基于细胞膜演算的 Web 服务事务处理形式化描述与验证

戚正伟 尤晋元

(上海交通大学计算机科学与工程系 上海 200030)

摘 要 采用细胞膜演算具体分析了当前比较主流的 Web 服务中原子事务协调协议 WS-AT. 针对 WS-AT 协议采用简单的状态转换表和转换图,无法描述协调者和多个参与者的复杂协调活动,采用细胞膜演算给出了其形式化描述,用于规范协调者和参与者的活动,并分析了该协议的活性和安全性,得到了 38187 个状态. 模型检验的实验结果表明,该协议满足稳定性、一致性和非平凡性,而不满足非阻塞性. 进而,分析出注册和协调协议混在一起是其不满足非阻塞性的原因.

关键词 细胞膜演算;事务处理;形式化方法

中图法分类号 TP311

The Formal Specification and Verification of Transaction Processing in Web Services by Membrane Calculus

QI Zheng-Wei YOU Jin-Yuan

(*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai* 200030)

Abstract It is important to adopt a suitable formal method to specify and verify complex transactions in Web services. The authors have developed a formal method called Membrane Calculus to describe the abstract model in long running transactions. This paper extends Membrane Calculus to analyze the practical atomic transaction commit protocol WS-AT. Due to the simple State Transition Table and Chart, WS-AT can not describe the complex coordination activities between the coordinator and several participants. Membrane Calculus is used to formally specify the behavior of the coordinator and participants and analyze the Safety and Liveness of WS-AT. The Model Checking experiments show that there are 38,187 states in the authors' model and Stability, Consistency, Non-Triviality are satisfied while Non-Blocking is not. The reason is that WS-AT mixes the registration and coordination protocols.

Keywords membrane calculus; transaction processing; formal methods

1 引 言

目前没有描述 Web 服务事务被普遍接受的方法[1]. 没有一个公认的统一处理框架,许多情况下,大多是用某种特定方法解决特定问题,例如,采用 Petri 网分析两阶段提交协议的性质[2]. 目前的形式化方法针对传统的事务处理居多,而缺乏对于 Web

收稿日期:2006-02-16;修改稿收到日期:2006-04-21. 本课题得到国家自然科学基金(60173033)与国家"八六三"高技术研究发展计划项目基金(2004AA104340,2004AA104280)资助.戚正伟,男,1976 年生,博士,讲师,主要研究方向为分布式计算、形式化方法以及事务处理. E-mail: qizhwei@sjtu.edu.cn. 尤晋元,男,1939 年生,教授,博士生导师,CCF 高级会员,主要研究领域为分布式系统与计算、面向对象技术和软件工程.

服务和网格中的长事务以及具体的协议等的形式化分析方法.

现有的方法主要分为两类[3]:一类主要是扩展现有的进程代数和 Petri 网等形式化方法[4]. Bruni 等人提出用 John 演算描述事务的方法,并给出了集中式/分布式两阶段提交协议的描述[5]. Bruni 等人又提出了一种 Join 演算的扩展方法,称为"提交 Join 演算"(Committed Join)[6],描述分布式协商(distributed negotiation)问题(事务是其特殊类型).提交 Join 演算满足了大部分分布事务处理的要求,但是,由于细胞膜是匿名的,没有位置的概念,反应规则里面没有涉及到跨细胞膜的操作.也不像灰箱演算[7]那样,具有移动细胞膜的能力,这些缺陷导致在描述分布事务的语义时,很难对其语义进行比较完整的形式化分析. Bruni 和 Montanari 等提出了用零安全网(Zero-safe Nets)[8]描述事务的性质.但零安全网没有描述隔离性和回滚等重要性质. Butler 与 Ferreira 提出一种在传统的进程代数 CCS 与 CSP 上加入补偿算子描述 Web 事务的方法 StAC(Structured Activity Compensation)[9]. StAC 的优点在于给出了事务失败后补偿的流程控制,其不足之处是语法不简洁,提供大量的操作原语,很难推导异常处理、补偿、并行和嵌套处理的语义,对于一个简单补偿操作也需要大量原语. Bruni 等在 POPL2005 会议上提出了一个比 StAC 简洁的补偿进程代数[8],定义了一系列语法构造,例如,串行、并行、嵌套等,并采用 Big Step Semantics 定义相应的语义.但这个进程代数是一个比较初步的原型,没有加入状态、变量、控制结构和活动之间的数据传输,有待于进一步完善.从事务处理的角度来说,StAC 和该进程代数从补偿的流程控制着手,试图给出一个理论上比较完善的补偿语义,用于描述 Web 服务中长时间运行的非原子事务的补偿.但是这些模型还是停留在理论阶段,需要进一步的改进,而且没有涉及模型检验 Model Checking 方面,离实际应用还有一段距离.

