基于规范流网的Web服务行为适配方法研究

曹国荣 谭庆平 解金刚 吴 浩
(国防科技大学计算机学院 长沙 410073)

摘 要 Web服务适配是面向服务计算领域的重要研究内容。针对现有服务行为建模和适配技术在循环服务行为、数据流建模和状态空间爆炸方面存在的问题,提出一种新的服务行为建模和适配方法,并结合实例阐述该方法如何以规范流网为基础,建模服务行为,构造服务行为的符号化可覆盖树,构建数据依赖关系和动作依赖关系,构建符号化执行轨迹适配器,直至最后完成服务行为适配的整个建模和适配过程。

关键词 规范流网,服务适配,符号化可覆盖树,数据依赖,动作依赖,适配器
中图法分类号 TP393 **文献标识码** A

Research on Web Service Behavior Adaptation Based on Regular Flow Nets

CAO Guo-rong TAN Qing-ping XIE Jin-gang WU Hao
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract Web service adaptation is an important research focus in the field of service-oriented computing. Aiming at problems of service behavior formalization and adaptation in cyclic service behavior, data flow modeling and state space explosion, we proposed a new approach for service behavior formalization and adaptation. The whole process of how to formalize service behavior based on regular flow nets, how to construct symbolic coverability tree of service behavior, how to build data dependence and actions dependence relationships, how to generate symbolic execution trace adapters, till accomplishing service behavior adaptation was illustrated through examples in the paper.

Keywords Regular flow net, Service adaptation, Symbolic coverability tree, Data dependence, Action dependence, Adapter

1 引言

Web服务适配是面向服务计算领域的重要研究内容。它旨在解决服务组合过程中的各种不匹配问题。目前,服务不匹配问题仍是面向服务的软件工程开发思想实用化的最主要障碍之一。面向服务的软件工程开发技术是软件开发技术发展的必然选择,因此服务不匹配问题是一个无法绕过的技术难点。服务适配的提出为服务不匹配问题提供了一条切实可行的解决途径。

现有的服务适配研究主要包括语法适配、行为适配、语义适配和非功能性适配。语法适配着重从语法上构建服务接口模型,解决服务在方法名、参数名、类型名、复杂类型数据结构、方法类型、参数类型、异常类型、参数顺序和参数个数等不一致情况下的失配问题;行为适配针对服务间的交互行为构建服务接口模型,解决服务在消息交互顺序不一致、消息冗余和消息缺失等情况下出现的失配问题;语义适配将服务接口语义理解为本体,构建基于本体概念的服务接口模型,解决因服务语义描述不一致或者服务功能不完全满足系统需求而引发的失配问题,如概念近似、概念包含和概念等价;非功能性适配,也称为质量适配,基于服务的质量属性构建服务接口模型,解决服务间在安全性、持久性、事务性、可靠性和执行性能等不一致情况下的失配问题。本文关注Web服务行为方面的适配研究。

对服务行为不匹配问题,业界已提出许多解决方案。文献[1,2]提出一种基于服务执行轨迹的适配方法,其使用YAWL对服务建模。其中文献[1]从现有服务的执行轨迹出发寻找是否存在轨迹与目标服务的执行轨迹匹配。该方法要求目标服务的功能是现有服务的子功能,否则就无法找到合适的执行轨迹,从而无法适配。文献[2]则把两个服务的执行轨迹合成作为一个新服务的执行轨迹。该方法在适配过程中,如果路径合成失败,则无法分析失败原因。这两种方法都是基于服务执行轨迹的,很难形式化地证明两个服务的交互过程不会产生死锁。文献[3]基于自动机理论提出模式匹配方法。模式匹配方法首先确定与提供服务匹配的所有服务共同遵循的某种模式,然后确定请求服务是否满足这个模式,如果满足就证明匹配成功。该方法对服务行为做了较多简化,在实际的适配过程中难以应用。文献[4]也提出一种模式匹配方法。不同的是,该方法使用开放工作流网[5]对服务行为

到稿日期:2012-01-20 返修日期:2012-03-25
曹国荣(1982—),男,博士生,主要研究方向为形式化方法和Web服务行为适配,E-mail:guorong.cao@gmail.com;谭庆平(1965—),男,教授,主要研究方向为分布式软件工程、形式化方法和Web服务行为适配;解金刚(1985—),男,硕士,主要研究方向为形式化方法和Web服务行为适配;吴 浩(1973—),男,博士生,主要研究方向为分布式软件工程和软件演化。

建模，描述服务的外部接口。目前该方法主要用来解决服务发现和发布问题，但同样可应用于服务适配问题。文献[6]也基于模式匹配思想，但仅限于有限状态服务。文献[7]提出一种基于转换规则的适配器生成方法。该方法在理论上可以应用于任何服务适配，但转换规则及服务间接口映射关系需人工完成，工作量大且易出错。文献[8]提出一种确定并消除服务间接口和协议不匹配的方法。该方法首先为不匹配的服务生成不匹配树，然后根据不匹配树生成适配器描述并最终产生适配器[9,10]。文献[11]提出通过模型检验生成适配器的方法。该方法使用 tableau-based 算法来确定适配器是否存在，如果存在则自动生成满足自定义公平性约束的适配器。尽管模型检验生成适配器的方法较易验证正确性，但状态空间爆炸问题是该方法的主要缺点。

从上述相关工作可以看出，服务适配研究在取得一定成果的同时，主要还存在以下问题：

(1) 服务行为是非循环的，且存在较多的假设限制。现有方法难以针对服务中的循环实施适配，有必要采用新的适配方法适配服务中的循环行为。

(2) 涉及控制流描述和建模较多，涉及数据流很少。控制流和数据流是服务行为适配的同等重要的两个不同方面，现有的服务行为适配方法仅考虑服务交互的控制流，有必要采用新的适配技术对服务的数据流和控制流同时建模和适配，以发挥数据流在服务行为适配中的作用。

