基于CPN的服务交互行为关键属性的运行时确保机制

朱 俊[1],郭长国[1,2],吴泉源[1]
(1. 国防科技大学计算机学院,湖南长沙 410073;2. 中国电子设备系统工程公司,北京 100039)

摘 要: 为了保证服务交互行为与其定义的关键属性相一致,本文提出了一种基于有色Petri网模型的运行时确保机制,从而提高服务组合运行的可靠性.文章首先介绍了服务交互行为CPN模型,用于精确刻画服务交互行为及其重要属性.结合服务实例,进一步对模型可达集和关键监控属性展开深入分析.在可达集的基础上,描述将模型运用于运行时监控的方式确保服务交互行为的关键属性.同时,本文从并行化执行流程和交互行为检测算法两方面介绍了服务交互行为的运行时监控机制.最后,评测结果表明这种运行时确保机制在性能、检测效率等方面都具有良好的表现.

关键词: 运行时确保; 服务交互行为; 关键属性; 有色Petri网
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2011) 05-1064-08

A CPN Based Runtime Assurance Mechanism for Critical Properties of Services Interactive Behaviors

ZHU Jun[1], GUO Chang-guo[1,2], WU Quan-yuan[1]
(1. *Department of Computer Science, National University of Defense Technology, Changsha, Hunan* 410073, *China*;
2. *China Electronic Equipment and Systems Engineering Co., Ltd., Beijing* 100039, *China*)

Abstract: To assure the accordance of services interactive behaviors and defined critical properties, this paper provides a CPN based runtime assurance mechanism. A service interactive behaviors CPN model is introduced to analyze reachability sets and critical properties of a composed service sample. It also depicts the mechanism on how to runtime monitoring these properties. At the end, evaluations show that this runtime assurance mechanism has very good performance and efficiency.

Key words: runtime assurance; service interactive behaviors; critical property; colored Petri nets

1 引言

服务组合的本质是协调不同功能的服务实现相互之间的消息交互,从而配合完成更为复杂,更为高级的功能[1].组合服务进程需要与其他的伙伴服务以一系列交互操作序列的形式进行会话,例如,以一串有次序的消息序列的方式.这些服务交互行为的发生必须满足一定的关键性属性,如偏序关系属性,Liveness活性,Safety安全性等,否则服务将无法完成预定的功能.然而由于一部分恶意客户或者运行异常的伙伴服务的存在,服务交互行为的这些属性很容易受到侵犯,导致服务组合产生异常.

同时,在目前的服务运行环境中,还没有特别有效的方法来确保实际服务交互行为与定义的属性相一致.已有多种形式化方法,比如进程代数[2,9],状态机[3,10]和Petri网[4,5,11],被运用于理解和分析服务组合规约的语义,其内在目的是为了描述和验证其中的某些重要属性.而在属性确保机制方面,运行时监控在监控信息及时反馈方面的优势,使得其格外受到青睐.如文献[6~8,12]都是这类典型的监控方式,但是这些方法都没有关注于如何在运行时确保服务交互行为的重要属性.

针对这方面的不足,本文扩展性地重用了文献[8]中提出的模式映射和组合规则,用于实现本文提出的服务交互行为模型的建模,并进一步通过运行时监控的方法确保服务交互过程中服务交互行为与其定义的关键属性的一致性,从而提高服务组合执行的可靠性和可信性.本文首先介绍了一个服务组合实例,并提出了一个服务交互行为的有色Petri网(Colored Petri Nets, CPN)

收稿日期:2010-03-25;修回日期:2010-10-28
基金项目:国家自然科学基金(No.90818028);国家863高技术研究发展计划(No.2007AA010301);"核高基"重大专项(No.2009ZX01043-002-004)

模型.完成由实例到模型的映射后,进一步对模型的状态空间,可达集和关键属性进行了深入分析.在可达集的基础上,文章详细描述了通过将服务交互行为模型应用于运行时监控的方式保证服务交互行为中偏序关系属性,Liveness 属性和 Safety 属性.同时,也从运行时监控机制的并行化执行流程和服务交互行为检测算法两个方面描述了服务交互行为的运行时监控机制.最后,全面的评测表明这种针对服务交互行为关键属性的运行时确保机制在性能、检测效率等方面具有良好的表现.

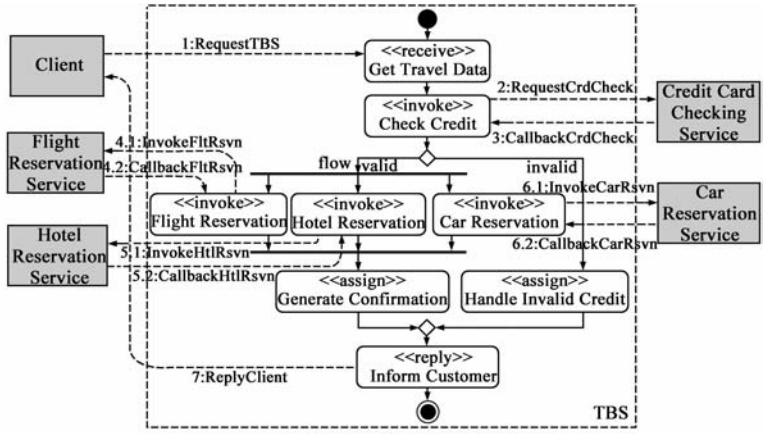


图1　TBS服务的基本流程和交互

2　TBS 实例

2.1　实例介绍

本文将一个服务旅行预订系统(Travel Booking System,TBS)作为案例进行研究.TBS 服务为用户提供了旅行所需要的各类预订功能.如图 1 所示,服务流程包括信用卡验证、机票/酒店/租车预订.用户在提交旅行计划的数据后,服务会调用信用卡验证服务,验证用户信用卡是否有效,是否存在充足余额.如果有效,TBS 服务就会并发调用预订服务.预订完成后创建反馈给用户的确认信息;如果验证失败,则转到处理无效信用卡的步骤.最后,TBS 将预订确认或失败的信息回复给用户.

