# 基于模型的网构软件可达性检测方法研究

赵会群 孙 晶 魏 莹 王文文 郭 峰
(北方工业大学计算机科学与技术系 北京 100144)

**摘 要** 针对网构软件(Internetware)可达性检测中存在状态空间"爆炸"等问题,提出了一种基于网构软件代数模型的可达性检测方法.根据网构软件特性建立其代数模型,通过引入网构相关和网构空间概念,进一步扩展网构软件代数模型.通过明确网构软件可达性与网构组合运算表达式的关系,把可达性判定转化成递归表达式(网构线性相关)判定上来;通过建立网构空间到线性空间映射,把网构线性相关判定问题转化成齐次线性方程组非零解的判定上来.转换过程把线性相关的网构进行压缩,从而有效地抑制了状态空间的增长.给出了可达性检测算法,并讨论了该方法的实际应用.

**关键词** 网构软件;软件可达性;进程代数;线性代数
**中图法分类号** TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01001

# Study on the Method of Checking the Reachability of Internet Ware Based on Model

ZHAO Hui-Qun SUN Jing WEI Ying WANG Wen-Wen GUO Feng
(*Department of Computer Science and Technique, North China University of Technology, Beijing* 100144)

**Abstract** Aim at the challenge of exploded state space where an Internet Ware is searched by using the method of model checking, a new method based on the algebraic model of Internet Ware for checking his reachability is proposed. With respect to the attributes of Internet Ware an algebraic model is proposed first, and then it is fully developed by introducing a series of new concept, for example Internet Ware Correlative and Internet Ware Space. By discovering the relationship between reachability of Internet ware software and Internet ware Operation Expression, converts the reachability of Internet ware software into the recursion of Internet ware Operation Expression. By mapping the Internet Ware Space to Linear Space translates his linear correlativity checking into solving his equations. All above effort can cut the volume of spaces down dramatically with proposed algorithm. Finally, the application aspects of checking method are discussed too.

**Keywords** internetware; reachability; process algebra; linear algebra

## 1 引 言

随着 Internet 技术的不断发展,基于 Internet 的网络应用也在日新月异的变化.电子商务实现了网上购物和贸易往来[1];网格计算实现了全网络计算能力的共享[2];普适计算实现了无处不在的计算资源[3];服务计算提供了灵活的域间业务组合机

收稿日期:2010-11-20;最终修改稿收到日期:2011-05-03.本课题得到国家自然科学基金(61070030,61111130121)、北京市市属高校学术创新团队项目(PHR201107107)资助.**赵会群**,男,1960 年生,博士,教授,研究领域为可信软件.E_mail: zhaohq6625@sina.com.**孙 晶**,女,1968 年生,硕士,现在从事程序设计与软件测试方面的教学与研究工作.**魏 莹**,女,硕士研究生,研究方向为软件测试.**王文文**,女,硕士研究生,研究方向为软件体系结构.**郭 峰**,男,博士,讲师,研究兴趣为模型检测.

制[4]；云计算给用户提供了无需配置的计算和信息服务模式[5]. 网络终端用户已经逐渐地把 Internet 看成是一个大的计算机系统，即“The Network is the Computer”和“Global Ubiquitous Computer”[6]. 这些新的计算模式的出现，促使计算机工作者对软件技术有了新的思考，如何在这种新的“计算机平台”上开发、运行和维护计算机软件，已经成为计算机科学与技术面临的具有挑战性的问题[10]. 为了迎接这一挑战，软件技术也经历了许多变迁. 从面向对象[7]到基于构件[8]的软件开发，再到面向服务[9]的软件集成都体现出这种变化，而这种变化也孕育了一种新的软件技术体系“网构”软件[10].

所谓网构是以开放、自主的方式存在于 Internet 的各个节点之上，可在开放的环境下通过某种方式加以发布，并以各种协同方式与其它软件实体进行跨网络的互连、互通、协作和联盟的软件实体[10]. 网构软件是应能感知外部环境的动态变化，并随着这种变化按照功能指标、性能指标和可靠性指标等进行静态(离线)的调整和动态(在线)的演化，以使系统具有尽可能高的用户满意度的一种新的软件形态. 网构软件具有自主性、协同性、反应性、演化性和多目标性等特征[10].

由于网构是以开放、自主的方式存在于 Internet 的各个节点之上，所以网构软件的可靠性成为用户关注的焦点，而焦点之一就是网构组合的可达性. 可达性(reachability)的原始定义可以在图论中找到，如下：“有向图 $D=(V,A)$，$V$、$A$ 分别是点集和边集；一个可达性关系是 $A$ 上的传递闭包(transitive closure)，满足对所有的有序点对$(s,t)$，存在一个点序列 $v_0=s,v_1,\cdots,v_d=t$，使得$(v_{i-1},v_i)\in A$(其中，$1\leq i\leq d$)”[11]. 计算可达性通常指的是计算能力是否能够被激活，并正常工作. 计算可达性是计算机科学中的经典问题，如无穷状态系统、重写系统、动态系统和混合系统中计算的可达性问题，计算机界也有关于可达性的国际学术会议，如 RP2010. 为了分析计算机系统可达性，可达性检测与分析方法自然成为研究的热点，如基于代数结构的可达性分析[12]、基于 Petri-Nets 的可达性分析[13].

本文提出一种基于代数模型的网构软件可达性检测方法. 该方法通过对被测系统建立代数模型，从代数结构方面分析网构软件的可达性问题. 分析中为了抑制模型状态空间的“爆炸”，把模型的状态空间变换到线性空间，利用线性代数中线性相关理论和方法检测网构软件的可达性. 文中的第 2 节首先建立网构软件的代数模型，在该模型基础上第 3 节讨论网构软件的可达性检测方法，给出具体的检测算法；第 4 节介绍一个模拟检测案例，分别用提出算法和模型检测工具实现两个可达性检测系统，并对两个检测系统进行比较，分析二者的性能；第 5 节通过与相关研究工作比较给出本文结论.

## 2 网构软件代数模型

下面提出一种称为网构代数的代数系统，试图用代数学方法描述网构软件的主要特征，为进一步分析网构软件的性质奠定基础. 模型建立过程中借鉴了进程代数和传统的代数学理论来定义网构、网构组合以及网构软件体系结构等概念，最后给出网构软件代数模型.

### 2.1 网 构

网构是可在开放的环境下通过某种方式加以发布，并以各种协同方式与其它软件实体进行跨网络的互连、互通、协作和联盟的软件实体. 下面分别给出网构、网构组合和网构软件体系结构的形式化描述以及网构同态和同构概念.

**定义 1.** 网构是一个软件实体，它由网构接口、网构实现构成. 网构接口是网构与外部接触点的集合，即$\langle Port_1,Port_2,\cdots,Port_n\rangle$，而每一个接触点 $Port_i$是一个十元组$\langle Dom,ID,Publ_i,Refl_i,Exte_i,Priv_i,Beha_i,Msgs_i,Cons_i,Non\text{-}Func_i\rangle$. 其中：

$Dom$ 是网构所在域，如公司或机构名称等.

$ID$ 是网构的标识，是一个基于标准编码的数据结构，可以用于安全认证、地址解析和信号位等；

$Publ_i$是网构第 $i$ 个接触点能提供给环境或其它网构的功能集合；

$Refl_i$是网构第 $i$ 个接触点能提供给环境或其它网构元数据的集合，该集合有一个布尔型的标志元素，当元数据变化时置该元素为 $T$，当与其交互的环境获得元数据信息后，置 $F$；

$Exte_i$是网构第 $i$ 个接触点运行所需环境或其它网构的功能集合，它具有环境感知能力；

$Priv_i$是网构第 $i$ 个接触点私有属性和功能的集合；

$Beha_i$是网构第 $i$ 个接触点行为语义描述，是一组谓词表达式；

$Msgs_i$是网构第 $i$ 个接触点功能中活动所产生消息的集合；

$Cons_i$是对网构第 $i$ 个接触点行为约束，它通常

包括网构运行的初始条件、前置条件和后置条件，有时为了明确表示这 3 个条件，可把它写成 $Cons(init, pre\text{-}cond, post\text{-}cond)$，其中 $init$、$pre\text{-}cond$ 和 $post\text{-}cond$ 分别表示初始条件、前置条件和后置条件的集合；

$Non\text{-}Func_i$ 是网构第 $i$ 个接触点非功能说明，包括网构的业务策略、合同、安全性、可靠性说明等.