另外一类是基于逻辑方法对协调协议进行形式化描述与正确性证明. Johnson 等采用 TLA+逻辑验证了 WS-AT 协调协议的部分正确性[1]. 关于事务处理的形式化方法很多,说明这方面的研究比较活跃.但是就进程代数而言,像 StAC 的语法太复杂[1],而且没有成熟的方法对具体的协调协议进行描述.对于逻辑的方法而言,可以用 TLA+对 WS-AT 一部分正确性进行验证,这种描述方式与其它并发系统一样,并没有像进程代数那样专门针对 Web 事务有一种比较系统的方法.现有的事务处理形式化方法对事务处理的描述居多,但并发系统中的有些性质需要用时序逻辑描述和验证.上述形式化方法目前缺少对事务处理模型检验的实际应用,大都没有形成形式化描述与模型检验统一的框架.而且现有的方法侧重于理论上的探索,像TLA+那样将理论成果应用于目前主流的事务提交协议的分析和验证比较少,但 Web 服务事务提交协议又迫切需要进行形式化描述和模型检验.

在文献[12]中,我们提出一种新的形式化方法,叫做细胞膜演算,用于描述 Web 事务,该方法是基于 Petri 网与细胞膜计算[13]模型,将进程代数与 Petri 网结合起来描述 Web 事务,但是该演算还没有形式化工具支持,需要像 TLA+那样能够对 Web 服务进行正确性证明.本文继续按照这种思路,采用细胞膜演算对其进行形式化描述,并对其安全性和活性进行模型检验.

本文第 2 节介绍有关细胞膜演算的基本概念;第 3 节给出 WS-AT 的细胞膜演算的形式化描述;第 4 节讨论 WS-AT 模型检验的活性与安全性;第 5 节给出 WS-AT 的实验结果并与 TLA+的结果相比较;最后总结全文并给出了进一步研究的方向.

2 基本概念

2.1 细胞膜演算

P-systems[13]由 Păun 在 1998 年提出①,作为细胞膜计算(membrane computing)的模型,在理论上得到了比较充分的发展. P-systems 作为一种计算模型,是由生物学上的细胞膜特性启发的.简单说来,在细胞膜结构中,每个细胞膜包含一些分子,这些分子对象能够按照一定的反应规则演化.分子可以在细胞膜之间穿越,细胞膜本身能够溶解或者分裂. P-systems 就是按照这些特性构建出来的计算模型,并且自提出以来,出现了许多种变形,大部分是具有和图灵机等价的表达能力[15]. 关于基本模型的一个重要结果是四细胞膜结构能够模拟图灵机[15](规则之间没有优先级).

细胞膜演算是其相应的演算系统.令类型对象集合为 Σ,其元素为 T,对象多集中的元素为 $a,b,c,\cdots$,标识符集合为$\aleph$(标识符也叫做名字 name),其元素为 $i,j,k,\cdots$,细胞膜演算定义如下:

① P-systems 名称来源于 Păun systems 的简称.

$M,M'::=0|[_i\ \ O,\ \ R,\ \ M]|MM'$,

$O,O'::=0|a:T|OO'$,

$R,R'::=0|O\rightarrow O'|O\rightarrow O'_{out}|O\rightarrow O'_{In(j)}|O\rightarrow \lambda\,|R|R'$,

$\lambda::=\delta|\sigma n|\eta n|M$.

类似于 Join 演算中的递归定义方式，我们首先定义一系列的细胞膜 M,M' 等，然后定义细胞膜中的对象 O,O' 和反应规则 R,R'. 每个细胞膜是形如 $[_i\ \ O,\ \ R,\ \ M]$ 的表达式. 其中，i 为该细胞膜的标识符，代表这一个或这一类细胞膜. 并且，从递归定义可以看出，细胞膜是可以嵌套的，相邻的在同一个细胞膜下的可以称为兄弟细胞膜，细胞膜与其子细胞膜为父子关系；O 为该细胞膜中的对象多集；R 为反应规则集合；M 是其子细胞膜的多集. 对于每个对象，存在一个对应的类型，而且类型是有结构的，存在父子类型的偏序关系，每个对象的类型采用 $a:T(a\in O, T\in\Sigma)$ 的形式定义类型①. 相对于基本 P-system 中的 out 和 $In(j)$ 这样的通信方式，这里采用相应的反应规则来表示.

反应规则也相应地用表达式 $O\rightarrow O'|O\rightarrow O'_{out}|O\rightarrow O'_{In(j)}$ 和 $O\rightarrow\lambda$ 表示. 前三条表示对象规则，仅改变对象在细胞膜结构中的分布和数量，不涉及细胞膜结构的改变. 最后为细胞膜规则，通过移动、溶解、生成来改变细胞膜的结构. λ 定义为 $\lambda::=\delta|\sigma n|\eta n|M$，其中 δ 表示溶解，σn 表示移出标识符为 n 的父细胞膜，成为其兄弟细胞膜. ηn 表示移入到标识符为 n 的细胞膜，成为其子细胞膜. M 表示创建一个细胞膜作为其子细胞膜.