2.2　实例中的服务交互行为属性

TBS 服务需要与若干伙伴服务进行交互,如 Client 向 TBS 服务发出请求;TBS 处理完后会返回预订确认信息等.整个交互过程涉及到多种类型的信息,包括同步,异步以及带确认的消息调用等.然而,服务交互行为需要满足一定的关键性属性约束,如,偏序关系属性,活性及安全性属性等.这些关键属性揭示了系统所期望发生的交互行为.然而,一般的静态检测难以很好地处理此类动态、复杂的服务交互行为,难以有效确保这些交互行为属性.当前的服务执行环境也缺少相应的机制来保证交互行为能够遵循系统的期望.因此,一方面需要一个模型来精确刻画交互行为及其属性;另一方面,还需要在运行状态下实时监视交互协议和属性有没有受到侵犯.

3　服务交互行为的 CPN 模型

3.1　服务交互行为模型

定义 1(服务交互行为的 CPN 模型).该模型的形式化表示采用一种 CPN 网 $N=(P,T,F,C)$,其中:

(1)P 为有穷的库所集合,T 为有穷的变迁集(和 P 不相交,即 $P\cap T=\varnothing$),弧 F 满足 $F\subseteq(P\times T)\cup(T\times P)$,而 C 代表一个有穷的颜色集.

(2)$P=P_I\cup P_M$ 且 $P_I\cap P_M=\varnothing$,其中 P_I 是内部的库所,而 P_M 为与服务进程中交互消息有关消息库所.

(3)颜色集 C 中的每一个颜色值代表了由一个客户端访问服务进程所产生的一个实例以及对应的交互会话.颜色集 C 用于区分不同的服务实例和交互会话过程.

(4)$F=F_M\cup F_I$,且 $F_M\cap F_I=\varnothing$,其中 F_I 为内部普通弧 $F_I\subseteq(P_I\times T)\cup(T\times P_I)$,而 F_M 是与消息库所 P_M 相连接的弧(消息弧)$F_M\subseteq F_M^R\cup F_M^S$. $F_M^R\subseteq(P_M\times T)$ 代表接收消息弧,而 $F_M^S\subseteq(T\times P_M)$代表发送消息弧.

通过运用文献[8]中提出的模式映射规则以及组合规则,生成如图 2 所示的服务交互行为的 CPN 模型 N_{TBS}.该模型对 TBS 服务与其他伙伴服务之间交互行为进行精确的形式化描述.模型有一类特殊的库所,用灰色标识的消息库所 P_M.通过分析该模型的状态空间,证明这个模型是有界的和安全 Safe 的.

3.2　模型的可达状态分析

定义 2(状态可达性).标识 m 可达,是指通过一个变迁发生序列 $\sigma=\{T0\cdots Ti\cdots\}\in T^*$(以 T 为字母表得到

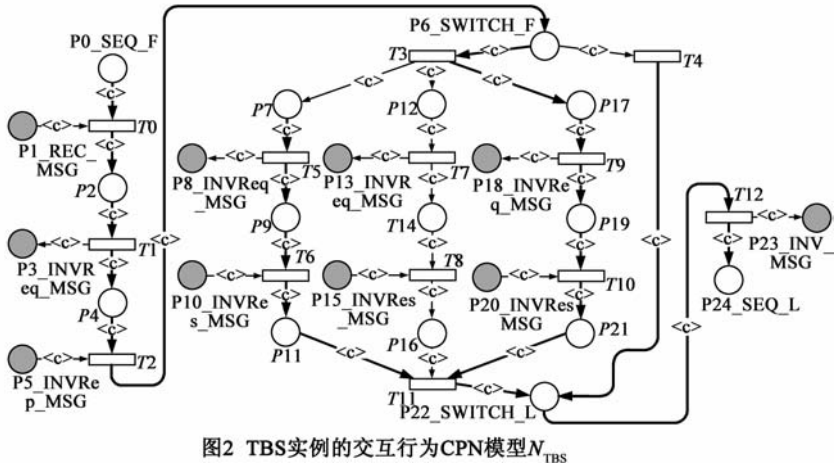


图2　TBS实例的交互行为CPN模型N_{TBS}

的集合 T^* 中的元素)能够到达该标识,记做 $m_0 \xrightarrow{\sigma} m$.那么,在一个 CPN 网模型 N 中,从初始标识 m_0 开始经过一定变迁发生序列能够到达的标识集合,被称为可达集,记为

$$RS(N, m_0) = \{m \mid \exists \sigma \in T^* . m_0 \xrightarrow{\sigma} m\}$$

将该模型的标识定义为 $\boldsymbol{m}_n = (P0, P1, P2, \cdots, P24)$,其中 Pi 对应于模型中库所 i 的状态.如 $P0$ 代表库所 $P0\_SEQ\_F$ 的状态.初始标识 $\boldsymbol{m}_0 = (1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0)$.则交互行为 CPN 模型 $<N_{TBS}, \boldsymbol{m}_0>$ 的可达图 $RG(N_{TBS}, \boldsymbol{m}_0)$,如图 3 所示.$S0$ 为初始状态所对应的标识 $\boldsymbol{m}_0$.可达状态 $S6$ 和 $S34$ 分别为服务流程中的两个终止状态.