(3) 当服务规模较大时，其面对状态空间爆炸问题力不从心。现有方法在分析复杂服务行为(包括分支、循环、并发及它们相互嵌套的时序逻辑和依赖关系)时，无法考虑服务执行过程中出现的全部变量值，有必要采用某种方法以有限的服务行为状态描述表示大量的甚至无穷多的状态。

针对上述问题，本文提出一种基于规范流网的服务行为适配方法。该方法的核心思想包括：(1)基于规范流网(由 Petri 网[12,13]及 YAWL 语言[14,15]扩展而来)对服务行为的控制流和数据流建模，并产生具有良好控制结构、准确表达数据流转的服务行为模型；(2)以符号化可覆盖树的符号化状态表示服务运行过程中大量甚至无穷多的状态(例如循环中状态的表示)；(3)基于数据流构建服务之间数据的输入输出依赖关系，基于符号化可覆盖树生成服务的符号化执行轨迹。在此基础上，利用图的相关知识产生所有符号化执行轨迹对的适配器，并将所有符号化执行轨迹适配器集成为交互服务的最终适配器。该服务行为适配方法为上述问题提供了较好的解决方案。

本文第 2 节概述规范流网的相关概念；第 3 节阐述新的服务行为适配方法，包括基于规范流网的服务行为建模、符号化可覆盖树的定义及相关证明、数据和动作依赖关系及具体的适配算法；第 4 节结合实例说明整个方法的建模和适配过程；最后总结全文。

2 预备知识

本部分首先介绍文中使用的相关符号约定。图 1 是本文用到的图形符号，在基于规范流网的服务行为适配方法中，要用到输入库所、输出库所、控制库所、任务、流关系、各种类型的分支/合并以及数据库所。在符号化可覆盖树中，要用到标识/状态、变迁关系。

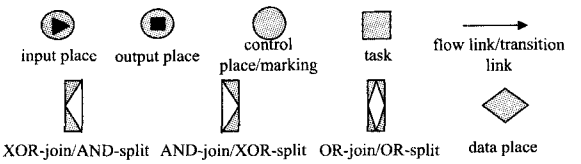

图 1 建模中用到的图形符号

接下来介绍流网、规范流网的概念，其它有关概念及详细阐述可参考文献[16]。

定义 1 流网(FN)是一个七元组 $N=\langle C,T,F,split,join,i,o\rangle$，满足下列条件：

(a) C 表示控制库所的有限集合；

(b) $i,o\in C$ 分别表示输入库所和输出库所；

(c) T 表示任务结点的有限集合；

(d) $F\subseteq((C\backslash\{o\})\times T\cup(T\times(C\backslash\{i\}))$表示流关系；

(e) 图$(C\cup T;F)$的每个结点都在 i 到 o 的有向路径上；

(f) $split:T\rightarrow\{\text{AND,OR,XOR}\}$指明任务的分支行为；

(g) $join:T\rightarrow\{\text{AND,OR,XOR}\}$指明任务的合并行为。

用符号 $C_N,T_N,F_N,split_N,join_N,i_N,o_N$ 来表示流网 N 的各个分量。此外，对于 $\forall u\in C_N\cup T_N,u\bullet=_{def}\{v|\langle u,v\rangle\in F_N\},\bullet u=_{def}\{v|\langle v,u\rangle\in F_N\}$，用 $|A|$ 表示有限集合 A 的势，用 $|p|$ 表示有限路径 p 的长度。

定义 2 规范流网(RFN)可递归定义为：

(a) 一个线性流网、简单分支-合并流网或简单循环流网是一个规范流网；

(b) 如果 $N1$ 和 $N2$ 是规范流网，则 $N1$ 和 $N2$ 的连接，即 $N1\oplus N2$ 是规范流网；

(c) 如果 $N1$ 和 $N2$ 是规范流网，且 $t\in T_{N1}$ 满足 $|\bullet t|=|t\bullet|=1$，则 $N2$ 和 $N1$ 关于 t 的嵌入，即 $N1\Theta_t N2$ 是规范流网；

(d) 其它形式的流网都不是规范流网。

规范流网是一种具有良好控制结构的流网，基于规范流网的服务行为模型的控制流是顺序、分支、循环或它们相互连接嵌套的时序逻辑结构。这种结构使规范流网的语义及使能判定方法比流网的语义及使能判定方法更加高效[17,18]。

3 服务行为适配方法

3.1 基于规范流网的服务行为建模

定义 3 基于规范流网的服务行为模型(SRFN)是一个六元组 $S=\langle N,D,H,type,dom,predicate\rangle$，满足下列条件：

(a) N 是一个规范流网，称为 S 的基网。

(b) D 是数据库所的集合，其中包含两个子集 ID 和 OD，分别代表输入数据库所集合和输出数据库所集合，并且 $ID\cap OD=\emptyset$。

(c) $H\subseteq((D\backslash OD)\times T_N)\cup(T_N\times(D\backslash ID))$是数据流关系，并且满足 $\forall\langle t_1,d\rangle,\langle d,t_2\rangle\in H,t_1$ 在图$(C_N\cup T_N,F_N)$的每条从 i_N 到 t_2 的路径中都会出现。

(d) $type:D\rightarrow TYPE$ 表示每个数据库所的数据类型。$TYPE$ 是所有数据类型的集合。

(e) $dom:TYPE\rightarrow 2^{DOM}$ 表示每个数据类型可以表示的值。DOM 是一个通用个体空间。

(f) $predicate:F_N\rightarrow EXP_{BOOL}$表示 N 中的 OR 和 XOR 任务结点的路由条件。EXP_{BOOL}是所有布尔表达式的集合，这里

的表达式中仅包含集合 D 中出现的变量。也就是说，对于任意的 $e\in EXP_{BOOL}$ 都有 $vars(e)\in D$。在流关系中，$\forall\langle t,c\rangle\in F_N$，如果 $split_N(t)\in\{OR,XOR\}$ 并且 $|t\bullet|>1$，那么 $predicate(t,c)\in EXP_{BOOL}$；否则，$predicate(t,c)$ 就没有定义，用 $predicate(t,c)\uparrow$ 来表示。