网构实现是网构在计算机上的表现形式，可以是一段代码、也可以是一个软件运行环境，甚至可以是一个文件. 一个网构接口的属性可以表示成 $Att_i(A)$，或者进一步写成 $Att_i(x)$，其中 $Att_i$ 表示某一个属性，如 $Publ_i(x)$，$x$ 表示 $A$ 中的一个活动.

上述的网构属性构成可以随着网构所担当的角色有所不同. 如一个用 Java 实现的跨平台网构通常需要 $\langle Dom, ID, Publ_i, Refl_i, Exte_i, Priv_i, Beha_i, Msgs_i, Cons_i, Non\text{-}Func_i\rangle$ 中的所有描述信息；而对网构的支撑软件（网构中间件），其网构的 $Refl_i$ 和 $Exte_i$ 两个属性可能为空 $\varnothing$；而对一个数据库文件，不需要网构的行为属性，所以有 $\langle Dom, ID, \varnothing, \varnothing, \varnothing, \varnothing, Msgs_i, Cons_i, \varnothing, \varnothing\rangle$.

多个接触点是对网构多目标性的刻画；而属性 $Dom$、$Publ_i$、$Exte_i$、$Priv_i$、$Beha_i$、$Msgs_i$、$Cons_i$、$Non\text{-}Func_i$ 和 $Refl_i$ 分别是对网构自主性和反应性的抽象.

下面给出网构相等和网构演化概念.

**定义 2.**　设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构，$U$ 为网构集合，下同. 如果对所有的接触点，$A$、$B$ 满足下列条件：

(1) $Publ(A)=Publ(B)$；
(2) $Refl(A)=Refl(B)$；
(3) $Extn(A)=Extn(B)$；
(4) $Priv(A)=Priv(B)$；
(5) $Beha(A)\Leftrightarrow Beha(B)$；
(6) $Msgs(A)=Msgs(B)$；
(7) $Cons(A)\Leftrightarrow Cons(B)$；
(8) $Non\text{-}Func(A)\Leftrightarrow Non\text{-}Func(B)$，

则称 $A$、$B$ 相等，记作 $A=B$. ‘$\Leftrightarrow$’为逻辑等价，‘$\Rightarrow$’为永真蕴涵，下同.

**定义 3.**　设 $A$ 和 $B$ 是论域 $Dom(U)$ 中的两个网构，如果对所有的接触点，$A$ 和 $B$ 满足下列条件：

(1) $Dom(A)=Dom(B)$；
(2) $Publ(B)\supseteq Publ(A)$；
(3) $Refl(A)=T$；
(4) $Extn(B)\subseteq Extn(A)$；
(5) $Priv(B)\subseteq Priv(A)$；
(6) $Beha(B)\Rightarrow Beha(A)$；
(7) $Msgs(B)=Msgs(A)$；
(8) $(Cons(B)=Cons(A))$ or $(Cons(B)\Rightarrow Cons(A))$；
(9) $(Non\text{-}Func(B)=Non\text{-}Func(A))$ or $(Non\text{-}Func(B)\Rightarrow Non\text{-}Func(A))$，

则称 $B$ 是 $A$ 的一个演化，记为 $Evolve(B,A)$.

网构演化是对网构的演化性的抽象. 下面进一步讨论网构组合的抽象方法.

### 2.2　网构组合

下面结合网构软件特点，对进程代数中的算子进行扩展，把网构组合解释成网构连接运算的实现.

**定义 4.**　设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构，若 $\exists x\in Extn(A)\wedge\exists y\in Publ(B)$ 使得 $[pre\text{-}cons(x)\wedge pre\text{-}cons(y)]\wedge[Msgs(x)\Rightarrow Msgs(y)]\wedge[Msgs(A)=Msgs(y)\cup Msgs(A)]$，即网构 $A$ 通过发送一个消息“激发”网构 $B$ 中的 $Publ(B)$ 来实现功能需求，并回应执行结果，就称网构 $A$、$B$ 进行了一次“激发”运算，记作 $C=A\oplus B$. 特别地把 $C=A\oplus B$ 记为 $c=x\oplus y$.

$A\oplus B$ 仍然是一个网构，它满足下列性质：

(1) $Dom(C)=Dom(A)\cup Dom(B)$；
(2) $Publ(C)=Publ(A)\cup Publ(B)$；
(3) $Refl(C)=Refl(A)\cup Refl(B)$；
(4) $Extn(C)=Extn(A)\cup Extn(B)$；
(5) $Priv(C)=Priv(A)\cup Priv(B)$；
(6) $Beha(C)\Leftrightarrow Beha(A)\wedge Beha(B)$；
(7) $Msgs(C)=(Msgs(A)\cup Msgs(B))$；
(8) $Cons(C)\Leftrightarrow Cons(A)\wedge Cons(B)$；
(9) $Non\text{-}Func(C)\Leftrightarrow Non\text{-}Func(A)\wedge Non\text{-}Func(B)$.

**定义 5.**　设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构，若 $\exists x\in Extn(A)\wedge\exists y\in Publ(B)$ 使得 $[pre\text{-}cond(x)\wedge pre\text{-}cond(y)]\wedge[Msgs(x)\Rightarrow Msgs(y)]\wedge[Priv(A)=\{y\}\cup Priv(A)]\wedge[Extn(A)=Extn(A)-\{y\}]$，即网构 $A$ 通过拷贝网构 $B$ 中的 $y$ 到 $Priv(A)$ 来实现功能需求，就称网构 $A$、$B$ 进行了一次“使用”运算，记作 $C=A\otimes B$. 特别地把 $C=A\otimes B$ 记为 $c=x\otimes y$.

$A\otimes B$ 仍然是一个网构，它满足下列性质：

(1) $Dom(C)=Dom(A)\cup Dom(B)$；
(2) $Publ(C)\subseteq Publ(A)\cup Publ(B)$；
(3) $Refl(C)=Refl(A)\cup Refl(B)$；

(4) $Extn(C)\subseteq Extn(A)\cup Extn(B)$;

(5) $Priv(C)\subseteq Priv(A)\cup Priv(B)$;

(6) $Beha(C)\Rightarrow Beha(A)\wedge Beha(B)$;

(7) $Msgs(C)=(Msgs(A)\cup Msgs(B))$;

(8) $Cons(C)\Rightarrow Cons(A)\wedge Cons(B)$;

(9) $Non\text{-}Func(C)\Rightarrow Non\text{-}Func(A)\wedge Non\text{-}Func(B)$.

"使用"和"激发"运算是最基本的网构组合运算.在不需要区分二者的情况下,可以把二者统称为"调用"运算,记为 $A \copyright B$.可以证明"调用"运算满足结合率[14].

**定义 6.** 设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构,若 $\exists x\in Publ(A)$, $\exists y\in Extn(B)$ 使得 $[(pre\text{-}cond(y))\Rightarrow((pre\text{-}cond(x))]\wedge[Msgs(y)\Rightarrow Msgs(x)]$,反之 $\exists x\in Extn(A)$, $\exists y\in Publ(B)$ 使得 $[(pre\text{-}cond(x))\Rightarrow((pre\text{-}cond(y))]\wedge[Msgs(x)\Rightarrow Msgs(y)]$,则称 $B$ 与 $A$ 协同运算,记作 $A\Theta B$. $H=pre\text{-}cond(A)\cap pre\text{-}cond(B)$ 称为协同条件集.特别地把 $C=A\Theta B$ 记为 $c=x\Theta y$.