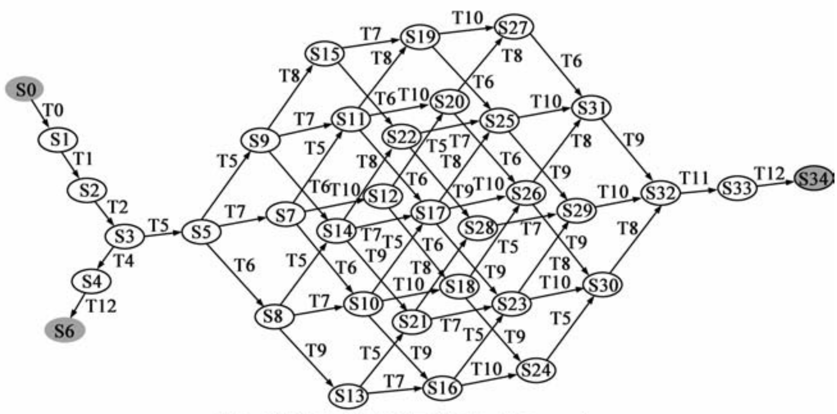


图3　模型 N_{TBS} 的状态可达图 $RG(N_{TBS}, m_0)$

弧线代表从一个状态标识转换到下一个可达状态标识所经历的变迁.初始状态可经过变迁 $T0$ 转换为状态 $S1$.而该模型的可达集

$$\boldsymbol{RS}(N_{\mathrm{TBS}}, m_0) = \begin{Bmatrix} (1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0) \\ (0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0) \\ \vdots \\ (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0) \\ (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1) \end{Bmatrix}$$

3.3　服务交互行为序列

定义 3(服务交互行为序列).假设 I 为某系统中的所有交互行为(或消息)的集合,即交互行为 $i_m \in I$.则定义服务交互行为序列 σ_m 为一个有限的、具有偏序关系的序列 $i_1 i_2, \cdots, i_m, \cdots i_n$,其中 $\forall i_m, 0 \leqslant m \leqslant n, i_m \in \sum$.

性质 1(服务交互行为序列与变迁发生序列的一致性).服务流程正常运行的情况下,监控系统捕获到的交互行为(消息)序列 σ_m 与服务中实际发生的变迁发生序列 $\sigma = \{T1 \cdots Tm \cdots Tn\}$ 揭示了相同的状态转换过程 $\sigma_S = \{S0 \cdots Sm \cdots Sn\} \in \boldsymbol{RS}(N, m_0)^*$,其中,$0 < m \leqslant n$.

在 TBS 实例中,如果系统发生了状态转换 $\sigma_S = \{S0 S1 S2 S3 S5 S9\}$,则根据图 3 的状态可达图 $RG(N_{TBS}, \boldsymbol{m}_0)$ 得出,必定存在一个变迁发生序列 $\sigma = \{T0 T1 T2 T3 T5\}$ 使得模型 $<N_{TBS}, \boldsymbol{m}_0>$ 的状态从初始状态 $S0$ 变迁到 $S9$,即 $\exists \sigma \in T^*, S0 \xrightarrow{\sigma} S9$.同时,这个状态变迁过程 σ_S 经历的服务交互序列为 $\sigma_m = \{P_M 1 P_M 3 P_M 5 P_M 8\}$.

性质 1 表明从系统运行过程中捕获的交互行为(或消息)序列揭示了系统实际发生的状态转换(执行轨迹).通过将服务交互行为序列作为监控证据,并结合对应的交互行为模型的期望进行一致性约束分析,就能够判断实际发生的状态变迁,从而确定当前的系统所处的状态.

4　服务交互行为关键属性的确保

偏序关系属性,Liveness 属性,Safety 属性都是攸关分布式服务系统可靠性的重要属性.而服务交互行为 CPN 模型能够精确地刻画这些属性,从而能够通过运行时监控机制进一步确保实际交互行为满足监控属性的约束.

4.1　偏序关系属性的确保

定义 4(服务交互行为的偏序关系).服务交互行为的严格或非严格偏序关系是在给定交互行为集合 $I = \{i_m\}$ 上二元关系(用操作符 $<_i$ 表示严格偏序关系,用操作符 $\leqslant_i$ 表示非严格偏序关系).如果服务交互行为 i_1 的发生严格优先于交互行为 $i_2(i_1, i_2 \in I)$,则表示为 $i_1 <_i i_2$;如果交互行为 i_1 和 i_2 是并发关系,则其关系表达为 $i_1 \leqslant_i i_2 \cup i_2 \leqslant_i i_1$.

定义 5(服务交互行为序列的偏序关系属性).服务交互行为序列 σ_m 满足如下属性:$\forall i_p, i_q, 0 \leqslant p < q \leqslant n$,则有 $\sigma_m = \{i_0 \cdots i_p \cdots i_q \cdots i_n\}$ 且 $i_p <_i i_q$.

为了协调各个服务,服务交互行为需要满足一定严格或非严格偏序关系属性.图 1 反映出 TBS 服务具有如下的交互场景:首先,由 Client 向 TBS 服务发出请求消息,产生交互行为"1:RequestTBS"(i_1);然后 TBS 服务向信用卡验证服务发出对用户信用卡进行验证的请求,产生交互行为"2:RequestCrdCheck"(i_2);信用卡验证服务再根据查询结果返回有关用户信用卡是否有效的

消息“3:CallbackCrdCheck”(i_3).这三个交互行为之间必须满足严格偏序关系:$i_1 <_i i_2 <_i i_3$.采用 UML2.0 顺序图可以把这种关系表达成如图 4(a)所示.而用交互行为模型可以进一步将这种关系精确刻画成如图 4(b).