用符号 $N_S,D_S,H_S,types_S,dom_S,predicates_S$ 来表示服务行为模型 S 的各个分量，如果 S 的基网是 N，那么就用 C_S，$T_S,F_S,splits_S,joins_S,i_S,o_S$ 代替 $C_N,T_N,F_N,split_N,join_N,i_N,o_N$。

定义 4 服务行为模型 S 的一个标识是一个三元组 $M=\langle M_C,M_D,M_R\rangle$，满足：

(a) $M_C:C_S\to\{0,1\}$，$M_C(c)=1$ 表示在控制库所 c 里有一个令牌，反之则没有令牌。

(b) $M_D:D_S\to DOM_S$，$d\in D_S$，如果 d 中有数据，那么 $M_D(d)\in dom_S(types_S(d))$，用 $M_D(d)\downarrow$ 表示，而且 $M_D(d)$ 就代表当前数据库所 d 中存储的值。相反，用 $M_D(d)\uparrow$ 表示数据库所 d 中没有数据。

(c) $M_R:C_S\times C_S\to\{0,1\}$，$M_R(c_1,c_2)=1$ 表示在 T_S 中存在一个 OR 分支结点 t_α 满足 $\langle t_\alpha,c_1\rangle\in F_S$ 且 $\langle c_2,dual(t_\alpha)\rangle\in F_S$，其中 c_1 和 c_2 都出现在从 t_α 到 $dual(t_\alpha)$ 的同一条路径上。$dual(t_\alpha)$ 表示分支结点 t_α 的对偶结点。

定义 5 服务行为模型 S 的一个符号化标识是一个三元组 $SM=\langle SM_C,SM_D,SM_R\rangle$，满足：

(a) $SM_C\subseteq C_S$；

(b) 映射 $SM_D:D_S\to EXP$，这里 EXP 是所有表达式的集合。$\forall d\in D_S$，如果 $SM_D(d)\downarrow$，那么必定就有 $type(SM_D(d))=type_S(d)$ 且 $vars(SM_D(d))\subseteq D$；

(c) $SM_R\subseteq C_S\times C_S$。

实际上，一个符号化标识代表了一个普通标识的集合。也就是说，$SM=\{\langle M_C,M_D,M_R\rangle\mid\forall c\in C_S:(c\in SM_C\Rightarrow M_C(c)=1),\forall d\in D_S:SM_D(d)\downarrow\Rightarrow SM_D(d)[\theta]=M_D(d),\forall c_1,c_2\in C_S:((c_1,c_2)\in SM_R\Rightarrow M_R(c_1,c_2)=1)\}$。这里 θ 是一个替换，用 $M_D(x)$ 的值来替换在 $SM_D(d)$ 中出现的所有 x，其中 $x\in D_S$。用 $M\in SM$ 表示符号化标识 SM 可以代表普通标识 M。

3.2 符号化可覆盖树

定义 6 服务行为模型 SRFN 的符号化可覆盖树(SCT)是一个四元组 $SCT=\langle SM,F,sm,OM\rangle$，满足：

(a) SM 是 $SRFN$ 的所有可达符号化标识集合；

(b) sm 是初始符号化标识，即符号化可覆盖树的根；

(c) OM 是终止符号化标识集合，$OM\subseteq SM$；

(d) F 是变迁关系，F 具有这样的形式，即 $sm_i[[cond]t>sm_j$，其中 $sm_i,sm_j\in SM,t\in T,cond$ 是一个布尔表达式。

定理 1 SRFN 的符号化可覆盖树是有限的。

证明：假定存在无限的符号化可覆盖树 SCT，其根结点为 sm_0。由于规范流网中的任务结点是有限的，因此 sm_0 只有有限个后继，但 SCT 是无限树，那么至少存在 sm_0 的一个后继是某个无限子树的根，假设这个无限子树的根为 sm_1。那么类似地，sm_1 也必定存在一个后继是某个无限子树的根，将其设为 sm_2。继续构造下去，就找到一条无限的有向路径 $sm_0,sm_1,sm_2,\cdots$。分下面两种情况讨论：

(1)如果 SCT 对应的规范流网 N 中不包含循环，由于任务结点是有限的，因此必定可以经过有限多次变迁到达某个终止标识，这与上述的无限路径矛盾。

(2)N 中包含循环。若令牌处于某个基本分支-合并结构中，则经过有限次传递，令牌会传出此结构，这个传递过程仅产生有限次变迁，不会产生无限路径。基本分支-合并结构可以通过 sj-消除操作成为一个任务结点，对 N 反复进行 sj-消除，得到的 $sj\text{-}collapse^*(N)$ 中仅包含循环结构。不失一般性，对其中的任意一个循环，如图 2 所示，假设循环入口 t_1 和循环出口 t_2 之间只有一个控制库所。用一个四元组表示此循环结构的令牌分布，如 $\langle1,0,0,0\rangle$ 表示 c_1 中含有令牌而其他 3 个控制库所中没有令牌。那么此循环的运行过程就可以表示为 $\langle1,0,0,0\rangle\to\langle0,0,1,0\rangle\to\langle0,1,0,0\rangle\to\langle0,0,1,0\rangle\to\cdots\to\langle0,0,0,1\rangle$。显然，第二次循环时令牌会回到 c_3，也就是上述迁移序列中第二个和第四个元组的令牌分布完全相同。如此重复，第六个也和第二个令牌分布相同，它们对应的标识也仅仅是数据库所中的值不同，根据定义 5 知，它们属于同一个符号化标识。根据算法 CSCT[16]，如果是第二次进入循环，此时的普通标识必定可以由已经存在的某个符号化标识表示，那就不再继续为其产生后继，也就是仅执行一遍循环。所以循环也是用一条有限的有向路径表示，这与上述的无限路径矛盾。