显然,"协同"运算满足交换率.

**定义 7.** 设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构,若 $\exists x\in Publ(A)$, $\exists y\in Extn(B)$ 可以有 $[(pre\text{-}cond(y))\Rightarrow((pre\text{-}cond(x))]$ 成立,反之 $\exists x\in Extn(A)$, $\exists y\in Publ(B)$ 可以有 $[(pre\text{-}cond(x))\Rightarrow(pre\text{-}cond(y))]$ 成立,但 $H=pre\text{-}cond(A)\cap pre\text{-}cond(B)=\varnothing$,则称 $A$ 与 $B$ 并行,记作 $C=A\parallel B$.特别地把 $C=A\parallel B$ 记为 $c=x\parallel y$.

显然,并行运算是协同运算的特例,并都满足交换率.

**定义 8.** 设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构,$\exists x_1\in Publ(A)$、$\exists y_1\in Extn(A)$, $\exists x_2\in Publ(B)$、$\exists y_2\in Extn(B)$,若 $[(post\text{-}cond(y_1)\Rightarrow pre\text{-}cond(x_2))]\Leftrightarrow[post\text{-}cond(y_2)\Rightarrow pre\text{-}cond(x_1)]$,则称 $A$ 与 $B$ 重复,记为 $C=A\cdot B$.特别地,当 $A=B$ 时,称 $A$ 重复执行,记为 $C=x_1\cdot A$,简记 $C=\cdot A$.

**定义 9.** 设 $A$、$B$ 是论域 $Dom(U)$ 中的两个网构,若 $\exists x\in Publ(A)$, $\exists y\in Publ(B)$ 使得 $[pre\text{-}cond(x)]\vee[pre\text{-}cond(y)]$,则称网构 $A$ 与网构 $B$ 选择执行,记为 $C=A+B$.特别地把 $C=A+B$ 记为 $c=x+y$.

定义 6~9 中的运算也有与式(3)类似的性质.这里略.

可以证明"选择"运算满足交换率."重复"、"激发"与"使用"、"协同"与"并行"运算对"选择"运算满足分配率.下面仅就"重复"对"选择"的分配律加以证明,其它证明略.

**定理 1.** "重复"运算对"选择"运算满足分配律,即

$$(A+B)\cdot C=A\cdot C+B\cdot C \tag{1}$$

证明. 要证明式(1)成立,只要证明等式的左边和等式右边满足定义 2 中的 8 个条件即可.为此,我们有代表性地证明第 1 个和第 5 个条件成立,其它证明同理.

首先,对"·"和"+"运算,都有 $Publ(AOP_iB)=Publ(A)\cup Publ(B)$ 性质,所以对 $\forall x\in Publ((A+B)\cdot C)$,有 $x\in Publ(A)\vee x\in Publ(B)\vee x\in Publ(C)$,可得 $x\in Publ(A\cdot C+B\cdot C)$.同理,对 $\forall y\in Publ(A\cdot C+B\cdot C)$ 有 $y\in Publ((A+B)\cdot C)$.因此,有 $Publ((A+B)\cdot C)=Publ(A\cdot C+B\cdot C)$,定义 2 的性质(1)成立.

首先,对"·"运算,有 $Beha(A\cdot B)\Leftrightarrow Beha(A)\wedge Beha(B)$ 性质;而对"+"运算,有 $Beha(A+B)\Leftrightarrow Beha(A)\vee Beha(B)$ 性质.由 $Beha((A+B)\cdot C)\leftrightarrow(Beha(A+B))\wedge Beha(C)\leftrightarrow(Beha(A)\vee Beha(B))\wedge Beha(C)\leftrightarrow(Beha(A)\wedge(Beha(C))\vee(Beha(B)\wedge(Beha(C))\leftrightarrow Beha(A\cdot C+B\cdot C)$,则 $Beha((A+B)\cdot C)\Leftrightarrow Beha(A\cdot C+B\cdot C))$.定义 2 的性质(5)成立. 证毕.

以上借鉴进程代数中的算子概念,对组合进行分类,下面进一步讨论网构组合概念.

**定义 10.** 网构组合是网构运算的实现.它是一个六元组 $\langle ID, Role, Beha, Msgs, Cons, Non\text{-}Func\rangle$.其中:

$ID$ 是组合的标识;

$Role$ 为网构组合与网构的交互点的集合,每个 $Role=\langle Id, Action, Event, LConstrains\rangle$.其中:$Id$ 是 $Role$ 的标识;$Action$ 是 $Role$ 活动的集合,每个活动由事件的连接(谓词)组成;$Event$ 是 $Role$ 产生的事件集合;$LConstrains$ 是 $Role$ 的约束集合.我们把 $Role$ 从组合的其它属性分开来描述的目的是突出组合的多态性,即一个组合可同时实现多个网构的组合.

$Msgs$ 是组合中各 $Role$ 中活动产生事件的集合.

$Beha$ 是组合行为的语义描述.

$Cons$ 是组合约束的集合,它包括组合的初始条件、前置条件和后置条件,有时为了明确表示这 3 个条件可把它写成 $Cons(init, pre\text{-}cond, post\text{-}cond)$,$init$、$pre\text{-}cond$ 和 $post\text{-}cond$ 的含义与网构中的定义

相同.

*Non-Func* 是组合的非功能说明，包括按何种策略、合同、安全性和可靠性组合等.

### 2.3　网构软件体系结构代数模型

下面给出网构软件体系结构的定义.

**定义 11**.　设 $U=\{Dom_1, Dom_2, \cdots, Dom_n\}$ 是多个域的集合，

(1) 网构是一个网构软件体系结构；

(2) 网构组合是一个网构软件体系结构；

(3) 由网构经有限次网构组合(网构运算)后是网构软件体系结构.

网构软件体系结构(ISA)，记为 $ISA=\langle C, O\rangle$. 其中，$C$ 表示组成网构集合，$O$ 表示网构组合(运算)的集合.

由定义 11 可得 ISA 的性质如下：

(1) 封闭性. 即网构与网构，网构与 ISA，ISA 与 ISA 组合后仍是一个 ISA.

(2) 层次性. 即网构可由网构组合而成，而网构又可以再经过组合组成更高层的网构.

(3) 可扩充性. 即一个满足条件的新网构可以通过组合加入到 ISA 中.

从网构组合是网构运算实现角度，可以进一步证明 ISA 对任意一个运算构成代数系统.

**定理 2**.　设 $ISA=\langle C, O\rangle$，则 $ISA$ 对 $O$ 中的每一个组合运算都构成代数系统.

证明.　由网构组合运算的封闭性可得定理 2 的正确性.　　证毕.

为此，把 $ISA=\langle C, O\rangle$ 称为网构代数模型，也称网构软件的代数表达式.

在 ISA 代数模型定义基础上，可以进一步利用代数学方法挖掘网构软件的代数性质，为网构软件研究奠定基础.

**定义 12**.　设 $ISA_1=\langle C_1, O_1\rangle$、$ISA_2=\langle C_2, O_2\rangle$，若 $\forall x\in C_1$ 总有 $\exists y\in C_2$，使得 $y$ 与 $x$ 对应，则称 $ISA_1$ 与 $ISA_2$ 之间存在着一个映射，记为 $f: ISA_1\rightarrow ISA_2$，或 $y=f(x)$. 若映射是满射，则称网构间映射为满射；若映射是一对一的，则称网构间为一一映射.

**定义 13**.　设 $ISA_1=\langle C_1, O_1\rangle$、$ISA_2=\langle C_2, O_2\rangle$ 是两个网构，$f$ 是 $ISA_1$ 到 $ISA_2$ 的一个映射，若对 $\forall x\in C_1$ 和 $\forall y\in C_1$ 有 $f(xOP_i y)=f(x)OP_j f(y)$，其中 $OP_i\in O_1$，$OP_j\in O_2$，则称 $f$ 为从 $S_1$ 到 $S_2$ 的同态，也称 $ISA_1$ 与 $ISA_2$ 同态；若 $f$ 是单射，则称 $f$ 是从 $ISA_1$ 到 $ISA_2$ 的单一同态；若 $f$ 是一一映射，则称 $f$ 是从 $ISA_1$ 到 $ISA_2$ 的同构，也称 $ISA_1$ 与 $ISA_2$ 同构.