和灰箱演算相比较，这里的移动仅仅考虑细胞膜自身移动的效果，而没有像灰箱演算那样有一套安全机制来控制细胞膜的溶解和移动. 这里缺少了安全方面的考虑，但由于事务处理机制和安全是个正交的概念，在 Web 服务中用不同的协议来实现. 为使我们的目标更为集中，这里仅仅考虑比较单纯的细胞膜结构改变情况，而不加入安全方面的控制机制. 并且，在现有的基础上加入安全机制也是比较容易的.

2.2　带条件的扩展细胞膜演算定义

在文献[3]中采用重写逻辑(rewriting logic)定义基本细胞膜演算模型. 我们可以对照重写逻辑，加入条件规则，也就是说，有些规则能够起作用是有一定条件的. 重写逻辑采用的是等式成员逻辑的条件和重写条件，在 π 演算中采用形如 if $M=N$ then P else Q 这样的格式. 这里采用重写逻辑中的条件形式. 同时，将针对细胞膜演算的反应规则扩展为一般的重写规则. 在这些扩展基础上，对基本细胞膜演算进行扩展，得到通用的细胞膜演算，从而使我们的方法具有更强的表达能力. 在重写逻辑框架下，条件可以采取三种形式，即等式 $u=u'$，成员判断 $v:s$ 和重写 $w\rightarrow w'$，其含义和重写逻辑的条件相一致.

定义 1.　带条件的扩展细胞膜演算是一个四元组 $\mathcal{R}_{EMC}=(\Sigma_{EMC},E_{EMC},\varnothing,R_{EMC})$，其中 (Σ_{EMC},E_{EMC}) 是扩展的成员等式理论，包括基本的 (Σ_{MC},E_{MC}) 以及扩展的基调和相应的等式；R_{EMC} 为带条件的重写规则集合，包含基本细胞膜演算规约规则集 $\mathbb{R}$ 以及扩展的带条件规则集.

这里的扩展只限于两种情况，一种是根据需要定义对象的子类型以及该子类型的等式和算子，并且定义针对该子类型的重写规则. 另外一种是将规则扩展为条件重写规则，支持条件判断. 扩展的原则是不改变基本细胞膜演算模型 MembraneCalculus 的语义，只是在这个模型上的进一步细化. 有关扩展细胞膜演算的模型检验 Linear Temporal Logic (LTL)[17] 框架，是基本细胞膜演算的相应扩展，即将原来的 $\mathcal{R}_{MC}$ 换成 $\mathcal{R}_{EMC}$，这里不再赘述.

3　WS-AT 形式化描述

3.1　WS-AT 状态转换图和状态转换表

WS-AT/WS-BA 由 BEA、IBM 和 Microsoft 等机构发布，将协调框架和事务框架分开处理，定义了两种不同事务类型：原子事务 AT(Atomic Transaction)和业务事务 BA(Business Activities). 原子事务类型是传统的事务处理方式，符合 ACID 性质，是传统事务在 Web 服务中的扩展. 业务事务采用前向补偿，是松耦合的 Web 服务的事务处理方式，不符合 ACID 性质. 本文将采用形式化方式对 WS-AT 进行模型检验. 为了便于描述协议的协调过程，WS-AT 规范给出了状态转换图和状态转换表. 状态转换图如图 1 所示.

图 1 简单描述了协调者与参与者之间接受不同消息之后的状态转换. 比较形象直观，可以帮助我们理解 WS-AT 协议. 在该规范附录中给出了状态转换表(文献[14]的附录 1 也给出相关状态转换表)，比状态转换图要详细，而且规定了在给定状态下接受到所有可能的消息后采取的动作和状态转换. 表面上看，状态转换图与状态转换表比较规范地描述了 WS-AT 协议. 但是，这些图与表仅仅给出了一个比较粗略的描述，而且没有考虑到多个参与者之间

① 如果上下文能够指出 a 的类型，$a:T$ 简写为 a.

的相互交互.这个也是自然的,本来并发系统的形式化描述是一件很困难的事情.仅靠状态转换图和表无法描述这些并发系统中互相交互的复杂关系.我们采用细胞膜演算刻画其中的复杂的消息传递和状态转换关系.而且我们需要对 WS-AT 协议作一些更高层次的抽象,因为像协议中的 XML 之类的消息表示语法比较复杂,这里将消息抽象成为一种特殊的对象.这种抽象方式也是许多形式化系统里面所采用的.