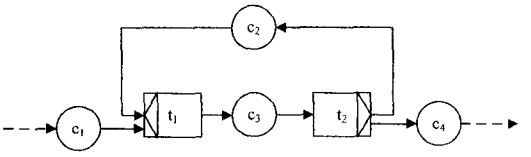

图 2 循环结构示例

3.3 依赖关系

定义 7 给定两个服务的服务行为模型 S1 和 S2。若 $\exists d_1\in ID_{S1}\wedge\exists d_2\in OD_{S2}$ 或者 $\exists d_1\in ID_{S2}\wedge\exists d_2\in OD_{S1}$ 满足 $type(d_2)=type(d_1)\vee dom(type(d_2)\subseteq dom(type(d_1))$，则称 d_1 依赖于 d_2，标记为 $d_1\infty d_2$。

数据依赖关系的集合记为 ∞，对任意数据元素 d，它的依赖项记为 $d\infty$，它的被依赖项记为 ∞d。在实际的服务交互过程中，一个输入或输出动作往往包含多个序列化的数据输入或输出，而且服务交互过程是以输入或输出动作作为基本的信息交互单元。因此，仅仅定义数据依赖关系是不够的。接下来定义输入输出动作间的依赖关系。

在基于规范流网的服务行为模型中，任务结点动作类型可以抽象为：(1)输入消息动作，记为 $?A(x_1,x_2,\cdots,x_m)$；(2)输出消息动作，记为 $!A(y_1,y_2,\cdots,y_n)$。

定义 8 给定数据依赖关系集合 ∞，输入动作 $?A_x(x_1,x_2,\cdots,x_m)$，输出动作 $!A_y(y_1,y_2,\cdots,y_n)$。若 $\exists x_i,y_j,x_i\infty y_j$，则称输入动作 $?A_x(x_1,x_2,\cdots,x_m)$ 依赖于输出动作 $!A_y(y_1,y_2,\cdots,y_n)$，标记为 $?A_x(x_1,x_2,\cdots,x_m)\infty\ !A_y(y_1,y_2,\cdots,y_n)$ 或简记为 $?A_x\infty!A_y$，其中 $1\leqslant i\leqslant m,1\leqslant j\leqslant n$。

根据上述定义，在给定两个服务行为模型后，可以很快计算获得它们的数据依赖关系集。在此基础上，可以计算出每个输入动作的依赖项。

3.4 适配算法

本节提出的服务行为适配算法基于服务行为模型的符号化可覆盖树，利用数据依赖关系和动作依赖关系获取。这一方面避免了适配器产生过程中的处理循环，另一方面也充分

利用了服务行为模型在数据流建模方面的优势。该适配算法如图 3 所示，它的主要步骤包括：(1)以服务行为模型的符号化可覆盖树为输入，获取要适配符号化可覆盖树的每条执行轨迹(代码第 2-3 行及方法 traceGeneration)；(2)以数据依赖关系集、服务双方的输入动作和输出动作集为输入，获取服务双方所有输入输出动作的依赖关系(代码第 4-9 行及方法 actionDependenceGeneration)；(3)以数据依赖关系集和服务双方的符号化执行轨迹为输入，获取每条执行轨迹的最佳可适配执行轨迹(代码第 10 行及方法 getPossibleCoupleTraces)；(4)针对每对要适配的符号化执行轨迹，自动生成它们的轨迹适配器(代码第 11-12 行及方法 traceAdaptation)；(5)最后合成这些轨迹适配器以产生两个服务的最终行为适配器(代码第 13 行及方法 adapterSynthesis)。

在步骤(4)中，首先将动作间的执行时序关系和依赖关系转换为一个以动作为顶点集、时序关系或依赖关系为边的有向图表示。在此基础上，利用图的相关操作生成一个符号化的轨迹适配器。给定图 $G=(V,E)$，对于任意 $u\in V$，$u\cdot\equiv_{def}\{v|\langle u,v\rangle\in E\}$，$\cdot u\equiv_{def}\{v|\langle v,u\rangle\in E\}$，$|u\cdot|$和$|\cdot u|$分别表示顶点 u 的出度和入度。此外，对于给定的动作? A(! A)，! A(? A)称为它的对偶动作，并记为 $dual(?A)(dual(!A))$。