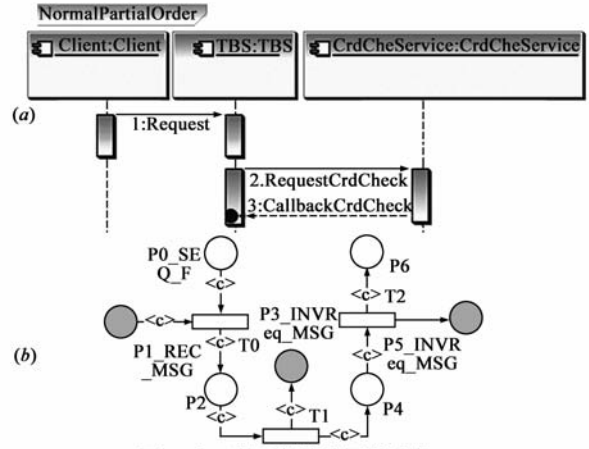


图4　交互行为的偏序属性确保

如果在实际监控过程中捕获的交互行为序列是 $\sigma_m = \{P_M1P_M3P_M5\}$,则根据性质 1 以及交互行为模型可以得到该过程中实际对应出现的变迁发生序列是 $\sigma = \{T0T1T2T3\}$.由此,结合实例模型的可达图 $RG(N_{TBS}, m_0)$进一步推断,系统经变迁发生序列 σ 从状态 $S0$ 运行到了状态 $S5$,即 $S0 \xrightarrow{\sigma} S5$.从而得出结论:系统实际的运行以及在到当前状态为止所产生的服务交互行为序列都是与该服务组合系统的规约保持一致的.假设监控到的交互行为序列是 $\sigma_m = \{P_M3P_M1P_M5\}$,则可知在该模型中并不存在这样的变迁发生序列能够实现可达状态之间的转换,而产生出与 σ_m 一致的交互行为序列.

尽管如此,不是所有的服务交互行为都需要满足偏序关系属性,特别是并发的服务交互行为.例如实例 TBS 中的“4.1:InvokeFltRsvn”($i_{4.1}$),“5.1:InvokeHtlRsvn”($i_{5.1}$)和“6.1:InvokeCarRsvn”($i_{6.1}$)这三个交互行为,不需要满足偏序关系属性,即,$i_{4.1} \leqslant_i i_{5.1} \cup i_{5.1} \leqslant_i i_{4.1}$,$i_{5.1} \leqslant_i i_{6.1} \cup i_{6.1} \leqslant_i i_{5.1}$,$i_{4.1} \leqslant_i i_{6.1} \cup i_{6.1} \leqslant_i i_{4.1}$.

4.2　Liveness 监控属性的确保

定义 6(变迁的潜在发生权).如果存在一个变迁发生序列 σ 使之达到标识 m',且变迁 t 在标识 m' 下具有发生权(即,$m \xrightarrow{\sigma} m' \geqslant$向前关联矩阵 $Pre[P,t]$),则称变迁 t 在给定的标识 m 下具有潜在发生权.

定义 7(Liveness 属性).在模型 N 中,如果变迁在任一可达标识($m' \in RS(N, m_0)$)下都具有潜在发生权($\exists\, m' \in RS(N, m_0), m_0 \xrightarrow{\sigma} m' \geqslant Pre[P,t]$),则称变迁具有活性.

一旦系统到达某个可达状态后,如果没有后继的变迁能够点火发射,就意味着进入死锁状态.而 Liveness 属性能够保证系统不会进入或者部分进入死锁状态.在 TBS 服务中,不管预订过程成功与否,用户最后都应该收到与旅行预订状态相关的信息.否则用户将不知道预订结果怎样,也不知道预订的过程何时才会结束.这是一个典型的有关变迁活动是否具有活性的问题.

这个 Liveness 属性用 UML2.0 顺序图表达为如图 5(a)所示.该图用在 UML2.0 中新扩展的操作子 assert 来描述一种必须满足的断言.在该实例中,断言表明交互行为“1:RequestTBS”发生后的一段时间内,Client 必须接收到来自 TBS 服务的反馈确认行为“7:ReplyClient”(i_7).这个 assert 断言本身不关注这两个交互行为之间具体发生的其他交互行为,但是必须确保这两个交互行为先后依次出现.在交互行为模型中,该 Liveness 属性的断言约束展现为图 5(b).虚线框代表省略的状态变迁过程.

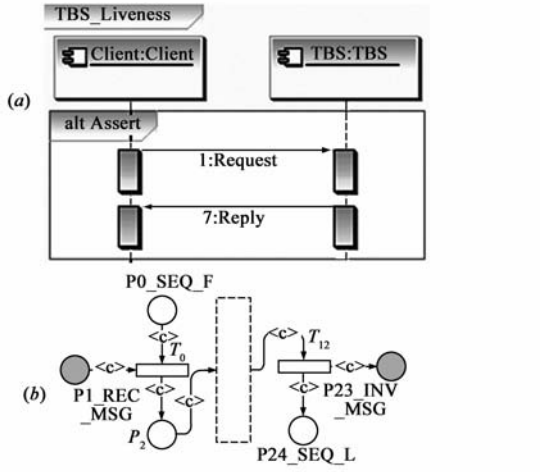


图5　交互行为的Liveness属性的确保

由可达图 $RG(N_{TBS}, m_0)$可知,从初始状态 $S0$ 开始,变迁 $T1$ 因为接收到 Client 的服务请求而触发.即 $S0 \rightarrow S1$ 的状态变迁触发了交互行为“1:RequestTBS”.而两个终止状态 $S6$ 和 $S34$ 都可由变迁 $T12$ 触发而达到.变迁 $T12$ 触发的结果是产生了发送确认信息“7:ReplyClient”给 Client 的交互行为.而在该模型的变迁发生序列上能够保证满足 $\sigma = \{T0 \cdots T12\}$(其中,$T1 < T12$),从而确保 $i_1 <_i i_7$.