用同态与同构概念可以进一步描述基于网构软件的进化性、网构之间的关系等概念，这里仅给出后者.

**定义 14**.　给定 $ISA_1=\langle C_1, O_1\rangle$、$ISA_2=\langle C_2, O_2\rangle$，构造一个新的网构 $ISA_1\times ISA_2=\langle C_1\times C_2, OP_k\rangle$. 其中 $C_1\times C_2$ 是网构集合的笛卡尔乘积，而 $OP_k$ 定义成对 $\forall x_1, x_2\in C_1$ 和 $\forall y_1, y_2\in C_2$ 有 $\langle x_1, y_1\rangle OP_k\langle x_2, y_2\rangle=\langle x_1 OP_i x_2, y_1 OP_j y_2\rangle$，$OP_i\in O_1$、$OP_j\in O_2$. 称 $ISA_1\times ISA_2$ 是 $ISA_1$ 到 $ISA_2$ 的积结构，而 $ISA_1$ 和 $ISA_2$ 是 $ISA_1\times ISA_2$ 的因子，这里的 $OP_i$、$OP_j$ 和 $OP_k$ 运算是任意网构组合运算.

在定义 14 的基础上可以定义网构间的关系.

**定义 15**.　对 $\forall x_1、x_2\in C_1$ 和 $\forall y_1、y_2\in C_2$，在定义 14 中，所有满足 $\langle x_1 OP_i x_2\rangle$ 的 $(x_1, x_2)$ 的任意一个子集合称为运算 $OP_i$ 的一个关系；所有满足 $\langle y_1 OP_j y_2\rangle$ 的 $(y_1, y_2)$ 的任意一个子集合称为运算 $OP_j$ 的一个关系.

### 2.4　网构软件可达性概念

**定理 3**.　任意一个 $ISA=\langle C, O\rangle$ 的代数表达式都可以写成下面的标准型.

$$ISA=a_1 Op^{\wedge} C_1+a_2 Op^{\wedge} C_2+\cdots+a_m Op^{\wedge} C_m \quad (2)$$

或简写成

$$ISA=a_1 C_1+a_2 C_2+\cdots+a_m C_m \quad (3)$$

其中，$a_i\in C_j$；$C_j\in C$，$m$ 为所有网构的个数，$i, j$ 可能不等，$Op^{\wedge}$ 是“使用”或“激发”运算.

证明.　采用归纳法. 当 $k=1$ 时，用户通过发送命令启动系统，所以 $ISA=a_1 Op^{\wedge} C_1$，定理成立. 现假设 $k=n-1$ 成立，即 $ISA=a_1 Op^{\wedge} C_1+a_2 Op^{\wedge} C_2+\cdots+a_{n-1} Op^{\wedge} C_{n-1}$，去证明 $k=n$ 有 $ISA=a_1 Op^{\wedge} C_1+a_2 Op^{\wedge} C_2+\cdots+a_n Op^{\wedge} C_n$. 下面就 ISA 中的各种运算证明式(2)的正确性.

(1) 对选择运算“+”. $ISA+a_n Op^{\wedge} C_n=a_1 Op^{\wedge} C_1+a_2 Op^{\wedge} C_2+\cdots+a_{n-1} Op^{\wedge} C_{n-1}+a_n Op^{\wedge} C_n$，根据定义 11，$ISA_{n-1}+a_n Op^{\wedge} C_n$ 仍然是一个网构软件体系结构. 因此，$ISA=a_1 Op^{\wedge} C_1+a_2 Op^{\wedge} C_2+\cdots+a_n Op^{\wedge} C_n$，定理成立.

(2) 对重复运算“·”. 设 $a_n\in C_j$，$C_n\in C$，构造 $a_n Op^{\wedge} C_n\cdot ISA_{n-1}=a_n Op^{\wedge} C_n\cdot(a_1 Op^{\wedge} C_1+\cdots+a_{n-1} Op^{\wedge} C_{n-1})$，由于“·”对“+”满足分配律，则 $a_n Op^{\wedge} C_n\cdot(a_1 Op^{\wedge} C_1+\cdots+a_{n-1} Op^{\wedge} C_{n-1})=a_n Op^{\wedge} C_n\cdot a_1 Op^{\wedge} C_1+\cdots+a_n Op^{\wedge} C_n\cdot a_{n-1} Op^{\wedge} C_{n-1}$，选择 $x_i\in Publ(C_n)$ 展开上式有 $a_n Op^{\wedge} C_n\cdot$

$a_1 Op^\wedge C_1 + \cdots + a_n Op^\wedge C_n \cdot a_{n-1} Op^\wedge C_{n-1} = a_n Op^\wedge x_1 \cdot a_1 Op^\wedge C_1 + \cdots + a_n Op^\wedge x_{n-1} \cdot a_{n-1} Op^\wedge C_{n-1} + C_0$，其中 $C_0$ 是一个空网构. 构造 $b_i = a_n Op^\wedge x_i \cdot a_i$，$i \in [1; n-1]$ 并且 $b_n Op^\wedge C_n = C_0$，则 $a_n Op^\wedge C_n \cdot ISA_{n-1} = b_1 Op^\wedge C_1 + \cdots + b_n Op^\wedge C_n = S_{n-1} + b_n Op^\wedge C_n$，又因为 $a_n Op^\wedge C_n \cdot ISA_{n-1}$ 仍然是一个网构软件体系结构，因此，有 $ISA = b_1 Op^\wedge C_1 + b_2 Op^\wedge C_2 + \cdots + b_n Op^\wedge C_n$.

(3) 对协同运算“Θ”. 设 $a_n \in C_j$，$C_n \in C$，构造 $a_n Op^\wedge C_n \Theta ISA_{n-1} = a_n Op^\wedge C_n \Theta (a_1 Op^\wedge C_1 + \cdots + a_{n-1} Op^\wedge C_{n-1})$，由于“Θ”对“+”满足分配律，则 $a_n Op^\wedge C_n \Theta (a_1 Op^\wedge C_1 + \cdots + a_{n-1} Op^\wedge C_{n-1}) = a_n Op^\wedge C_n \Theta a_1 Op^\wedge C_1 + \cdots + a_n Op^\wedge C_n \Theta a_{n-1} Op^\wedge C_{n-1}$，选择 $x_i \in Publ(C_n)$ 展开上式有 $a_n Op^\wedge C_n \Theta a_1 Op^\wedge C_1 + \cdots + a_n Op^\wedge C_n \Theta a_{n-1} Op^\wedge C_{n-} = a_n Op^\wedge x_1 \Theta a_1 Op^\wedge C_1 + \cdots + a_n Op^\wedge x_{n-1} \Theta a_{n-1} Op^\wedge C_{n-1} + C_0$，其中 $C_0$ 是一个空网构. 构造 $b_i = a_n Op^\wedge x_i \Theta a_i$，$i \in [1; n-1]$，并且 $b_n Op^\wedge C_n = C_0$，则 $a_n Op^\wedge C_n \Theta ISA_{n-1} = b_1 Op^\wedge C_1 + \cdots + b_n Op^\wedge C_n = S_{n-1} + b_n Op^\wedge C_n$，又因为 $a_n Op^\wedge C_n \Theta ISA_{n-1}$ 仍然是一个网构软件体系结构. 因此，有 $ISA = b_1 Op^\wedge C_1 + b_2 Op^\wedge C_2 + \cdots + b_n Op^\wedge C_n$.

(4) 同样的方法可以证明对使用和激发运算“$Op^\wedge$”定理成立.