符号化执行轨迹适配器的构建过程是一个循环体(代码第 37-45 行)：寻找有向图中入度为 0 的顶点集，把寻找到的顶点集作为正在构建的符号化执行轨迹适配器的下一个执行单元，同时删除有向图中该顶点集及相关的边。如果寻找不到这样的顶点集并且有向图的顶点集已经为空，则说明符号化执行轨迹适配器构建完成，退出循环并返回符号化执行轨迹适配器；否则说明正在适配的符号化执行轨迹存在无法适配的情形(如死锁)，退出循环并返回空集。

```
//适配器产生方法
1. adapterGeneration(SCT SCT1=(SM1,F1,sm1,OM1), SCT SCT2=
     (SM2,F2,sm2,OM2),data dependence ∝)
2.     traces1=traceGeneration(SCT1);
3.     traces2=traceGeneration(SCT2);
4.     forall transition such as smi[[cond]action>smj∈F1∪F2 do
5.         if action is an input action do
6.             inputActions=inputActions∪{action};
7.         if action is an output action do
8.             outputActions=outputActions∪{action};
9.     actionDependence = actionDependenceGeneration (inputAc-
         tions,outputActions,∝);
10. possibleCoupleTraces=getPossibleCoupleTraces(traces1,traces2,
      actionDependence);
11. forall ⟨trace1,trace2⟩∈possibleCoupleTraces do
12.     traceAdapters = traceAdapters ∪ {traceAdaptation(trace1,
          trace2, actionDependence)};
13. return adapterSynthesis(traceAdapters);
    //符号化执行轨迹产生方法
14. traceGeneration(SCT SCT=(SM,F,sm,OM))
15.     forall om∈OM do
16.         trace={⟨action1,action2⟩,⟨action2,action3⟩,…,⟨ac-
              tionn-1,actionn⟩}，其中 sm[[cond1]action1>
              sm1[[cond2]action2> sm2[[cond3]action3>
              sm3…[[condn-1]actionn-1>smn-1[[condn]ac-
              tionn>om;
17.         traces=traces∪{trace};
18.     return traces;
    //动作依赖关系产生方法
19. actionDependenceGeneration(set inputActions,set outputActions,
      data dependence ∝)
20.     forall inputAction∈inputActions do
21.         forall outputAction∈outputActions do
22.             actionDependence = actionDependence ∪ {⟨in-
                  putAction, outputAction⟩ | ∃x∈vars(in-
                  putAction),y∈vars(outputAction): x∝y};
23.     return actionDependence;
    //可能的要适配符号化执行轨迹对产生方法
24. getPossibleCoupleTraces(set traces1, set traces2, action depen-
      dence actionDependence)
25.     forall trace such as {⟨action1,action2⟩,…,⟨actionn-1,ac-
          tionn⟩}∈traces1 do
26.         actions1={action1,action2,…,actionn-1,actionn};
27.         forall trace such as {⟨action1,action2⟩,…,⟨actionn-1,
              actionn⟩}∈traces2 do
28.             actions2={action1,action2,…,actionn-1,actionn};
29.             if actionDependence(actions1)⊆actions2 ∧ actionDe-
                  pendence(actions2⊆actions1 then
30.                 possibleCoupletraces=possibleCoupletraces∪
                      {⟨trace1,trace2⟩};
31.     return possibleCoupletraces;
    //符号化执行轨迹适配器生成方法
32. traceAdaptation(trace trace1, trace trace2, action dependence ac-
      tionDependence)
33.     vectors=trace1∪trace2∪actionDependence-1;
34.     actions={action1,action2,…,actionn-1,actionn}，其中
          actioni(1≤i≤n)是 trace1∪trace2 中的动作。
35.     以actions 为顶点集，vectors 为边集，构建有向图 TG=(ac-
          tions,vectors);
36.     preActions=Φ
37.     while actions≠Φ do
38.         tempActions={action| action∈actions ∧ |·action|=
              0};
39.         if tempActions==Φ then
40.             return Φ;
41.         if tempActions≠Φ∧preActions≠Φ then
42.             traceAdapter=traceAdapter∪{⟨preActions,dual
                  (tempActions)⟩};
43.             actions=actions\tempActions;
44.             vectors=vectors{⟨actioni,actionj⟩ | actioni∈temp-
                  Actions∧actionj∈action·i};
45.         preActions=dual(tempActions);
46.     return traceAdapter;
    //符号化执行轨迹适配器集成为最终的适配器方法
47. adapterSynthesis(set traceAdapters)
48.     forall traceAdapter∈traceAdapters do
49.         adapter=adapter∪traceAdapter;
50.     return adapter;
```

图 3　服务行为适配算法

4　实例分析

本部分通过一个实例来说明服务行为建模和适配的整个过程。实例中，两个服务协同完成一个网上支付功能，服务 Payment Client(PC)请求支付，服务 Payment Server(PS)保留用户的账户信息，完成 PC 的支付请求。

首先简要介绍两个服务的工作流程。服务 PC 需要提供登录信息，登录成功以后提供需要支付的金额和动态密码，然后等待此次支付的结果。如果支付成功，PC 接收 PS 生成的支付详单，然后显示账户余额；如果支付失败，服务 PC 接收失败的原因。服务 PS 接收 PC 的登录信息，登录成功以后，需要 PC 提供动态密码和支付金额，然后查询用户的账户余额，执行支付过程。如果支付成功，会为 PC 生成一个支付详单，然后显示用户账户的余额；如果支付失败，就返回给用户失败原因。服务 PS 包含循环，可以进行多次支付，一次支付完成以后就回到初始状态，等待用户下次支付的登录操作。两个服务协同存在不匹配的问题，例如 PC 发送支付金额和动态密码与 PS 的接收顺序不一致；如果支付失败，PC 和 PS 的余额操作无法匹配。

利用本文提供的服务行为适配方法，首先对服务 PC 和 PS 建立基于规范流网的服务行为模型。它们的基于规范流网的服务行为模型如图 4 所示。图中菱形代表数据库所。服务 PC 有 3 个输出库所，任务 login 的输出是用户的登录信息，要将登录信息发送给 PS；任务 sum 的输出是支付金额，任务 psw 的输出是用户的动态密码。服务 PC 还有 4 个输入，任务 payment 的输入是 PS 返回的支付成功与否的信息；任务 fail 的输入是 PS 发送的失败原因；任务 success 的输入是 PS 发送的支付详单；任务 balance 的输入是用户账户的余额。服务 PS 的输入输出与 PC 相对应。

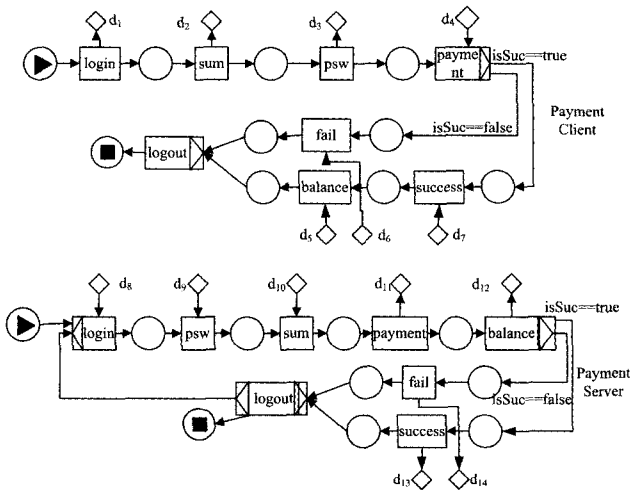

图 4　服务 PC 和 PS 的基于规范流网的服务行为模型

服务 PS 与 PC 的数据依赖关系为 $\propto_d = \{\langle d_4, d_{11}\rangle, \langle d_5, d_{12}\rangle, \langle d_6, d_{14}\rangle, \langle d_7, d_{13}\rangle, \langle d_8, d_1\rangle, \langle d_9, d_3\rangle, \langle d_{10}, d_2\rangle\}$。因此，它们的动作依赖关系为 $\propto_a = \{\langle$? payment,! payment$\rangle, \langle$? balance,! balance$\rangle, \langle$? fail,! fail$\rangle, \langle$? success,! success$\rangle, \langle$? login,! login$\rangle, \langle$? psw,! psw$\rangle, \langle$? sum,! sum$\rangle\}$。