同时,为了达到任意一个终止状态,必须触发变迁 $T12$,即必须产生交互行为 i_7,以完成整个服务流程.在这种情况下,当监控机制捕获到交互行为 i_7 时,就可以判断当前服务流程的执行是满足这条 Liveness 属性约束.

4.3　Safety 监控属性的确保

定义 8(Safety 属性).给定一个模型 $N = (P, T, F, C)$,假设 s 为模型中一个 Safety 约束属性,而假设标识

m'代表该 Safety 约束的否定所对应的状态标识.则该 CPN 网模型是 Safe 安全的,当且仅当不存在这样的变迁发生序列使得状态达到标识 m',即 $\sim\exists\,\sigma_s, m_0 \xrightarrow{\sigma_s} m'$.

Safety 监控属性就是保证在系统运行过程中坏的事情不会发生.实例中一个典型的 Safety 属性是:必须满足 TBS 服务在预订机票之前验证用户的信用卡.只有在信用卡有效的情况下,才能够进行机票预订等服务的调用操作.否则,就会产生预订成功后,因为信用卡有效性问题而导致无法完成支付,最终造成服务商的商业损失.该 Safety 属性约束用 UML2.0 顺序图表示如图 6(a)所示.

图 6(a)用 UML2.0 中新扩展的操作子 neg 描述那些不希望发生的交互场景.例如在 TBS 服务发出预订机票的请求之后,跨过验证信用卡有效性这一关键步骤而直接调用机票预订服务的交互场景是不允许发生的.

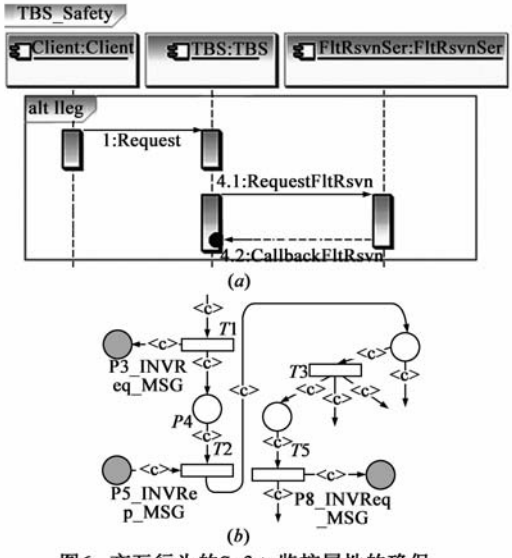


图6 交互行为的Safety监控属性的确保

图 6(b)为从服务交互行为模型中提取的片段.这个模型片段表明能够检测出所有不经过信用卡有效性验证步骤而直接进行预订服务操作的违背交互协议的问题.假设监控系统捕获到消息"4.1:RequestFltRsvn"(当前的模型标识为 m'),则由模型可知存在一个变迁发生序列 $\sigma=\{\cdots T1\,T2\,T3\cdots T5\}$,使得 $m_0 \xrightarrow{\sigma} m'$.从而确认监控系统应该捕获到的交互行为序列满足 $\sigma_m=\{P_M1\,P_M3\,P_M5\cdots P_M8\}$,就是监控器检测结果正确的前提就是在捕获到交互行为"4.1:RequestFltRsvn"之前必定已经先后捕获到了行为"2:RequestCrdCheck"和"3:CallbackCrdCheck".如果接收到的交互行为序列为 $\sigma_m'=\{\cdots P_M1\,P_M8\}$,那么监控器就能检测出这个违背 Safety 属性的问题.因为在这种情况下消息 P_M1 之后下一个期望到达的消息是 P_M3.

5 服务交互行为的运行时监控机制

5.1 服务交互行为捕获机制

服务组合实例的执行依赖于 SOAP 消息引擎、服务组合执行引擎等.为了能够捕获实时的服务交互行为,通过扩展 SOAP 消息引擎的方式实现了对交互会话过程中的 SOAP 消息进行捕获.SOAP 消息引擎处理所有与 Web 服务交互中的 SOAP 消息.在本文的方法中,一个 SOAP 消息监视器被嵌入到 SOAP 消息引擎中.SOAP 消息监视器能够实时转发当前被 SOAP 引擎处理的 SOAP 消息.每一个 SOAP 消息都被转发到与 SOAP 消息监视器建立连接的交互行为运行时监控器上.该监控器对陆续到达的交互消息进行检测,同时评估组合服务当前的运行状态.

5.2 运行时监控机制的并行化执行流程

如图 7 所示,监控器首先需要进行初始化操作,等待接收即将到来的服务交互行为事件 e.当交互行为事件 e 到达时,监控器将根据事件 e 所属的组合服务实例搜索"实例-监控线程"映射表.如果表中存在行为事件 e 所对应的组合实例,那么查找出对应的监控线程;如果不存在,则创建一个新的监控线程,同时将新的"实例-监控线程"项更新到映射表.交互行为事件被分派到对应的监控线程,并触发了线程中的交互行为检测算法(见 5.3 节).经过交互行为检测算法处理后将得到两种结果:如果检测结果正常,这说明对应的服务组合实例的交互行为满足各类属性的要求,则进一步更新当前的监控状态,同时进入到等待接收下一个服务交互行为事件到达的状态;如果检测结果异常,这说明当前的服务交互行为侵犯了某一个关键属性,则需要激活相应的异常处理过程.