上述过程证明了对所有的网构运算 ISA 都有 $ISA = a_1 Op^\wedge C_1 + a_2 Op^\wedge C_2 + \cdots + a_m Op^\wedge C_m$.

在(1)、(2)的证明中，通过构造新的活动实现了 ISA 的重新配置，这种构造方法是从左侧进行，其物理意义表现为更新应该主动适应 ISA 的风格.

标准型式(3)中的 $a_i$ 通常是一个行为明确的活动，如安全性确认等可信活动等，所以可以把 $a_i$ 看成是常量；

为了分析网构软件系统的可达性，下面给出两个相关概念——死锁与活锁.

**定义 16.** 如果一个网构软件无限循环执行某几个网构功能，则称系统出现“活琐”.

“活锁”表现为网构代数表达式中包括递归.

**定理 4.** “活锁”表现为网构代数表达式是一个递归表达式.

证明. 定理显然成立.

**定义 17.** 对一个网构软件系统，如果网构总能到达，那么称该网构软件系统有活动性；如果一个网构调用在执行某个网构组合后异常终止，则称该网构软件系统出现“死锁”，记为“⊥”.

“终止”活动“⊥”是一种正常的网构活动，但“⊥”出现在需要继续执行的调用过程中时，就会引起“死锁”. 根据定义 17，显然有定理 5 成立.

**定理 5.** 在有“终止”活动“⊥”的网构软件系统中，其网构表达式中某项的常量只有“⊥”.

证明. 反证法. 假设网构表达式 $ISA = a_1 C_1 + a_2 C_2 + \cdots + a_k C_k$，其中 $a_i$ 为“⊥”，记有 $ISA = a_1 C_1 + a_2 C_2 + \cdots . \perp C_i + \cdots + a_k C_k$. 根据“⊥”的定义，一定不存在 $x \in C_i$ 与“⊥”发生调用运算，所以 $\perp C_i = \perp$，定理成立. 证毕.

**例 1.** $F = aG + bF$；$G = cG + dG$. $G$ 无限自循环，系统出现活琐.

$X = aY + bX$；$Y = cY + dX + \perp$. 如果网构 $Y$ 执行常量“⊥”，那么系统进入终止状态.

**定义 18.** 如果一个网构代数表达式中没有出现“活锁”和“死锁”，则称该网构软件系统具有可达性.

## 3 网构软件可达性检测算法研究

本节讨论网构软件可达性检测算法，方法是通过定义网构的线性相关性，把网构软件可达性判断等价变换到网构的线性相关性判定上来，利用线性代数理论和方法求解网构软件的可达性.

**定义 19.** 设 $F_1, F_2, \cdots, F_n$ 是 $N$ 个可访问的网构，$a_1, a_2, \cdots, a_n$ 分别是 $F_1, F_2, \cdots, F_n$ 网构中的活动. 如果对任意的 $i \in [1, N]$ 都有 $F_i = a_1 F_1 + a_2 F_2 + \cdots + a_{i-1} F_{i-1} + a_i F_i + a_{i+1} F_{i+1} + \cdots + a_n F_n$，则称 $F_1, F_2, \cdots, F_n$ 线性相关.

网构线性相关与线性空间中向量的线性相关性具有相似的含义.

**定理 6.** 设 $S_1 = a_1 F_1 + a_2 F_2 + \cdots + a_n F_n$、$S_2 = b_1 G_1 + b_2 G_2 + \cdots + b_k G_k$ 是两个网构，$a_i$ 和 $b_i$ 分别是网构 $S_1$ 和 $S_2$ 中的活动，下同；$S_1 \Theta S_2 = c_1 H_1 + c_2 H_2 + \cdots + c_n H_n$ 是网构协同组合表达式，则 $S_1 \Theta S_2$ 是递归表达式的充要条件是 $H_1, H_2, \cdots, H_n$ 线性相关.

证明. 充分性，用归纳法证明. 当 $n=1$ 时，即 $H_1 = d_1 H_1$，则 $S_1 \Theta S_2 = c_1 H_1 = c_1 d_1 H_1$，显然是递归表达式；当 $n=2$ 时，即 $H_1 = d_{11} H_1 + d_{12} H_2$，$H_2 = d_{21} H_1 + d_{22} H_2$，分别代入 $S_1 \Theta S_2 = c_1 H_1 + c_2 H_2 + \cdots + c_n H_n$ 中，则 $S_1 \Theta S_2 = c_1 d_{11} H_1 + c_1 d_{12} H_2 + c_2 d_{21} H_1 + c_2 d_{22} H_2$，根据“$Op^\wedge$”运算对“+”运算的分配律整理后有 $S_1 \Theta S_2 = m_1 H_1 + m_2 H_2$，其中 $m_1 = c_1 d_{11} + c_2 d_{21}$，$m_2 = c_1 d_{12} + c_2 d_{22}$，显然也是递归表达式. 同理，可以证明在假设 $n=k$ 时 $S_1 \Theta S_2$ 是递归表达式，

$n=k+1$ 时 $S_1\Theta S_2$ 也是递归表达式.

必要性. 由于 $S_1\Theta S_2=c_1H_1+c_2H_2+\cdots+c_nH_n$ 是递归表达式，则一定 $\exists i\in[1,N]$ 使得 $H_i=h_iH_i$；或者 $\exists i_1,i_2,\cdots,i_k\in[1,N]$，使得 $H_{i1}=h_{i2}H_{i2}$，$H_{i2}=h_{i3}H_{i3}$，$\cdots$，$H_{ik}=h_{i1}H_{i1}$；或者 $\exists i_1,i_2,\cdots,i_k\in[1,N]$，使得：$H_{i1}=h_{i1}H_{i1}+h_{i2}H_{i2}+h_{i3}H_{i3}+\cdots+h_{ik}H_{ik}$，$H_{21}=h_{i1}H_{i1}+h_{i2}H_{i2}+h_{i3}H_{i3}+\cdots+h_{ik}H_{ik}$，$H_{ik}=h_{i1}H_{i1}+h_{i2}H_{i2}+h_{i3}H_{i3}+\cdots+h_{ik}H_{ik}$. 这 3 种情形分别代表直接递归和间接递归，所以可以得出 $H_1,H_2,\cdots,H_n$ 线性相关.

**定义 20.** 设 $ISA=\langle C,O\rangle$，“+”和“$Op^{\wedge}$”分别是网构的选择运算和调用运算，如果对“+”满足：$F_1+F_2=F_2+F_1$（交换率）；$(F_1+F_2)+F_3=F_2+(F_1+F_3)$（结合律）以及存在零网构和异常网构，即 $F_1+\sharp=F_1$（零网构 $\sharp$）；$F_1+(\sim F_1)=\sharp$（异常网构-$F$）；“$Op^{\wedge}$”对“+”满足：$H(h_1+h_2)=Hh_1+Hh_2$（左分配律），$(h_1+h_2)H=h_1H+h_2H$（右分配律），并且有内部协同网构活动，即 $\varepsilon F=F$（内部协同网构活动对 $\varepsilon$），则称 $ISA=\langle C,O\rangle$ 对“+”和“$Op^{\wedge}$”构成网构空间.

**定理 7.** $ISA=\langle C,O\rangle$ 对“+”和“$Op^{\wedge}$”构成网构空间.

证明. 设 $F_1,F_2,F_3$ 分别是 ISA 中的网构. 根据选择“+”运算性质有

$F_1+F_2=F_2+F_1$（交换率）；

$(F_1+F_2)+F_3=F_2+(F_1+F_3)$（结合律）.

又由于 ISA 软件中存在空网构和异常网构，分别记为 $\sharp$ 和 $\sim F$，则有

$F_1+\sharp=F_1$（零网构 $\sharp$）；

$F_1+(\sim F_1)=\sharp$（异常网构 $\sim F$）.

又由调用运算“$Op^{\wedge}$”性质，则有

$\varepsilon F=F$（内部协同活动对 $\varepsilon$，如一对应答网构）；

$H(h_1+h_2)=Hh_1+Hh_2$（左分配律）；

$(h_1+h_2)H=h_1H+h_2H$（右分配律）.