服务 PS 中包含循环，每次循环的令牌传递过程是相同的，但是数据库所每次的值不同，如果为 PS 生成普通可覆盖树，那么普通可覆盖树很有可能是无穷的。因此，为 PS 和 PC 的基于规范流网的服务行为模型生成符号化可覆盖树，如图 5 所示，符号化可覆盖树的结点由符号化标识组成，并且不包含回路，这就用有限的方法表示了服务的无限行为，这是符号化方法的优点。在生成符号化可覆盖树的过程中处理循环，就将循环处理提前到了服务行为建模阶段，从而避免了在适配过程中进行复杂的循环处理。

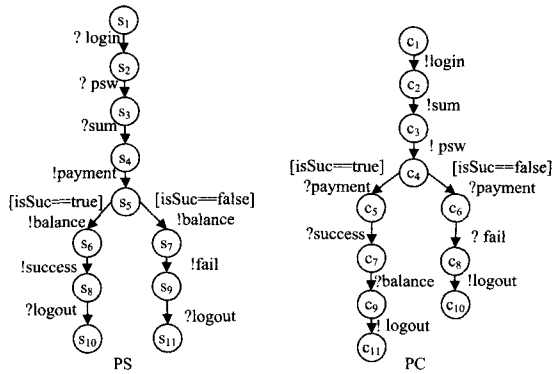

图 5　服务 PC 和 PS 的符号化可覆盖树

接下来，需要得到 PS 和 PC 的符号化执行轨迹。PS 和 PC 都有两条执行轨迹，PS 的两条执行轨迹为：

(1) $trace_{PS1}$ = {⟨? login,? psw⟩,⟨? psw,? sum⟩,⟨? sum,! payment⟩,⟨! payment,! balance⟩,⟨! balance,! success⟩,⟨! success,? logout⟩}

(2) $trace_{PS2}$ = {⟨? login,? psw⟩,⟨? psw,? sum⟩,⟨? sum,! payment⟩,⟨! payment,! balance⟩,⟨! balance,! fail⟩,⟨! fail,? logout⟩}

PC 的两条执行轨迹为：

(1) $trace_{PC1}$ = {⟨! login,! sum⟩,⟨! sum,! psw⟩,⟨! psw,? payment⟩,⟨? payment,? success⟩,⟨? success,? balance⟩,⟨? balance,! logout⟩}

(2) $trace_{PC2}$ = {⟨! login,! sum⟩,⟨! sum,! psw⟩,⟨! psw,? payment⟩,⟨? payment,? fail⟩,⟨? fail,! logout⟩}

如果两个需要适配的服务分别有 m 和 n 条符号化执行轨迹，那么对这些执行轨迹进行两两适配就需要 $m * n$ 次轨迹适配。因此，有必要借助某种计算方法来减少适配的计算量。利用输入输出动作依赖关系可快速获取可能的适配符号化轨迹对。首先获取上述 4 条符号化执行轨迹的对应动作集：

(1) $actions_{PS1}$ = {? login,? psw,? sum,! payment,! balance,! success,? logout}

(2) $actions_{PS2}$ {? login,? psw,? sum,! payment,! balance,! fail,? logout}

(3) $actions_{PC1}$ = {! login,! psw,! sum,? payment,? balance,? success,! logout}

(4) $actions_{PC2}$ = {! login,! psw,! sum,? payment,? fail,! logout}

很明显，对于 $trace_{PS1}$ 和 $trace_{PC1}$：$\propto_a(actions_{PS1}) \subseteq actions_{PC1}$ 并且 $\propto_a(actions_{PC1}) \subseteq actions_{PS1}$；对于 $trace_{PS2}$ 和 $trace_{PC2}$：$\propto_a(actions_{PS2}) \subseteq actions_{PC2}$ 并且 $\propto_a(actions_{PC2}) \subseteq actions_{PS2}$。而对于 $trace_{PS1}$ 和 $trace_{PC2}$：$\propto_a$(? fail) $\notin actions_{PS1}$；对于 $trace_{PS2}$ 和 $trace_{PC1}$：$\propto_a$(? success) $\notin actions_{PS2}$。所以，仅需对执行轨迹对 $trace_{PS1}$ 和 $trace_{PC1}$、$trace_{PS2}$ 和 $trace_{PC2}$ 施行轨迹适配。

本文以符号化执行轨迹对 $trace_{PS1}$ 和 $trace_{PC1}$ 的适配为例，阐述符号化轨迹的适配过程。首先构建以 $actions_{PS1} \cup actions_{PC1}$ 为顶点、以 $trace_{PS1} \cup trace_{PC1} \cup \propto_a^{-1}$ 为边的有向图。图 6(a)是该对符号化执行轨迹的有向图，其中实线表示动作间的时序逻辑关系，虚线表示输入输出动作依赖关系。

在有向图的基础上，按照方法 traceAdaptation 的执行流程有：

第 1 次循环：tempActions＝{! login}，preActions＝Φ，traceAdapter＝Φ；

第 2 次循环：tempActions＝{? login,! sum}，preActions＝{? login}，traceAdapter＝{⟨{? login},{! login,? sum}⟩}；

……

第 12 次循环：tempActions＝{? logout}，preActions＝{? logout}，traceAdapter＝{⟨{? login},{! login,? sum}⟩，⟨{! login,? sum},{? psw}⟩，…，⟨{! balance},{? logout}⟩，⟨{? logout},{! logout}⟩⟩；

第 13 次试图执行循环时，根据判定条件 actions＝Φ 退出循环，方法结束并返回 traceAdapter。图 6(b)即是符号化执行轨迹对 $trace_{PS1}$ 和 $trace_{PC1}$ 的适配器的有向图表示。