5.3 交互行为检测算法

交互行为检测算法基于 Web 服务交互行为 CPN 模

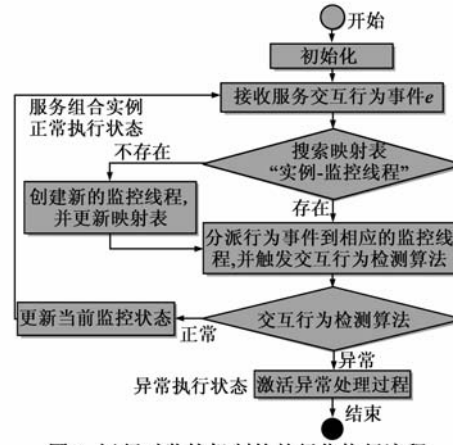


图7 运行时监控机制的并行化执行流程

型 N_{CPN}.定义 1 决定服务交互行为 CPN 模型中的消息库所 P_M 与服务交互行为之间满足双射的映射关系.依据这种双射关系,算法主要通过检测由 SOAP 监视器转发的交互消息和模型 N_{CPN}中期望的后继交互行为事件 P_M 是否一致来实现.假设某组合服务当前处于模型 N_{CPN}中的某一可达状态 $s(s \in \boldsymbol{RS}(N_{CPN}, m_0))$,则依据模型计算得到的期望到达的后继交互行为事件集合($E \subseteq (s\cdot)\cdot$).如果到达的交互行为事件 e 是该期望后继交互行为事件集合的元素之一,即满足 $e \in E$,则说明组合服务执行状态正常,同时将模型的当前状态变量更新到新的状态,并等待新的交互行为事件的到达;否则,则说明到达的交互事件并不是模型预期的($e \notin E$),组合服务系统出现异常,同时监控流程将进入异常处理过程.算法描述如算法 1 所示.

算法 1　服务交互行为的检测算法

输入:当前状态标识 *current_state* 和当前捕捉到的交互行为事件 *e*;
输出:更新后的状态标识 *current_state*;

```
1. switch(current_state)://匹配当前状态标识;
2.   case state_1:      //当前状态 current_state 与状态 state_1 匹配
     成功(current_state = state_1∈P.P 为可达状态标识集合);
3.     计算当前状态标识 current_state 下预期到达的后继交互行为
       事件集合 E⊆(current_state·)·;
4.     if(e = = event_1)//匹配当前到达事件 e 与期望后继交互行
       为事件 event_1(event_1∈E);
5.       计算出在接收到事件 event_1 后跃迁至的后继状态 p′;
6.       current_state = p′;  //更新当前状态标识 current_state 为
         新状态标识 p′;
7.       break;
8.     else if(e = = event_2) //匹配当前到达事件 e 与期望后继交
       互行为事件 event_2 (event_2∈E);
9.       …      //过程类同"event_1";
10.    else            //预期到达事件匹配失败(e ∉ E),Web 服务
       交互行为出现异常;
11.      throw Exception - handler;      //检测出交互行为异常,
         进入异常处理过程;
12.  case state_2:      //当前状态 current_state 与状态 state_2 匹配
     成功(current_state = state_2∈P);
13.    …  //过程类同"case state_1";
14.  default:           //当前状态匹配失败(current_state ∉ P);
15.    throw Exception - handler;      //当前状态标识不存在,进入
       异常处理过程;
16. return current_state;      //结束当前状态的匹配,返回更新后的
    当前状态;
```

5.4　算法的计算复杂性分析

设组合服务实例有 n 个状态,有 m 个($0 < m < n$)服务交互行为.则 switch 结构产生计算复杂性$\frac{1}{2}n$,因为 case 项的数量取决于系统可检测到的状态数.每一个 case 项内部($L2 \sim 11$)具有类似的计算逻辑:第 3 行代码计算预期到达的后继交互行为事件集合 $E \subseteq (current_state\cdot)\cdot$的计算复杂性为 m;第 4 ~ 11 行代码段为 if 结构,其计算复杂性为$\frac{1}{2}m$;第 5 ~ 7 行代码段的计算复杂性为常数.于是,交互行为检测算法的计算复杂性为 $\frac{1}{2}n \times (m + \frac{1}{2}m) = O(\frac{1}{2}n \times \frac{3}{2}m) = O(nm)$.由此可知,该服务交互行为检测算法具有较高的执行效率.

6　评测

6.1　服务交互行为的 CPN 模型评估

TBS 服务实例将作为测试用例.服务交互行为模型 N_{TBS}有 25 个库所,13 个变迁以及 40 条弧.因此,N_{TBS}具有规模小,结构紧凑,状态空间较小的特点,从而避免了状态空间爆炸的问题.同时,监控器具有 148 行代码,有利于获得较高监控检测效率以及产生相对较小的负载.

6.2　监控效率评估

通过模拟交互消息以不同顺序到达的方式,我们测试了监控器在检测服务交互行为的各类监控属性方面的效能.违背三类属性的消息子序列将按照如表 1 所示的组合方式和百分比率构成交互消息序列.该测试设计了三类违背属性的消息子序列集合($\{Par\}$,$\{Liv\}$和$\{Sav\}$分别为违背偏序属性、活性属性和安全性属性的消息子序列集合)和一类正确的消息子序列集合$\{Cor\}$.假设完整的消息序列用Σ表示(消息总数为 10^5个),而在完整的消息序列中三类违背属性和一类正确的消息子序列的百分比率分别为 P_{Par},P_{Liv},P_{Sav}和 P_{Cor},则有 $P_{Par} + P_{Liv} + P_{Sav} + P_{Cor} = 100\%$,即一串完整的消息序列$\Sigma$是由分别从集合$\{Par\}$,$\{Liv\}$,$\{Sav\}$和$\{Cor\}$中随机选出的 $10^5 P_{Par}$,$10^5 P_{Liv}$,$10^5 P_{Sav}$和 $10^5(1 - P_{Par} - P_{Liv} - P_{Sav})$个消息子序列所组成.根据属性的不同组合方式,测试类别划分三类:单一属性,两类属性组合和三类属性组合.即消息序列Σ含有违背某一类、两类或三类关键属性的消息子序列.