因此，$ISA=\langle C,O\rangle$ 对“+”和“$Op^{\wedge}$”构成网构空间. 证毕.

**定理 8.** 设 $ISA=\langle C,O\rangle$，一定存在一个线性空间 $V$ 与 ISA 同构.

证明. 设 $P$ 为一个数域，$H$ 为网构中活动集合. 按照以下规则构造映射 $\Phi$：对 $\forall h_i\in H$，令 $\Phi(h_i)=p_i$，$p_i\in P$ 是活动 $h_i$ 执行所需要的代价，特别地，令 $p_i$ 为活动使能的真假值；对 $\forall C_i\in C$，令 $\Phi(C_i)=\{\Phi(h_1),\Phi(h_2)\cdots\Phi(C_i)\}=\boldsymbol{X}_i$. 其中 $\boldsymbol{X}_i$ 为 $n$ 维向量，$\boldsymbol{X}_i=(id,x_1,x_2,\cdots,x_{n-1})$，$x_i=\Phi(h_i)$ 为整数，$id$ 是 $C_i$ 的标示；“+”与“$+_{整数}$”对应，“$Op^{\wedge}$”与“$\times_{整数}$”对应，则向量 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$ 以及“$+_{整数}$”和“$\times_{整数}$”构成线性空间.

对上述定义的映射 $\Phi$，由于 $id$ 的引用，显然 $\Phi$ 是一一映射.

又对 $\forall A,B\in C$，$A=\{a_1,a_2,\cdots,a_n\}$，$B=\{b_1,b_2,\cdots,b_n\}$，$a_i$ 是 $A$ 中的活动，$b_i$ 是 $B$ 中的活动，有 $A+B=\{a_1+b_1,a_2+b_2,\cdots,a_n+b_n\}$，则

$$\begin{aligned}\Phi(A+B)&=\{\Phi(a_1+b_1),\Phi(a_2+b_2),\cdots,\Phi(a_n+b_n)\}\\&=\{\Phi(a_1)+_{整数}\Phi(b_1),\Phi(a_2)+_{整数}\Phi(b_2),\cdots,\\&\quad\Phi(a_n)+_{整数}\Phi(b_n)\}\\&=\{id_1,x_1,x_2,\cdots,x_{n-1}\}+_{整数}\{id_2,y_1,y_2,\cdots,y_{n-1}\}\\&=\Phi(A)+_{整数}\Phi(B)=\boldsymbol{X}_1+_{整数}\boldsymbol{X}_2.\end{aligned}$$

又对 $\Phi(h_iOp^{\wedge}C_i)=\Phi(h_1Op^{\wedge}c_1,h_1Op^{\wedge}c_2,h_1Op^{\wedge}c_3,\cdots,h_1Op^{\wedge}c_n)=\{\Phi(h_1Op^{\wedge}c_1),\Phi(h_1Op^{\wedge}c_2),\Phi(h_1Op^{\wedge}c_3),\cdots,\Phi(h_1Op^{\wedge}c_n)\}=\{\Phi(h_1)\times_{整数}\Phi(c_1),\Phi(h_1)\times_{整数}\Phi(c_2),\Phi(h_1)\times_{整数}\Phi(c_3),\cdots,\Phi(h_1)\times_{整数}\Phi(c_n)\}=\Phi(h_i)\times_{整数}\{\Phi(c_1),\Phi(c_2),\Phi(c_3),\cdots,\Phi(c_n)\}=p_i\times_{整数}\Phi(C_i)=p_i\times_{整数}\boldsymbol{X}_i$，这里 $c_i\in C_i$，

又由于 $ISA=a_1Op^{\wedge}C_1+a_2Op^{\wedge}C_2+\cdots+a_mOp^{\wedge}C_m$，所以有

$$\begin{aligned}&\Phi(a_1Op^{\wedge}C_1+a_2Op^{\wedge}C_2+\cdots+a_mOp^{\wedge}C_m)=\\&\quad\Phi(a_1)\times_{整数}\Phi(C_1)+_{整数}\Phi(a_2)\times_{整数}\Phi(C_2)+\cdots+\\&\quad\Phi(a_m)\times_{整数}\Phi(C_m)\\&=p_1\times_{整数}\boldsymbol{X}_1+_{整数}p_2\times_{整数}\boldsymbol{X}_2+\cdots+p_m\times_{整数}\boldsymbol{X}_m,\end{aligned}$$

即任何一个网构表达式都可以找到唯一的线性表达式与其对应，反之亦然. 因此，定理成立.

在上面的推导过程中 $\Phi(h_1Op^{\wedge}c_i)=\Phi(h_1)\times_{整数}\Phi(c_i)$ 是根据马尔可夫链特性得出的结论，可以证明“使用运算”可以转化成“激发”运算，而“激发”运算下活动的激发序列满足半马尔可夫特性[22]. 因此，活动序列的非功能属性为各个活动属性值的乘积，如活动的可靠性等[14].

**推理 1.** 设 $F_1,F_2,\cdots,F_n$ 是 $N$ 个可访问的网构，$F_1,F_2,\cdots,F_n$ 线性相关的充分必要条件是在同构变换下与 $F_1,F_2,\cdots,F_n$ 对应的 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n$ 系数行列式的值为 0.

推理 1 说明可以通过网构的线性相关性判断网构的可达性.

根据以上推断，下面给出网构软件可达性检测步骤和算法.

求解的步骤如下：

1. 根据网构注册表，确定网构代数表达式的系数矩阵；

2. 通过计算系数矩阵的行列式，判断网构相关性，从而检测网构软件可达性.

具体求解算法如下.

**算法 1.** 基于 ISA 代数模型的软件可达性检测算法.

1. 根据网构注册表,确定网构代数表达式的系数矩阵;
输入:注册表的信息表
输出:相关服务元素集
for 一个注册表中的每个服务流($f_1,f_2,f_3,\cdots,f_n$)
for $f_i$
if $f_j \notin$ forward_set_rel[$f_i$]
/*建立网构关系对象集合*/
{forward_set_rel[$f_i$]=forward_set_rel[$f_i$]$\cup$
{$f_j$}
coefficient[$i,j$]=1
/*向量元素分量置 1,表示可以*/}

2. 扫描系数矩阵,判断网构的相关性.
输入:相关服务元素的系数矩阵
输出:活动路径中服务元素列表
for each $i$
for each $j>i$
{*result*=coefficient_matrix($i,j$) and
coefficient_matrix($j,i$)/*扫描矩阵系数*/
if *result*=1 then print($f_i,f_{i+1},f_{i+2},\cdots,f_j$)
/*输出相关网构流*/
return}

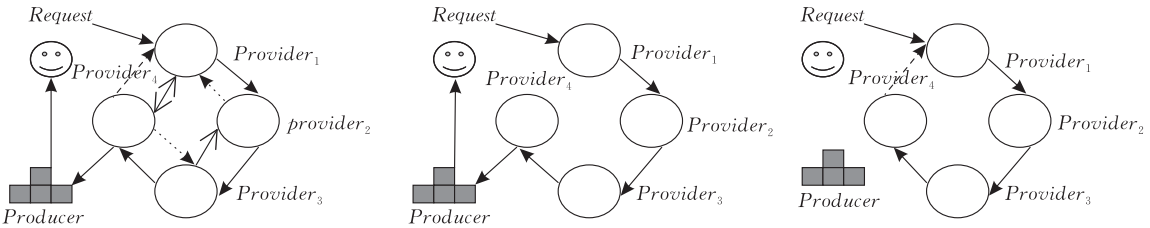
在算法 1 中步 1 的执行时间为 $O(H*L)$,其中,$H$ 是网构注册表的长度,$L$ 是最长网构流的长度;步 2 的执行时间是 $O(N^2)$,$N$ 为关系集合的元素个数.

## 4 案例分析

下面结合一个零售企业订货业务,介绍两种网构可达性检测方法.

