并用输入的名字替换进程P中z,然后执行替换后的进程P。

哑前缀$\tau.P$表示做哑动作τ然后执行P,一般来说,τ用于

基金项目:湖南省教育厅资助科研课题(the Research Project of Department of Education of Hunan Province,China under Grant No.052520)。
作者简介:黄贤明(1976-),男,讲师,研究方向:工作流,数据库;李长云(1972-),男,副教授,博士,研究方向:工作流、语义Web和软件体系结构;梁爱南(1976-),女,讲师,研究方向:工作流。

表示进程外部不可见的内部动作。

匹配前缀[x=y]π.P 表示当 x 和 y 是同一名字时,执行 π.P,否则不做任何工作。

(3)“和”M1+M2 表示选择执行进程 M1、M2 中的一个。

(4)组装 P1P2 表示并行执行 P1 和 P2。

(5)限制 vzP 表示名字 z 是 P 的局部名字,P 在名字 z 上的外部动作被禁止,但 P 通过名字 z 的内部通信是允许的。

(6)复制! P 表示无限多个 P 的组装。

3 用 π 演算形式化工作流模式

3.1 工作流模式简介

工作流技术将应用逻辑与过程逻辑分离开来,过程逻辑是通过工作流语言描述的。为了提供基本的工作流建模并评价工作流语言的描述能力,W.M.P.van der Aalst 等借鉴设计模式的思想,对工作流模式[7-8]进行了收集和研究。工作流模式指在过程建模中反复出现的模式,与特定的工作流语言无关。工作流模式从控制流的角度系统地描述了过程定义语言需要满足的业务需求。

工作流模式分为 6 类,共包含 20 个模式,具体分类如表 1 所示。

表 1 工作流模式分类表

模式分类	模式名称
基本控制流模式	顺序、并行分支、同步、互斥、简单汇聚
高级分支和同步模式	多重选择、同步汇聚、多重汇聚、鉴别器
结构化模式	任意循环、隐式终止
多实例模式	没有同步的多实例、具有先验设计时知识的多实例、具有先验运行时知识的多实例、不具有先验运行时知识的多实例
基于状态的模式	延迟选择、交叉并行路由、里程碑
取消模式	取消活动、取消实例

3.2 工作流模式的 π 演算建模方法

建模时,将活动模型化为进程(process),让每一个活动对应一个独立的进程,每一个进程有前置条件和后置条件。进程 P 是进程 Q 的前置条件表示 P 完成后,Q 才能开始;进程 P 是进程 Q 的后置条件表示 Q 完成后,Q 要给 P 发送消息。对进程定义时使用事件-条件-动作(Event-Condition-Action,简称 ECA)规则,ECA 规则描述了触发活动的事件和内部条件,实际上也描述了执行活动的执行依赖关系。ECA 规则中的事件和条件对应进程的前置条件,事件模型化为 π 演算中的输入前缀,条件模型化为 π 演算中的匹配前缀;ECA 规则中的动作分成两种:分别对应进程的内部动作和进程的后置条件。显然,如果一个进程定义时没有事件部分(即输入前缀),则这个进程表示一个开始活动;如果一个进程定义时没有后置条件,则这个进程表示一个最后的活动。因此,定义一个基本活动可用如下 π 演算建模:

$x.[a=b].\tau.\bar{y}.0$

下面使用上述方法对各种模式进行建模:

顺序模式的解决办法:对这种模式进行 π 演算形式化时,把一个活动模型化为进程 A,另一个活动模型化为进程 B,进程 A 和进程 B 通过通道 b 进行通信,如图 1。由上文介绍的 π 演算形式化工作流模式的基本思想,可得到如下的 π 演算建模:

$A=\tau_A.\bar{b}.0$

$B=b.\tau_B.B'$

其中,进程 B 在完成内部动作 τ_B 后,激发后续进程 B′。

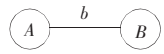

图 1 顺序模式

并行分支模式的解决办法:对这种模式进行 π 演算形式化时,把一个活动模型化为进程 A,另外两个活动模型化为进程 B 和进程 C,进程并行分支为进程 B 和进程 C,进程 A 和进程 B 通过通道 b 进行通信,进程 A 和进程 C 通过通道 c 进行通信,如图 2。由上文介绍的 π 演算形式化工作流模式的基本思想,可得到如下的 π 演算建模:

$A=\tau_A.(\bar{b}.0|\bar{c}.0)$

$B=b.\tau_B.B'$

$C=c.\tau_C.C'$

其中,进程 B 在完成内部动作 τ_B 后,激发后续进程 B′;进程 C 在完成内部动作 τ_C 后,激发后续进程。

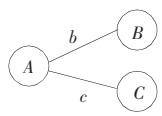

图 2 并行分支模式

鉴别器模式的解决办法:把一个活动模型化为进程 A,另外三个活动模型化为进程 B、进程 C 和进程 D,代表鉴别器的进程 D 分成进程 D_1 和进程 D_2,进程 D_1 和进程 D_2 有两个新名字 h 和 exec。如果进程 D_1 收到 d_1 或 d_2 或 d_3 就触发进程 D_2,通过触发 D_2 来激活进程 D 的内部动作 τ_D。进程 D_2 激活进程 D 的内部动作 τ_D 之后,进程 D_2 等待进程 D_1 其它所有分支的触发 h,最后,进程 D_2 使用递归又重置到鉴别器 D。如图 3,可得到如下的 π 演算建模:

$A=\tau_A.\bar{d}_1.0$

$B=\tau_B.\bar{d}_2.0$

$C=\tau_C.\bar{d}_3.0$

$D=(\nu h,exe)(D_1|D_2)$

$D_1=d_1.\bar{h}.0|d_2.\bar{h}.0|d_3.\bar{h}.0$

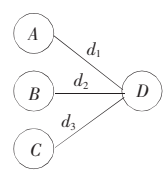

图 3 鉴别器模式

$D_2=h.\overline{exec}.h.h.D|exec.\tau_G.D'$

上述式子中,进程 D_1 并行等待输入触发 d_1 或 d_2 或 d_3,若收到一个触发,它通过发信号 h 给进程 D_2 来匿名收到的触发。

上面介绍的是有三个输入控制触发的鉴别器,对于有多个输入控制触发的鉴别器可用下式表示:

$$D=(\nu h,r)((\prod_{i=1}^{m}d_i.\bar{h}.0)|h.\bar{r}.\{h\}_1^{m-1}.D|r.\tau_D.D')$$

其中,引入了操作符∏表示 m 个输入触发,在收到第一个触发后,用操作符{}表示等待余下的 m-1 个匿名输入触发。

交叉并行路由模式的解决办法:把一个活动 A 模型化为

进程 A,另外两个活动 B 和 C 模型化为进程 B 和进程 C,进程 A、进程 B 和进程 C 共有两个新名字:$(\upsilon x,y)(A|B|C)$,x 用来以任意顺序触发进程 B 和进程 C,y 用来发送被触发了的进程已经完成了的消息,如图 4。

可得到如下的 π 演算建模:

$A=\tau_A.\bar{x}.\bar{y}.\bar{x}.\bar{y}.A'$

$B=x.\tau_B.\bar{y}.0$

$C=x.\tau_C.y.0$

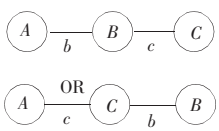

图 4 交叉并行路由模式

多实例模式在大部分的工作流引擎中都不支持在某一时刻同一活动存在一个以上的实例,所以文中没有讨论,关于其它模式的 π 演算建模见表 2。

4 实例研究

图 5 是用业务流程建模符号 (Business Process Modeling Notation,BPMN)表示的工作流,对它进行 π 演算建模,把图中流对象用 π 演算进程表示, 如: 开始事件用进程 N_1 表示,XOR-split 通路用进程 N_2 表示,活动 A 用进程 N_3 表示,活动 B 用进程 N_4 表示,XOR-join 通路用进程 N_5 表示,结束事件用进程 N_6 表示。图中顺序流用 π 演算中的名字表示,如图中分别用名字 e_1、e_2、e_3、e_4、e_5、e_6 表示。

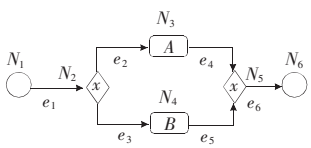

图 5 一个简单的 BPMN 图

进程 N_1 是开始事件,它没有前置条件,它的内部动作用 τ_{N_1} 表示,它的后置条件为输出名字 e_1,名字 e_1 将触发进程 N_2,可如下表示进程 N_1:

$N_1=\tau_{N_1}.\bar{e}_1.0$

进程 N_2 代表 XOR-split 通路,即互斥选择模式,可如下表示进程 N_2:

$N_2=e_1.\tau_{N_{21}}.(\bar{e}_2.0+\bar{e}_3.0)$

进程 N_3 和进程 N_4 可如下表示:

$N_3=e_2.\tau_{N_3}.\bar{e}_4.0$

$N_4=e_3.\tau_{N_4}.\bar{e}_5.0$

进程 N_5 代表 XOR-join 通路,即简单汇聚模式,可如下表示进程 N_5:

$N_5=\upsilon x((e_4.\bar{x}.0+e_5.\bar{x}.0)|x.\tau_{N_5}.\bar{e}_6.0)$

进程 N_6 是结束事件, 它没有后置条件, 可如下表示进程 N_6:

$N_5=e_6.\tau_{N_6}.0$

5 总结

π 演算语言表达清晰,可理解性强,具有良好的抽象机制和逻辑推理能力,具有较强的语义表达能力,是一种很好的建模方法。运用 π 演算作为理论基础,对各种工作流模式进行建模,能够方便地描述和分析工作流。基于工作流模式分析是从控制流的角度出发,所以本文的分析结果也只是说明这种模型描述过程控制流的能力, 通过使用数据和环境变量等方式,整个工作流管理系统能满足更广泛的需求。另外需要说明的是,工作流模式所描述的业务需求虽然较广泛和全面。但是,并不能完全涵盖实际应用的所有场景。(收稿日期:2006 年 11 月)

参考文献:

[1] Milner R.The polyadic π - calculus:a tutorial[M]//Bauer F L,

表 2 工作流模式的 π 演算建模

工作流模式	π 演算形式	工作流模式	π 演算形式
同步模式	$B=\tau_B.\bar{d}_1.0$ $C=\tau_C.\bar{d}_2.0$ $D=d_1.d_2.\tau_D.D'$	互斥模式	$A=\tau_A.(\bar{b}.0+\bar{c}.0)$ $B=b.\tau_B.B'$ $C=c.\tau_C.C'$
多重选择	$A=(\upsilon exec)\tau_A.(A_1\|A_2)$ $A_1=\overline{exec}\langle b\rangle.0+\overline{exec}\langle c\rangle.0+\overline{exec}\langle b\rangle.\overline{exec}\langle b\rangle.0$ $A_2=!\,exec(x).\bar{x}.0$ $B=b.\tau_B.B'$ $C=c.\tau_C.C'$	任意循环	$A=!\ a.\tau_A.\bar{b}.0$ $B=!\ b.\tau_B.\bar{c}.0$ $C=!\ c.\tau_C.(\bar{a}.0+\bar{d}.0)$ $D=d.\tau_D.D'$
		隐式终止	0
同步汇聚	$C=\tau_C.\bar{d}_2.0$ $B=\tau_B.\bar{d}_1.0$ $D=d_1.\tau_D.D'+d_2.\tau_D.D'+d_1.d_2.\tau_D.D'$	延迟选择	$A=\tau_A.(\bar{b}.0\|\bar{c}.0)$ $B=b.(b_{enm}.\overline{kill}.\tau_B.B'+kill.0)$ $C=c.(c_{enm}.\overline{kill}.\tau_C.C'+kill.0)$
多重汇聚	$B=\tau_B.\bar{d}.0$ $C=\tau_C.\bar{d}.0$ $D=!\,d.\tau_D.D'$	里程碑	$A=check(x).([x=\Gamma]\tau_{A1}.A'+[x=\perp]\tau_{A2}.A'')$ $B=M(\perp)\|b.\bar{m}\langle\Gamma\rangle.\tau_B.\bar{m}\langle\perp\rangle.B'$ $M(x)=m(x).M(x)+\overline{check}\langle x\rangle.M(x)$
简单汇聚	$B=\tau_B.\bar{d}.0$ $C=\tau_C.\bar{d}.0$ $D=d.\tau_D.D'$	取消活动	$A\|\varepsilon=a.\tau_A.A'+cancel.0!\ \tau_\varepsilon.\overline{cancel}.0$

(下转 73 页)