表 1　监控效率评估中违背各类属性的消息子序列的百分比率

测试类别	消息子序列比率分配	检错效率
单一属性	P_{Par}或P_{Liv}或P_{Sav} = 10%;30%;50%.	100%
两类属性组合	(P_{Par},P_{Liv})或(P_{Liv},P_{Sav})或(P_{Par},P_{Sav}) = (5%,5%);(15%,15%);(25%,25%).	100%
两类属性组合	(P_{Par},P_{Liv})或(P_{Liv},P_{Sav})或(P_{Par},P_{Sav}) = (5%,45%);(15%,35%);(35%,15%);(45%,5%).	100%
三类属性组合	(P_{Par},P_{Liv},P_{Sav}) = (5%,5%,5%); (10%,10%,10%);(15%,15%,15%)	100%
三类属性组合	(P_{Par},P_{Liv},P_{Sav}) = (40%,5%,5%); (5%,40%,5%);(5%,5%,40%)	100%
三类属性组合	(P_{Par},P_{Liv},P_{Sav}) = (25%,25%,5%); (25%,5%,25%);(5%,25%,25%)	100%

经过乱序化处理后的消息序列Σ将作为监控器的

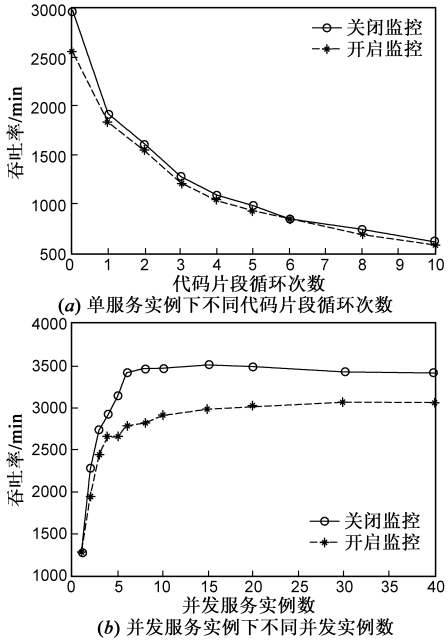
输入.通过对监控器输出的检测结果与实际消息序列作对比,得出了监控器的检错效率.测试结果证明监控器对偏序属性错误,Safety 属性错误等具有 100% 的检错效率.

6.3 负载评估

监控负载和性能评估都采用了如下的实验环境和条件:若干 Dell Optiplex755 商用台式机(配有 2.53GHz 的 Intel 双核 CPU E7200 和 2GB DDR2 SDRAM);操作系统采用 Ubuntu 9.10(Linux 内核版本:2.6.31).服务运行环境采用:JRE 1.6.0 和 Apache ODE 1.3.3.另外,将 Apache Jmeter 2.3.4 作为客户端测试工具.

在关闭和开启监控器两种情况下,记录服务器的系统吞吐率作为性能指标,从而通过比较这两种情况下服务器的性能差异来评估监控器执行所产生的实际负载.

(1)单服务实例下的负载评估 在该测试中,不同执行时间的服务被作为测试用例.不同的执行时间通过对同一代码段循环执行不同的次数来实现.该代码段在当前实验环境中的执行时间约为 8ms.实验评估结果如图 8(*a*)所示,并得出以下结论:(1)在相同的循环执行次数条件下,开启监控器时的系统吞吐率比关闭的情况下略有减少;(2)随着服务执行时间的增加,系统的吞吐率逐渐减小;(3)在各种不同的执行时间下,开启监控器所引起的吞吐率减少量都比较有限(可观察到的负载对系统吞吐率的影响保持在 5.5% 以内).



(*a*) 单服务实例下不同代码片段循环次数

(*b*) 并发服务实例下不同并发实例数

图8 监控器负载评估

(2)并发服务实例下负载评估 在并发服务实例负载测试中,循环代码段执行次数设定为 3 的服务流程被作为测试用例.并发进程实例的数量从 1 到 40.测试结果如图 8(*b*)所示,分析可以得出结论:(*a*)开启监控器时的吞吐率略有减少;(*b*)随着并发数量的增多,吞吐量也逐渐增加.当并发会话的数量达到 10 左右时,系统的服务能力开始达到饱和状态.监控器执行所产生的负载量也逐渐达到稳定;(*c*)监控器执行产生的负载对系统吞吐率的影响是在 10.8% 范围之内.

总之,不管在单进程还是并发进程的情况下,监控器执行所产生的负载较小,负载对系统效率的影响也较小.

6.4 性能评估

监控器性能评估,通过计算每次交互消息到达监控器到获得检测结果的时间长度,统计得到交互行为的平均检测时间,并以此作为对监控机制的性能进行评估的基本指标.经过 10^5 次对 TBS 流程服务的请求,获得图 9 所示的在多个并发用户并发请求下的性能评估曲线.