### 4.1 基于 ISA 代数模型的可达性检测

图 1 是一个网构组合案例. 其中,*Request*、*Producer*、$Provider_1\sim Provider_4$ 分别为通过 UDDI[15] 发布的网构,每一个网构又都有其内部功能,从而形成嵌套的业务逻辑.




(a) 网构服务组合案例 (b) 正常网构服务流 (c) 非可达的网构服务流

图 1

该网构组合描述了顾客、多个销售代理和生产厂家为顾客提供订货计划的服务流. 正常的服务流如图 1(b)所示,$Provider_1$ 根据 $Provider_2$ 提供的最佳折扣货物信息,把订货业务转包给 $Provider_2$;以此类推,最终 $Provider_4$ 通过 *Producer* 给 *Request* 提供订货服务. 最佳折扣服务信息流如图 1(a)所示,$Provider_1$ 与 $Provider_4$ 相互通报了最佳折扣服务信息,$Provider_4$ 在向 $Provider_1$ 通报的同时,也向 $Provider_3$ 通报,而 $Provider_3$ 又通报给 $Provider_2$,$Provider_2$ 又通报给 $Provider_1$,上述服务信息流用箭头表示. 正因为 $Provider_1$ 与 $Provider_4$ 相互通报了同一批货物信息,所以就会出现图 1(c)所示的非可达的服务流. 这种情况发生在 $Provider_1$ 没有及时发布与 *Producer* 失去最佳折扣关系服务信息,使得 *Request* 订单到来时,出现循环不可达的服务流.

应用提出的 ISA 代数模型和算法 1 的检测过程如下:

根据图 1 表述的服务组合,其 ISA 代数模型为

$$P=(a_{11}Provider_1+a_{22}Provider_2+a_{33}Provider_3+a_{44}Provider_4)\oplus f_1F \quad (4)$$

即 *Request* 可以连接到 $Provider_{1\sim4}$ 中的一个获得订货,$a_{ii}$ 是订货活动. 又由于 $Provider_1$ 与 $Provider_4$ 之间可以相互提供订货,即有

$$Provider_1=a_{14}Provider_4,\ Provider_4=a_{41}Provider_1 \quad (5)$$

把 $Provider_1$ 和 $Provider_4$ 分别代入 $P$ 的代数表达式有

$$P=(a_{11}a_{14}Provider_4+a_{22}Provider_2+a_{33}Provider_3+a_{44}a_{41}Provider_1)\oplus f_1F \quad (6)$$

显然,式(6)是递归表达式.

根据推论 1 及算法 1 得,齐次线性方程组(7)有非零解. 对其任何两个 2 阶系数行列式就解时,只有式(8)的式子为 0.

这里 $a_{ij}$ 分别表示与网构 $Provider_1$ 和 $Provider_2$ 相对应的线性空间中向量 $\boldsymbol{Y}_1$ 和 $\boldsymbol{Y}_2$ 的分量. 根据推论 1 证明中映射建立的方法,当 $Provider_1$ 通过 $a_1$($a_1$ 是 $Provider_1$ 的一个活动)向外界提供服务时,

$a_{11}$为真 1，当 $Provider_1$ 向 $Provider_2$ 提供 $a_1$ 服务时，$a_{12}$ 为真 1；同样地，当 $Provider_2$ 通过 $a_2$（$a_2$ 是 $Provider_2$ 的一个活动）向外界提供服务时，$a_{22}$ 为真 1，当 $Provider_2$ 向 $Provider_1$ 提供 $a_2$ 服务时，$a_{21}$ 为真 1.

$$\begin{cases} a_{11}\boldsymbol{X}_1+a_{12}\boldsymbol{X}_2+a_{13}\boldsymbol{X}_3+a_{14}\boldsymbol{X}_4=0 \\ a_{21}\boldsymbol{X}_1+a_{22}\boldsymbol{X}_2+a_{23}\boldsymbol{X}_3+a_{24}\boldsymbol{X}_4=0 \\ a_{31}\boldsymbol{X}_1+a_{32}\boldsymbol{X}_2+a_{33}\boldsymbol{X}_3+a_{34}\boldsymbol{X}_4=0 \\ a_{41}\boldsymbol{X}_4+a_{42}\boldsymbol{X}_2+a_{43}\boldsymbol{X}_3+a_{44}\boldsymbol{X}_4=0 \end{cases} \tag{7}$$

$$\begin{vmatrix} a_{11} & a_{14} \\ a_{41} & a_{44} \end{vmatrix}=0,\ \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix}=0 \tag{8}$$

### 4.2　基于模型检测工具的可达性检测

为了分析和比较基于 ISA 代数模型的检测算法有效性，采用模型检测工具集 CADP[16] 中的 EVELUATOR对上述业务进行可达性检测.

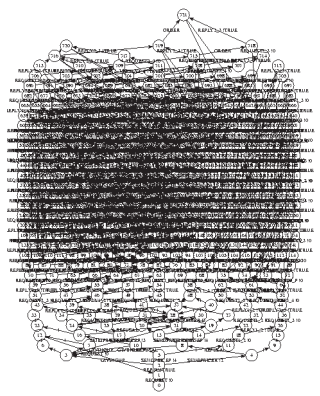
检测过程描述如下：

(1) 建立业务过程的 Lotos 描述，见图 2. 由于篇幅有限，这里仅给出行为描述总揽部分；其中[request, reply, order, refusal, sendpriceP, sendpriceR, givingup]是订货采用的抽象服务原语，这里作为 Requester 与 $Provider_{1\sim4}$ 交互的‘门’；而[request1_2, reply1_2, request1_4, reply1_4]是 $Provider_1$ 与其它进程交互的‘门’，以此类推.

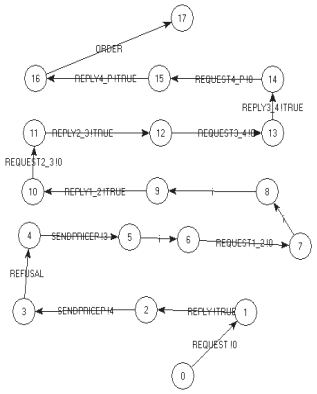
(2) 在 Windows XP-SP3 环境(Intel Core 2、2.09 GHz、2GB)执行 Lotos 程序，结果见图 3. 其中图 3(a)为执行 Lotos 后产生的状态空间图，共产生了 722 个状态，1580 条转换边. 图 3(b)是执行检测条件：“〈true *.″ORDER″〉true”，即可能订货成功的状态转换图；而图 3(c)是执行检测条件：“[true *.′REQUEST1_2 ! * ′] mu X. (〈true〉true and [not ″ORDER″]X)”，即一定订货成功的状态转换图. 可以发现图 3(c)中有一个‘环’，这表明业务在此循环不可达.

```
behaviour
  (Requester [request, reply, order, refusal, sendpriceP, sendpriceR, givingup](0, 2, pMax3, add1))
  |[request, reply, refusal, sendpriceP, sendpriceR, givingup, order]|
  (
   Provider1 [request,reply,refusal,sendpriceP,sendpriceR,givingup,request1_2,reply1_2,request1_4,reply1_4]
             (bs1, times2, minus1)
       |[request1_2, reply1_2, request1_4, reply1_4]|
   (
      Provider2 [request1_2, reply1_2, request2_3, reply2_3] (bs2)
       |[request2_3, reply2_3]|
      Provider3 [request2_3, reply2_3, request3_4, reply3_4] (bs3)
       |[request3_4, reply3_4]|
      Provider4 [request1_4, reply1_4, request3_4, reply3_4, request4_p, reply4_p] (bs4)
       |[request4_p, reply4_p]|
      Producer [request4_p, reply4_p, order] (bs5)
   )
  )
```

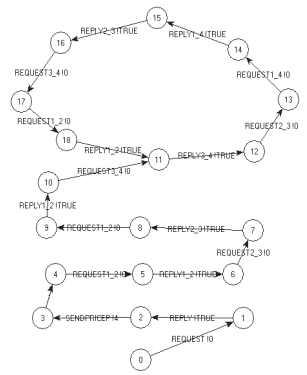
图 2　订货交易过程的 Lotos 描述摘要



(a) 状态空间图




(b) 可能成功的状态转换图




(c) 一定成功的状态转换图

图 3　CADP 检测结果

## 5　相关研究比较及结论

用模型检测方法检测计算可达性是一种常用的方法，与案例分析中类似的工作还有一些，比较相近的是 Nakajima[17] 的工作. 作者结合预定飞机票的服务业务作为被检测对象，用 SPIN 中的 Promela 对业务建模，在考虑了服务内部活动情况下，共形成了

700 行代码,生成了 280000 个状态,470000 个状态迁移.与我们遇到的问题相似,状态空间随着模型的复杂程度急剧增加,如果不进行有效的压缩无法在实际检测中应用.本文提出的基于 ISA 代数模型检测方法对模型状态空间通过"网构相关性"进行有效的控制,部分地解决了状态空间快速增长问题.当然,本文仅仅从网构软件体系结构角度检测计算可达性,并没有从网构行为上考虑计算的可达性.