同理，根据上述方法，可以获取符号化轨迹对 $trace_{PS2}$ 和 $trace_{PC2}$ 的适配器，其有向图表示见图 6(c)。

最终行为适配器的集成比较简单，仅需要求取各符号化轨迹适配器的并。图 6(d)是最终的行为适配器的符号化可覆盖树的表示。

(a)是以 $actions_{PS1} \cup actions_{PC1}$ 为顶点、以 $trace_{PS1} \cup trace_{PC1} \cup \infty_a^{-1}$ 为边的有向图；(b)是执行轨迹对 $trace_{PS1}$ 和 $trace_{PC1}$ 的轨迹适配器有向图表示；(c)是执行轨迹对 $trace_{PS2}$ 和 $trace_{PC2}$ 的轨迹适配器有向图表示；(d)是服务 PC 和 PS 最终行为适配器的符号化可覆盖树表示。

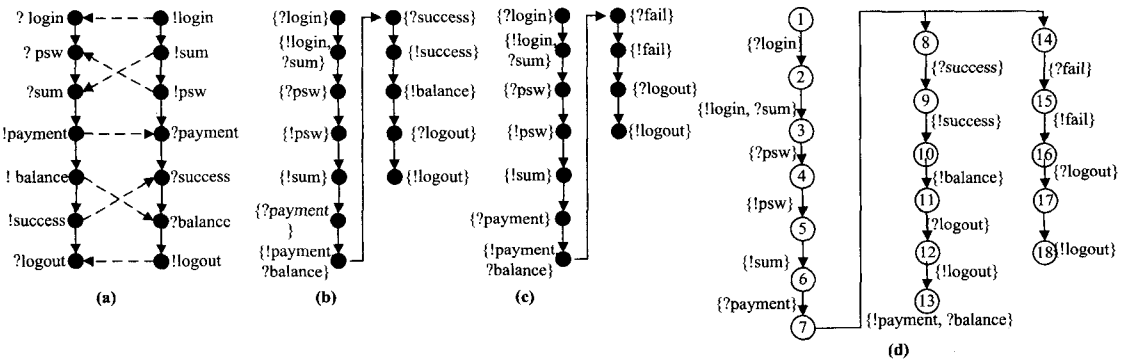

图 6　为 PC 和 PS 生成适配器

结束语　本文基于规范流网对 Web 服务行为建模，针对现有服务行为适配方法在服务循环行为、数据流建模和状态空间爆炸方面存在的问题，提出了一种较好的服务行为适配解决方案。该方案以生成符号化可覆盖树的方式大大减小了状态空间的规模，同时避免了行为适配器产生过程中的处理循环；其又基于符号化可覆盖树的执行轨迹，充分利用自身在数据流建模方面的优势自动生成数据和动作依赖关系，进而最终生成适配器。这为服务行为适配提供了新的思路，从而有助于实际服务行为适配问题的有效解决，为面向服务软件工程开发提供了强有力的技术支持。

以后的主要工作包括 3 个方面：(1)如何将现有不同的服务描述转换为等价的基于规范化流网的服务行为描述，如 BPEL 到服务行为模型的转换；(2)如何将服务行为适配器的符号化可覆盖树描述自动转换成具体的服务描述和可执行代码；(3)通过更多的实例测试来说明方法的有效性，验证解决大规模 Web 服务行为适配的高效性。

参 考 文 献

[1] Brogi A, Popescu R. Service Adaptation through Trace Inspection [C] // Proc of Workshop on Service-Oriented Business Process Integration. 2005: 44-58

[2] Brogi A, Popescu R. Automated Generation of BPEL Adapters [C] // Proc of 4th International Conference on Service Oriented Computing. 2006: 27-39

[3] Massuthe P, Wolf K. An Algorithm for Matching Nondeterministic Services with Operating Guidelines [J]. International Journal of Business Process Integration and Management, 2007, 2 (2): 81-90

[4] Massuthe P, Reisig W, Schmidt K. An Operating Guideline Approach to the SOA [J]. Annals of Mathematics, Computing & Teleinformatics, 2005, 1(3): 35-43

[5] Schmidt K. Controllability of Open Workflow Nets [C] // Proc of Workshop on Enterprise Modelling and Information Systems Architecture. 2005: 236-249

[6] Lohmann N, Massuthe P, Wolf K. Operating Guidelines for Finite-State Services [C] // Proc of 28th International Conference on Application and Theory of Petri nets. 2007: 321-341

[7] Gierds C, Mooij A J, Wolf K. Specifying and Generating Behavioral Service Adapter based on Transformation Rules [R]. Universität Rostock. Germany, 2008

[8] Nezhad H R M, Benatallah B, Martens A, et al. Semi-Automated Adaptation of Service Interactions [C] // Proc of the 16th World Wide Web Conference. ACM Press, 2007: 993-1002

[9] Bracciali A, Brogi A, Canal C. A Formal Approach to Component Adaptation [J]. Journal of Systems and Software, 2005, 74 (1): 45-54

[10] Brogi A, Canal C, Pimentel E. On the Semantics of Software Adaptation [J]. Science of Computer Programming, 2006, 61(2): 136-151

[11] Sinha R, Roop P, Basu S. A Model Checking Approach to Protocol Conversion [J]. Electronic Notes in Theoretical Computer Science, 2008, 203(4): 81-94

[12] Reisig W. Petri Nets: An Introduction [M]. Berlin: Springer, 1985

[13] Murata T. Petri Nets: Properties, Analysis, and Applications [C] // Proc of the IEEE. 1989: 541-580

[14] van der Aalst W M P, ter Hofstede A H M. YAWL: Yet Another

Workflow Language [J]. Inf. Syst., 2005, 30(4): 245-275

[15] van der Aalst W M P, Aldred L, Dumas M, et al. Design and Implementation of the YAWL System [C] // Proc of 16th International Conference Advanced Information Systems Engineering. 2004: 142-159

[16] Xie J G, Tan Q P, Cao G R. Modeling and Analyzing Web Service Behavior with Regular Flow Nets [C] // Proc of International Conference on Web Information Systems and Mining and International Conference on Artificial Intelligence and Computational Intelligence. 2009: 309-319