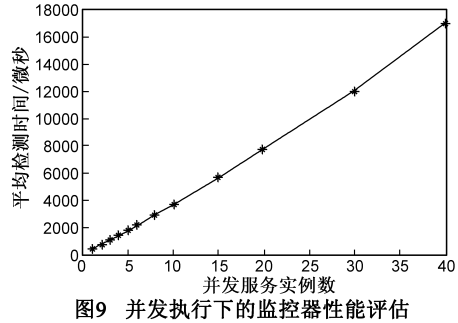


图9 并发执行下的监控器性能评估

从该曲线得出以下结论:(1)监控器的平均检测时间较短(10^{-4} ~ 10^{-2}s),性能较高,满足了运行时监控的需求.(2)平均检测时间随着并发实例数量的增加而增加,且平均检测时间的增长率也随着并发服务实例数量的增加而增加.平均检测时间与并发实例数量近似满足幂函数关系:$t_{check} = a \times x^n$,其中,$a \in R^+$,$1 < n < 2$,x 为并发实例数量.这种关系主要是由服务交互行为事件的分发机制所产生的.较多的并发实例意味着同一时刻有较多的交互消息等待检测,这使得每个交互行为的检测时间有所增加.

性能与负载评估测试是在相同的测试环境下进行.由负载评估的结果可知,监控器执行所产生的负载较小,占有较少的计算资源.而服务实例本身执行所产生的负载却很大,占有相当大的计算资源.当并发实例数量大于 10 时,系统吞吐率达到饱和,如图 8(*b*)所示.此时系统只能够为特定数量的并发用户提供服务.于是,当系统达到饱和状态时,相同的吞吐率决定了相同的服务系统负载和监控器负载,也决定了相同的交互行为平均检测时间.然而监控器在 1 ~ 10 个并发访问服务实例的情况下(未达到饱和状态),其平均检测时间在 3.9×10^{-4} 和 4.2×10^{-3} 之间,达到了实时监控的要

求.综上所述,当系统达到饱和状态之前,高效且低负载的监控器能够满足实时监控的要求.

7 结束语

为了有效地实时确保实际服务交互行为与定义的关键属性相一致,本文提出了一种基于服务交互行为 CPN 模型的运行时确保方法,用于实时检测服务交互行为属性.同时,文章深入分析了模型的状态空间,可达集以及关键属性,并侧重描述了通过运行时监控的方式确保服务交互行为关键属性的基本机制.本文也对服务交互行为的运行时监控机制进行了深入的探讨.最后,通过全面的测试评估验证了这种机制具有良好的检测效果.

参考文献

[1] OASIS. Web Services Business Process Execution Language Version 2.0 (WS-BPEL 2.0)[S].

[2] G Salaun, L Bordeaux, M Schaerf. Describing and reasoning on Web services using process algebra[J]. International Journal of Business Process Integration and Management, 2006, 1(2): 116 – 128.

[3] G Diaz, J Pardo, M Cambronero, et al. Automatic translation of WS-CDL choreographies to timed automata[A]. Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM'05)[C]. Springer, 2005. 230 – 242.

[4] W Tan, Y Fan, M Zhou. A Petri net-based method for compatibility analysis and composition of Web services in business process execution language[J]. IEEE Transactions on Automation Science and Engineering, 2009, 6(1): 94 – 106.

[5] H Verbeek, W van der Aalst. Analyzing BPEL processes using Petri nets[A]. Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management[C]. US: Florida International University, 2005. 59 – 78.

[6] F Barbon, P Traverso, et al. Run-time monitoring of instances and classes of Web service compositions[A]. Proceedings of the 4th IEEE International Conference on Web Services[C]. US: IEEE Computer Society, 2006. 63 – 71.

[7] Z Li, Y Jin, J Han. A runtime monitoring and validation framework for Web service interactions[A]. Proceedings of the 2006 Australian Software Engineering Conference[C]. US: IEEE Computer Society, 2006. 70 – 79.

[8] Jun Zhu, Fabrice Kordon. A Petri net based runtime monitoring method for Web services specified with BPEL[A]. Proceedings of the 2nd International Conference on Information Management and Engineering[C]. US: IEEE Computer Society, 2010. 304 – 310.

[9] 廖军,谭浩,等.基于 Pi-演算的 Web 服务组合的描述和验证[J].计算机学报,2005,28(4):635 – 643.
Liao Jun, Tan Hao, et al. Describing and verifying Web service using pi-calculus[J]. Chinese Journal of Computers, 2005, 28(4): 635 – 643. (in Chinese)

[10] 雷丽晖,段振华.一种基于扩展有限自动机验证组合 Web 服务的方法[J].软件学报,2007,18(12):2980 – 2990.
Lei Li-Hui, Duan Zhen-Hua. An extended deterministic finite automata based method for the verification of composite Web services[J]. Journal of Software, 2007, 18(12): 2980 – 2990. (in Chinese)

[11] 闫春钢,蒋昌俊,等.基于 Petri 网的 Web 服务组合与分析[J].计算机科学,2007,34(2):100 – 104.
Yan Chun-Gang, Jiang Chang-Jun, et al. The composition and analysis of Web service based on Petri net[J]. Computer Science, 2007, 34(2): 100 – 104. (in Chinese)

[12] 陈今梁,吴国全,魏峻.一种 WS-BPEL 流程的运行时监控方法[J].电子学报,2007,35(B12):174 – 178.
Chen Jin-liang, Wu Guo-quan, Wei Jun. An approach for runtime monitoring of WS-BPEL processes[J]. Acta Electronica Sinica, 2007, 35(B12): 174 – 178. (in Chinese)

作者简介

朱　俊　男,1981 年生于浙江嘉兴.国防科技大学计算机科学与技术专业博士生.2006 年获得计算机科学与技术专业硕士学位.研究方向为软件工程、可信软件和分布式计算等.
E-mail: mail_zhujun@nudt.edu.cn

郭长国　男,1973 年生于河南武陟.博士,副教授,主要研究领域为分布式计算,中间件等.

吴泉源　男,1942 年生于上海.教授,博士生导师,主要研究领域为分布式计算,人工智能和专家系统等.