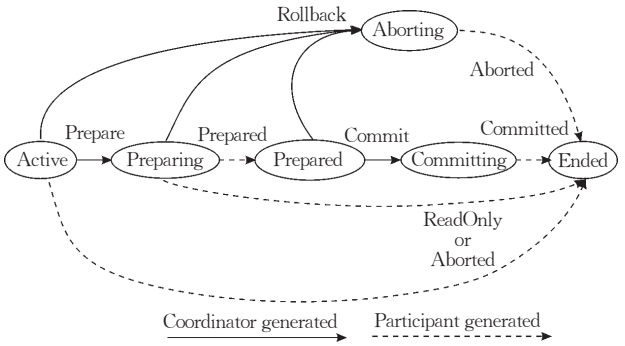

图 1 WS-AT 状态转换图

3.2 WS-AT 形式化模型

WS-AT 协议采用自然语言描述,并加入上述的状态转换图和表半形式化说明.在我们之前的研究工作中,已经可以采用重写逻辑来形式化描述抽象的事务处理过程[3].在该文献的 Web 服务经典例子中,基本的概念都可以用于 WS-AT 中.我们采用对象来描述状态与消息,采用规则来描述状态的变化和消息的发送与接收,采用细胞膜来层次化各个事务参与方.与抽象模型相比,WS-AT 协议的复杂之处在于描述消息传递时采用的 XML 消息结构以及其它 Web 服务的基础设施.对于一个分布式松耦合的 Web 服务系统,从模型上讲,消息如何表示和传递是不关心的,关心的是消息传递语义,即接受消息和发送消息以及状态转换是如何进行的.在状态转换表上,消息的发送和接受没有考虑到多个参与者之间的交互以及内部状态的变换.例如:当协调者处于 *Committing* 状态时,一个参与者发来 *Committed* 消息,状态转换表的动作是 *Forget* 动作,并且保持在 *Committing* 状态.这个对于只有一个参与者是对的,但是对于多个参与者来讲,每一个参与者发来的 *Committed* 消息都应该记录,并且等到所有这类消息到达后,给应用程序发一个 *Committed* 消息.因为形式化验证需要一个定义比较良好的形式化规范,我们这里对协议做了相应的抽象.具体做法如下:

(1) 规范中每个协调者的状态由两个部分表示,第一部分是状态标识符,与协议中一致,比如 *None*, *Preparing*[①] 之类,第二部分是注册到该协调者的参与者的信息,我们这里采用格式 $op\ RegInfo$: $source\ status \rightarrow RegList$. 即 *source* 表示参与者的编号,*status* 表示该参与者现在的状态.例如: $RegInfo('2,'Committed)$ 表示协调者知道参与者 $'2$ 的状态为已经提交.

(2) 我们采用 $op\ \{\_,\_,\_\}$: $source\ target\ msgtype \rightarrow Xmsg$ 表示一条消息,*source* 和 *target* 分别表示消息的源与目的,*msgtype* 表示消息的类型,这种表示方式很自然地表示消息的方式.例如 $\{'0,'2,'Commit\}$ 表示从协调者(这里用 $'0$ 表示)发送一条消息到参与者,该消息指示参与者提交事务.

(3) 我们采用与 TLA+ 中一致的方式,采用协调者的内部数据结构记录参与者的信息(就是前面所讲的 $RegInfo$ 数据结构).状态转换时按照当前状态,所接受到的消息或者内部的决定,根据 WS-AT 协议中所规定的动作,要么改变状态,要么发送消息.这里,我们采用 $RegInfo$ 形式化地描述了协议中记录内部状态的动作.

(4) 超时处理.如果没有超时处理,该协议是一个阻塞的协议.引入超时处理后,从理论上讲,采用后发式提交方式,所有的状态到最后是一致的,要么全部提交,要么全部回滚.但 WS-AT 是否符合这条性质,需要后面的模型检验加以检验.

(5) 消息重发送机制.有些状态下,消息可以再次发送,但是不改变状态.例如:协调者在 *PreparedSuccess* 状态下,如果有提交超时,那么会再次发送 *Prepared* 消息,但是没有改变状态本身.在形式上是这样一条状态转换规则:

$$'PreparedSuccess \rightarrow 'PreparedSuccess\{0,2,'Prepared\}.$$

很容易看出,这样的规则引入一个无限重写循环,而且在系统中除了发出许多同样的消息外,没有改变什么状态.所以我们在这里假定,忽略单纯的不改变状态的重发消息.

(6) 无效状态与错误处理.在状态转换表中,有许多栏是"N/A"、"Ignore"、"Invalid State",这些动作不改变状态,也不发送或接受消息,或者是根本不应该发生(出错状态).遇到这些状态转换,我们这里假定忽略这些动作,和状态转换表一致,没有具体出错方面的状态.

① 在 Maude 中,相应的表示方式为 $'None$, $'Preparing$,本文对这两种方式不加区分.

(7) 两种二阶段提交协议. WS-AT 支持 *Durable 2PC* 和 *Volatile 2PC* 两个提交方式，但这个只是先后顺序不同，整个流程是一样的，并且在TLA+中也没有特别加以区别，我们这里也不区分两者的差别.

WS-AT 的形式化描述是一个扩展细胞膜演算重写模型，该模型的完整描述请参考文献[14]附录 2. 按照状态转换表，在某种状态下，接受到某种消息，按照内部的 *RegInfo* 信息①，进入下一个状态，并且按照规则，如果需要，向对方发送相应的消息. 这个是比较简单的描述，实际上还是有许多细节需要考虑，由于 WS-AT 协议本身描述不完全，我们在形式化的过程中，只好做些假定，这也是 TLA+ 所采用的. 这些问题是协议制定者没有考虑到的问题，需要在以后的版本中加以改进. 这从一方面也说明了形式化的重要意义，采用自然语言或半形式化的描述方式，很容易产生误解和忽略掉某些细节和重要的方面.