[17] Wynn M T, Edmond D, van der Aalst W M P. Achieving A General, Formal and Decidable Approach to the OR-join in Workflow using Reset Nets [C] // Proc of 26th International Conference Applications and Theory of Petri Nets. 2005: 423-443

[18] 曹国荣，谭庆平，吴浩，等. 规范流网中 OR-join 任务的语义及使能判定算法[J]. 计算机科学与探索，2010，4(6)：542-551

(上接第 81 页)

分析图 5，并与图 1 进行比较可以看出，被感染主机 I 的数量在时间步为 30 的时候开始迅速增加，并在接近时间步为 60 的时候达到最大值，为 580000 台，感染的情况与图 1 有相似之处。但是该种僵尸程序开始大规模传播的时间比图 1 更早，因为这种僵尸网络在传播时没有根据节点度的多少来进行感染，是比较理想的一种情况，对僵尸网络在无尺度网络下的传播没有很好的描述。

最后对具有网络阻塞特征的僵尸网络传播模型进行仿真，如图 6 所示。

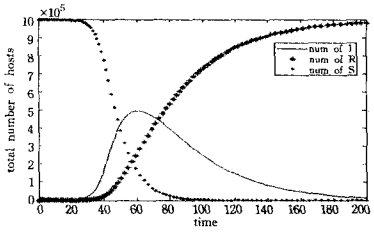

图 6　具有网络阻塞特征的僵尸网络传播模型

把图 6 与图 1 进行比较，由于网络阻塞的情况相似，僵尸程序在时间步接近 50 的时候达到传播的最大数量，并且感染的最大数量为 500000 台，与图 1 相同。但是由于没有考虑主机提前免疫的情况，所有 R 的值比图 1 更大。而且加上没有考虑无尺度网络的集散节点存在，所以在感染主机达到最大值以后，感染主机下降的速度明显没有图 1 快，这是因为图 1 中随着集散节点从感染被恢复到正常状态，I 的下降速度会随着时间步的增加下降得更快。所以图 6 也不能很好地反应出真实网络的僵尸程序传播情况。

综上所述，要对当今网络中的僵尸网络进行确切的描述与分析，就需要考虑无尺度网络的特点以及僵尸程序传播过程中的免疫特征与网络阻塞特征，因此图 1 的传播模型更加符合真实的僵尸网络传播。

结束语　随着科技的发展，僵尸网络作为一个可控的高效攻击平台，得到黑客们的广泛认同和利用[17]，攻击者通过各种手段来增强僵尸网络的隐蔽性和鲁棒性，在传播方面也融合了传统恶意代码的传播方式，这就增加了网络安全人员对其发现、监测以及预防的难度。鉴于当今网络的无尺度性，考虑僵尸网络在无尺度网络下的传播其有重要意义。由于僵尸程序的传播继承自蠕虫病毒，因此了解无尺度网络下的僵尸网络就需要从蠕虫的传播模型着手。传统的蠕虫模型并不能很好地反应出僵尸网络的特征。本文提出的传播模型，结合了无尺度网络的增长性和择优连接性，并考虑了部分主机未被感染前就通过免疫措施将其从脆弱状态转为免疫状态的情况，以及网络流量阻塞的因素。这种模型比传统的僵尸网络传播模型更符合真实网络。

参 考 文 献

[1] Barabasi A L, Albert R. Emergence of Scaling in Random Networks[J]. Science Magazine, 1999, 286(5439): 509-512

[2] Watts D J, Strogatzx S H. Collective Dynamics of "Small-World" Networks[J]. Nature, 1998, 393(6684): 440-442

[3] Liu Li-juan. Research on Dyamic Etworking of Scale-Free Network[R]. TP393. Harbin Institute of Technology, 2007

[4] 洪征，吴礼发，王元元. 三种构建无尺度蠕虫网络的蠕虫传播模型[J]. 吉林大学学报，2008，38(3)：690-694

[5] 张伟. 僵尸网络综述[J]. 软件导刊，2008，7(9)：188-189

[6] http://www.isc.org.cn/20020417/ca290326.htm

[7] Symantec Inc. Symantec Internet security threat report: Trends for July 06～December06. VolumeXI. 2007[OL]. http://eval.symantec.com/mktginfo/enterprise/white_papers/ent-whitepaper_symantec_internet_security_threat_report_x_09_2006.en-us.pdf

[8] Streftaris G, Gibson G J. Statistical Inference for Stochastic Epidemic Models[C] // Proc. of the 17th IWSM. Chania: [s. n.], 2002: 609-616

[9] Frauenthal J C. Mathematical Modeling in Epidemiology[M]. New York: Springer-Verlag, 1980

[10] Zou C C, Gong W, Towsley D. Worm propagation modeling and analysis under dynamic quarantine defense[C] // Proceedings of the ACM CCS Workshop on Rapid Malcode. 2003: 51-6

[11] Dagon D, Zou C C, Lee W. Modeling botnet propagation using time zones[C] // Proc. of the 13th Annual Network and Distributed System Security Symp(NDSS 2006). 2006

[12] 刘浩广，蔡绍洪，张玉强. 无标度网络模型研究进展[J]. 大学物理，2008，27(4)：43-47

[13] 李涛，关治洪，吴正平. 病毒在无标度网络上的传播及控制仿真研究[J]. 计算机应用研究，2007，24(12)：177-182

[14] Wang L, Zhao X, Pei D, et al. Observation andAnalysis of BGP Behavior under Stress[C] // Internet Measurement Workshop. France, November 2002

[15] Kim J, Radhakrishnan S, Dhall S K. Measurement and analysis of worm propagation on Internet network topology[C] // Proc. of the IEEE Int'l Conf. on Computer Communications and Networks(ICCCN2004). 2004: 495-500

[16] 黄彪，谭良，欧阳晨星，等. 无尺度网络下具有免疫特征的僵尸网络传播模型[J]. 计算机应用研究，2012，29(3)：1028-1031

[17] 黄彪，谭良. 无尺度半分布式 P2P 僵尸网络的构建[J]. 计算机工程，2012，38(11)：130-132