例如注册问题，正如 TLA+ 在描述 WS-AT 时所讲述，这部分没有描述清楚. 这里的′*None* 状态让人比较迷惑，这个状态在状态转换图中是没有的. 如果是初始状态，那么应该有注册的请求/反馈消息. 但根据状态转换表，只有协调者接受′*Register* 消息，但是参与者永不发送′*Register* 消息. 这里是一个硬伤，出现这个问题的原因在于协议制定者把二阶段完成协议与注册协议没有作一致考虑. 我们这里只好增加一个′*Registering* 中间状态，当参与者收到应用程序要求参与事务的时候，参与者向协调者发送一个注册消息，进入′*Registering* 状态，当协调者记录好该参与者注册信息，反馈一个′*RegisterResponse*，那么参与者与协调者进入′*Active* 状态.

4　WS-AT 模型检验

4.1　安全性与活性定义

在形式化描述的基础上，可以对这个模型进行模型检验. 根据细胞膜演算重写逻辑[3]，采用 LTL 时序逻辑作为模型检验方法. 在 TLA+ 中，仅仅验证了其部分正确性，即一致性. 这里对事务提交协议做更进一步的考察. 根据文献中提到的提交协议应该具有的性质[18]，给出如下定义.

定义 2.　安全性.

稳定性(Stability). 一旦进入 *Committed* 或 *Aborted* 状态，就会永远保持在这个状态②.

一致性(consistency). 所有的参与者要么都进入 *Committed* 状态，要么都进入 *Aborted* 状态. 不可能一部分进入 *Committed*，另一部分进入 *Aborted* 状态.

这两条是安全性方面的要求，意味着一旦有一个参与者进入 *Committed* 状态，没有其他参与者能够进入 *Aborted* 状态. 反过来也是.

定义 3.　活性.

非阻塞性(Non-Blocking). 参与者与协调者可以达到协议的终止状态.

非平凡性(Non-Triviality). 如果所有的参与者准备好，那么协议有可能进入 *Committed*③.

第一条保证协议总是会终止的，不会陷入死循环或其它非终止状态. 理论上讲，如果没有超时机制，两阶段提交协议是阻塞协议，不满足这条要求. 现在 WS-AT 加入超时机制，理论上是非阻塞的，但是具体实现会不会满足这条要求呢？需要模型检验来证明. 非平凡性是说每个参与者会终止，但不希望总是进入 *Aborted* 终止状态，因为这是一个满足一致性的最简单情况，这里希望如果可以提交，那么协议会达到 *Committed* 终止的状态.

4.2　安全性与活性的 LTL 表达

首先是稳定性(Stability)，稳定性比事务性质里面的持久性 Durablity 要求高，持久性说的是一旦事务提交，结果便不会被事务修改. 稳定性要求不管提交还是流产状态，一旦达到这个状态，就永远无法被本事务修改. 对于参与者和协调者来说，持久性用 LTL 表达，就是

$$Stability(J1)=[](((Committed(J1)\rightarrow [](Committed(J1)\backslash/(Aborted(J1)\rightarrow [](Aborted(J1))). \quad (1)$$

表明在任何状态下，一旦进入提交或流产状态，那么便永远处于这个状态.

一致性(Consistency)是事务处理中最基本的要求，也是事务协调的基本目标. 在文献中对两阶段协议进行了验证[1]. 这里针对 WS-AT 协议，定义如下：

$$Consistency=[](((Committed('0)\rightarrow CommittedEq('2)/\backslash CommittedEq('3))/\backslash (Aborted('0)\rightarrow Aborted('2)/\backslash Aborted('3))).$$

这条性质说明，在任何一条路径上，当协调者处于提交状态时，参与者也是处于提交状态；如果处于

① 这种信息为半形式化的状态转换表所缺少，这里采用细胞膜演算将这部分形式化表述.

② 这里说的是一个事务上下文中，原文献仅仅涉及资源管理器，这里要求协调器也要具备这一性质.

③ 如果协议一开始都进入 *Aborted* 状态，那么一致性得到满足，但这个是最平凡地满足一致性的情况

流产状态,则参与者也都处于流产状态.

非阻塞性(Non-Blocking)采用如下公式:

$$NonBlock(J1)=\langle\rangle(Committed(J1)\backslash/Aborted(J1)) \quad (2)$$

表明在任何一条路径上,最终的状态要么是提交状态,要么是流产状态,也就是说协议可以终止,而不是停留在其它中间状态上.

非平凡性(Non-Triviality)表明在某些路径上协议最终是可以成功提交的.由于 LTL 是对所有的路径描述的,对于某些路径上的性质无能为力.那么该如何表示这个性质呢?在 Maude 里面有一条 *search* 命令,可以搜索满足某些条件的状态.

5 实验结果与分析

5.1 安全性和活性实验

我们采用重写逻辑工具 Maude 2.0 对我们建立的 WS-AT 形式化模型进行模型检验. Maude 2.0[17] (http://maude.cs.uiuc.edu)是一个由 UIUC 大学开发的同时支持等式和重写规则的工具.在我们之前的工作中,可以将一个形式化规范转换为 Maude 中可执行的模型[3].根据重写逻辑框架,如果再把协议要验证的性质定义出来,那么 Maude 将会根据 LTL 逻辑进行模型检验,验证该性质是否得到满足.在上面的讨论中,第 3 节采用重写逻辑语义定义出 WS-AT 形式化规范,第 4 节采用 LTL 逻辑定义了事务的安全性与活性(也是重写逻辑语义).

从 Maude 的角度来看,定义了可执行模型,并且有了协议性质的形式化描述,接着可以做模型检验实验.下面是我们模型检验验证的结果,首先察看一下有多少终止状态:运行命令

$$search\ in\ RUN-WSAT: wsat2pc \Rightarrow !X:Mem.$$

从图 2 中可以看出,共有 38187 个状态,花费 136879ms,重写了 249299 次,在两分多种就完成这么多次重写,效率也是很高的.其状态空间就有 3 万多个,靠人工检查不可能,下面采用重写逻辑中的 LTL 模型检验框架,验证各个性质.

首先是稳定性验证,实验结果如图 3.可以看出,通过 249361 次重写,稳定性是满足的.

其次是一致性,实验结果如图 4.可以看出,通过 261035 次重写,一致性是满足的.

接下来是非阻塞性,实验结果如图 5.可以看出,结果输出一个死循环路径不满足非阻塞性.我们分析了该违反非阻塞性的情况,原因是注册协议和协调协议混在一起,有些状态进入到注册失败的状态,而不是协调失败.而且注册本身也有许多情况,这些状态是在两阶段提交协议之外的.如果用户不考虑注册的情况,那么会得到一些比较奇怪的结果,而不是期望的要么提交,要么回滚的一致性结果.这个情况提醒 WS-AT 协议的制定者将来需要考虑注册语义和协调语义的一致性,并且采用一套机制处理注册和两阶段提交协议之间的相互作用.

```
No more solutions.
states: 38187  rewrites: 249299 in 134020ms cpu (136870ms real) (1860
    rewrites/second)
```

图 2 WS-AT 模型检验实验结果 1:终止状态

```
> reduce in CHECK-WSAT : modelCheck (wsat2pc, Statility('0) ) .
reduce in CHECK-WSAT : modelCheck(wsat2pc, Statility('0)) .

rewrites: 249361 in 144660ms cpu (150510ms real) (1723 rewrites/second)
result Bool: true
```

图 3 WS-AT 模型检验实验结果 2:稳定性

```
Maude> reduce in CHECK-WSAT : modelCheck (wsat2pc, Consistency ) .
reduce in CHECK-WSAT : modelCheck(wsat2pc, Consistency) .
rewrites: 261035 in 136970ms cpu (138390ms real) (1905 rewrites/second)
result Bool: true
```

图 4 WS-AT 模型检验实验结果 3:一致性

```
    'Preparing {'0,'3,'Commit},'Aborting > | < 'Preparing {'0,'3,
    'RegisterResponse},'Aborting > | < 'Preparing {'0,'3,'Rollback},'Aborting {
    '3,'0,'Abortcd} > | < 'Rcgistcring {'0,'3,'RcgistcrRcsponsc},'Activc >,0]],
    deadlock})
Maude> _
```

图 5 WS-AT 模型检验实验结果 4:非阻塞性

协议的非平凡性可以用 *search* 命令实现,采用命令

$$search\ in\ RUN-WSAT: wsat2pc \Rightarrow !['0:'CommittedSuccess\ O1:obj, R1:Rule, M1:Mem].$$

结果如图 6.该命令找到 5 个提交成功的终止状态,而且在最后一个终止状态(第 38186 状态)中,协调者细胞膜的对象有 $RegInfo('2,'Committed)$

$RegInfo('3,'Committed)$，说明本次提交成功是建立在两个参与者均成功提交的条件上的，也就是说，走完了完整的两阶段提交协议，最后成功提交，也就说明该协议满足非平凡性.

```
Solution 5 (state 38186)
states: 38187  rewrites: 249299 in 138520ms cpu (142110ms real) (1799
    rewrites/second)
01:Obj --> RegInfo('2, 'Committed) RegInfo('3, 'Committed)
R1:Rule --> < 'Active,'Aborting > | < 'PreparedSuccess,'Aborting > | <
    'Preparing,'Aborting > | < 'AbortedSuccess {'1,'0,'Commit},'AbortedSuccess
    {'0,'1,'Aborted} > | < 'AbortedSuccess {'1,'0,'Rollback},'AbortedSuccess {
    '0,'1,'Aborted} > | < 'Aborting {'1,'0,'Commit},'Aborting > | < 'Aborting {
    '1,'0,'Rollback},'Aborting > | < 'Active {'1,'0,'Commit},'Preparing > | <
    'Active {'1,'0,'Rollback},'Aborting > | < 'CommittedSuccess {'1,'0,
    'Commit},'CommittedSuccess {'0,'1,'Committed} > | < 'CommittedSuccess {'1,
    '0,'Rollback},'CommittedSuccess > | < 'Committing {'1,'0,'Commit},
    'Committing > | < 'Committing {'1,'0,'Rollback},'Committing > | < 'None {
    '1,'0,'Commit},'None {'0,'2,'Aborted} > | < 'None {'1,'0,'Rollback},'None {
    '0,'2,'Aborted} > | < 'PreparedSuccess {'1,'0,'Commit},'PreparedSuccess > |
    < 'PreparedSuccess {'1,'0,'Rollback},'PreparedSuccess > | < 'Preparing {'1,
    '0,'Commit},'Preparing > | < 'Preparing {'1,'0,'Rollback},'Aborting >
```

图 6　WS-AT 模型检验实验结果 5：非平凡性

5.2　实验结论

从上面的实验结果我们得到以下结论：采用 LTL 验证 WS-AT 在多参与者实施两阶段提交策略情况下的安全性和活性，整个状态空间达到 38187 个状态，WS-AT 满足稳定性、一致性和非平凡性，但是不满足非阻塞性.

非阻塞性是两阶段提交协议的主要问题，在没有超时机制情况下，协议一定是阻塞的. 但是加入超时机制，WS-AT 还是阻塞的，这是一个不好的性质. 在分析其违反非阻塞性的反例后，我们了解到其原因. 因为把注册和协调混在一起，使得协议无法按照标准的两阶段提交协议完成，这个是 WS-AT 协议的制定者需要以后改进的地方. 可以看出，细胞膜演算不但形式化地描述了 WS-AT 协议，发现了其不规范和含糊的地方，而且采用模型检验技术，验证了协议的安全性和活性，并给出以后协议需要改进的方向. 可以说是建立了关于原子事务比较完整的框架，从形式化描述到模型检验，读者不难将细胞膜演算推广到其它原子事务协调协议的描述和验证中去.

5.3　与 TLA+的比较

细胞膜演算和 TLA+一样，均对 WS-AT 做了形式化方面的工作. 下面就形式化描述和模型检验方面做一比较. 下面的分析以文献[1]中提到的 TLA+对 WS-AT 的分析为对照.

在形式化描述方面，两者均对 WS-AT 进行了分析，给出了协议的形式化规范. TLA+给出了两种模型，一种是基于消息传递的详细模型，一种是基于共享内存的抽象模型. 抽象模型是用来做高层次分析的，简化程度很高，我们以基于消息传递的详细模型来比较. 在 TLA+中注意到两种协调方式的区别，但是其协调方式比较抽象，没有严格按照协议规定的消息转换表处理，而且没有注意到注册和协调协议的不一致性. 由于对客户端协议过于简化，没有客户端的完成协议与协调者交互，因而没有处理很多的内部细节.

细胞膜演算很多方法从 TLA+中得到启发，也是采用了内部数据结构记录参与者状态，采用异步消息传递机制交互. 注意到应用程序与协调程序的交互以及注册和协调协议的交互，而且严格按照消息转换表处理各种消息，其描述的细节较 TLA+为多，而且正如上文所述，在做形式化的过程中，发现了协议许多不规范和混淆的地方，这些情况在 TLA+中是忽略掉的，细胞膜演算可以发现自然语言描述的协议中不清楚的地方，给出严格的语义.

在模型检验方面，TLA+仅仅给出了一致性验证，对于其它安全性和活性没有更多分析，而细胞膜演算对稳定性、一致性、非阻塞性、非平凡性等做了比较完整的分析. 并不是说 TLA+不能做其它方面的分析，但是到目前为止，没有在文献中发现对其它性质的分析.

6　结　论

本文采用细胞膜演算形式化描述 Web 服务中的原子事务协调协议 WS-AT. 讨论了 WS-AT 细胞膜演算形式化模型，接着引入 WS-AT 模型检验方法，定义该协议的安全性和活性，并采用重写逻辑表示这些性质，最后采用 Maude 工具自动进行模型检验. 实验结果表明：WS-AT 满足稳定性、一致性和非平凡性，但是不满足非阻塞性，并且也说明注册和协调协议的混杂是其不满足非阻塞性的原因，为协议的完善指明了方向. 本文最后给出了细胞膜演算与 TLA+的比较，说明了我们采用的方法的优点. 下一步的研究方向是继续完善基于细胞膜演算的事务形式化方法，并与制定 Web 事务标准的国际组织合作，对其所开发的提交协议给出形式化分析的结果，并对嵌套事务、动态事务以及混合事务的协议进行活性与安全性模型检验.

参 考 文 献

1 Johnson J. E., Langworthy D. E., Lamport L. *et al*. Formal specification of a Web services protocol. In: Proceedings of the 1st International Workshop on Web Services and Formal Methods, Pisa, Italy, 2004, 147～158

2 Jacinto R., Juanole G., Drira K.. On the application to OSI-TP of a structured analysis and modeling methodology based on Petri net models. In: Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems, Lisbonne, Portugal, 1993, 404～410

3 Qi Zheng-Wei, Mao Hong-Yan, You Jin-Yuan. The formal specification of transaction processing in Web services by rewriting logic. Chinese Journal of Computers, 2005, 28(4): 661～666(in Chinese)
(戚正伟,毛宏燕,尤晋元. 基于重写逻辑的 Web 服务事务处理形式化描述. 计算机学报, 2005, 28(4): 661～666)

4 Tang Fei-Long, Li Ming-Lu, Huang Zhe-Xue, Wang Cho-Li. A transaction service for service grid and its correctness analysis based on Petri net. Chinese Journal of Computers, 2005, 28(4): 667～676(in Chinese)
(唐飞龙,李明禄,黄哲学,王卓立. 服务网格中的事务服务及基于 Petri 网的正确性分析. 计算机学报, 2005, 28(4): 667～676)

5 Bruni R., Laneve C., Montanari U.. Orchestrating transactions in Join calculus. In: Proceedings of the 13th International Conference on Concurrency Theory, Brno, Czech Republic, 2002, 321～337

6 Bruni R., Melgratti H. C., Montanari U.. Nested commits for mobile calculi: Extending Join. In: Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, Toulose, France, 2004, 563～576

7 Cardelli L., GordonA. D.. Mobile ambients. Theoretical Computer Science, 2000, 240(1): 177～213

8 Bruni R., Montanari U.. Transactions and Zero-Safe nets. Lecture Notes in Computer Science 2128, Springer, 2001, 380～426

9 Butler M., Ferreira C.. An operational semantics for StAC, a language for modelling long-running business transactions. Lecture Notes in Computer Science 2949, Springer, 2004, 87～104

10 Bruni R., Melgratti H. C., Montanari U.. Theoretical foundations for compensations in flow composition languages. In: Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Long Beach, California, USA, 2005, 209～220

11 Lin Hui-Min, Zhang Wen-Hui. Model checking: Theories, techniques and applications. Acta Electronica Sinica, 2002, 30(12A): 1907～1812 (in Chinese)
(林惠民,张文辉. 模型检测:理论、方法与应用. 电子学报, 2002, 30(12A): 1907～1812)

12 Qi Zheng-Wei, Fu Cheng, Shi Dong-Yu, You Jin-Yuan, Li Ming-Lu. Membrane calculus: A formal method for grid transactions. Lecture Notes in Computer Science 3251, 2003, 73～80

13 Păun Gh.. Computing with membranes. Journal of Computer and System Sciences, 2000, 61(1): 108～143

14 Qi Zheng-Wei. Membrane calculus: A new formal method for transaction processing[Ph. D. dissertation]. Shanghai Jiaotong University, Shanghai, 2005(in Chinese)
(戚正伟. 细胞膜演算:一种新的事务处理形式化方法研究[博士学位论文]. 上海:上海交通大学, 2005)

15 Păun Gh., Rozenberg G.. A guide to membrane computing. Theoretical Computer Science, 2002, 287: 73～100

16 Meseguer J.. Rewriting as a unified model of concurrency. In: Proceedings on Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, 1990, 384～400

17 Eker S., Meseguer J., Sridharanarayanan A.. The Maude LTL model checker. In: Proceedings of the 4th International Workshop on Rewriting Logic and Its Applications, Pisa, Italy, 2002, 143～168

18 Gray J., Lamport L.. Consensus on transaction commit. ACM Transactions on Database Systems, 2006, 31(1): 133～160

QI Zheng-Wei, born in 1976, Ph.D.. His main research interests include distributed computing, formal methods and transaction processing.

You Jin-Yuan, born in 1939, professor, Ph. D. supervisor. His main interests include distributed computing, object oriented methods and software engineering.

Background

The increased complexity of Web Services and the explosive growth of Web based applications have turned their design and construction into a challenging problem. Systematic, formal approaches to the analysis and verification can address the problems of this particular domain. Roberto Bruni proposed one kind of process algebra called Join Calculus to prescribe Orchestrating Transactions. Another semantic model, called Zero-Safe Nets, is based on the special Petri Nets and provides the unifying notion of transaction. These methods are not suitable to describe the dynamic transactions in Web Service. The authors have developed a new formal method called Membrane Calculus inspired by P-Systems. The main contribution of this paper is to continue this way to use Rewriting Logic semantic to verify the properties of transactions. Membrane Calculus is used to formally specify the behavior of the coordinator and participants and analyze the Safety and Liveness of WS-AT. In the feature, the authors will study the loose-coupled dynamic transactions in Grid environment.