并行程序的可达性测试是一类经典问题,Hwang 等人[18]针对并行程序因时序不确定而导致的测试困难,结合确定性(deterministic)和非确定性(nondeterministic)测试策略,提出了一个并行程序可达性测试方法.该方法首先选择指定的测试用例,在指定的同步序列(SYN-Sequence)上执行;执行结束后,立即随机选择测试用例在不确定的同步序列上执行,从而测试并行软件的可达性.在 Hwang 等人之后,文献[19]进一步提出了一个并行程序的执行模型,该模型的主要贡献是:提出了多种同步和异步结构用于消息传递和控制,提出了基于时间戳的竞争事件标志方法,提出了同步序列中竞争变量的计算方法,有效地解决了并行程序的可达性测试问题.与本文研究的问题不同,并行程序是在单域设计并工作的程序,而本文面对的是可以在域间工作的、无法进行集中控制的网构,所以不能进行方法的替代.

相关工作还有一些.Schlingloff[20]等人提出一种基于 Petri 网的 Web 服务可达性检测方法,该方法用 BPEL4WS 描述的 Web 服务,并把 BPEL 转换成 Petri 网模型,然后用基于 Petri 网模型检测工具检测可达性.周立等人[21]针对网构软件行为中的不确定性和不完整性,提出了一种支持协商的网构软件体系结构行为建模与验证方法;通过扩展 UML 建模元素支持行为的不确定与不完整建模;提出基于模型检验 Spin 反例引导的正确性验证方法.

本文在 ISA 代数模型基础上,把 ISA 软件可达性检测转化成递归代数表达式和齐次线性方程组非零解判定上来,从而降低了问题求解的代价.用代数学方法建模软件并分析软件属性并非新的做法,作者在文献[14]中介绍了一个 SOA 软件代数模型,本文提出的 ISA 代数模型有部分内容来自于文献[14]中的工作,但不同的是本文从网构软件角度重新审视代数模型方法,为网构计算可达性奠定了基础.本文仅从网构体系结构角度讨论了可达性检测问题,并没有深入到网构行为细节上分析网构软件的特性,我们将深入讨论 ISA 代数模型的行为模型,开展更细致的研究工作.

**致 谢** 本文所述工作得到了北京大学软件研究所梅宏老师的指导,在此表示衷心地感谢!

## 参 考 文 献

[1] Bonnett Kendra. An IBM Guide to Doing Business on the Internet. USA: McGraw-Hill, 2000

[2] Foster I, Kesselman C. The Globus project: A status report//Proceedings of the 7th Heterogeneous Computing Workshop 1998 (HCW 98). Oriando, FL, USA, 1998: 4-18

[3] Marco M, Franco Z. Programming pervasive and mobile computing applications: The TOTA approach. ACM Transactions on Software Engineering and Methodology, 2009, 18(4): Article 15

[4] Huhns M, Singh M P. Service-oriented computing: Key concepts and principles. IEEE Internet Computing, 2005, 9(1): 75-81

[5] Zhang Liang-Jie, Zhou Qun. CCOA: Cloud computing open architecture//Proceedings of the ICWS. Los Angeles, CA, USA, 2009: 607-616

[6] Milner R. Theories for the global ubiquitous computer//Proceedings of the Foundations of Software Science and Computation Structures. LNCS 2987. Berlin: Springer-Verlag, 2004

[7] Booch G. Object-Oriented Analysis and Design with Applications. Reading: Addison-Wesley, 1994

[8] Bachman F, Bass L, Buhman C. Technical concepts of component-based software engineering. USA: CMU/SEI-2000-TR-008, ESC-TR-2000-007, 2000

[9] Michael Stal. Using architectural patterns and blueprints for service-oriented architecture. IEEE Softw, 2006, 23(2): 54-61

[10] Yang Fu-Qing, Mei Hong, Lu Jian, Jin Zhi. Some discussion on the development of software technology. Acta Electronica Sinica, 2002, 30(12A): 1901-1960(in Chinese)
(杨芙清,梅宏,吕建,金芝.浅论软件技术发展.电子学报,2002,30(12A):1901-1906)

[11] Geng Shu-Yun, Qu Wan-Ling, Wang Han-Pin. Discrete Mathematics. Beijing: Peking University Publishing House, 2004(in Chinese)
(耿素云,屈婉玲,王捍贫.离散数学教程.北京:北京大学出版社,2004)

[12] Yeh Wei Jen, Youg Michal. Compositional reachability analysis using process algebra//Proceedings of the Symposium on Testing, Analysis, and Verification. Victoria, British Columbia, Canada, 1991: 49-59

[13] Blurock E S, Buchberger B et al. Reachability test in Petri nets by Gröbner bases. RISC-LINZ Report Series No. 95-03, 27-Jan, 1995

[14] Zhao Hui-Qun, Sun Jing. An algebraic model of service oriented trustworthy software architecture. Chinese Journal of Computers, 2010, 33(5): 890-900(in Chinese)

(赵会群，孙晶. 面向服务的可信软件体系结构代数模型. 计算机学报，2010，33(5)：890-900)

[15] Bellwood T，Clement L，von Riegen C. UDDI-Universal Discovery，Description，and Integration，Version 2.0. Standard UDDI. org，2002. http://www. uddi. org/ipubs/uddi_v3. htm

[16] Hubert Garavel，Frédéric Lang，Radu Mateescu. An overview of CADP 2001//Proceedings of the European Association for Software Science and Technology (EASST) Newsletter. 2002，4：13-24

[17] Nakajima S. Verification of Web service flows with model-checking techniques//Proceedings of the 1st International Symposium on Cyber Worlds. Tokyo，Japan，2002：378-385

[18] Hwang Gwan-Hwan，Tai Kuo-Chung，Huang Ting-Lu. Reachability testing：An approach to testing concurrent software//Proceedings of the Software Engineering Conference. Tokyo，Japan，1994：246-255

[19] Yu Lei，Richard H Carver. Reachability testing of concurrent programs. IEEE Transactions on Software Engineering，2006，32(6)：382-403

[20] Schlingloff Holger，Martens Axel，Schmidt Karsten. Modeling and model checking Web services. Electronic Notes in Theoretical Computer Science，2005，126(8)：3-26

[21] Zhou Li，Chen Xiang-Ping，Huang Gang，Sun Yan-Chun，Mei Hong. Negotiation-enabled modeling and verification of architectural behavior of internetware. Journal of Software，2008，19(5)：1099-1112(in Chinese)
(周立，陈湘萍，黄罡，孙艳春，梅宏. 支持协商的网构软件体系结构行为建模与验证. 软件学报，2008，19(5)：1099-1112)

[22] Zhao Hui-Qun，Wang Guo-Ren，Gao Yuan. An abstract model of software architecture. Chinese Journal of Computers，2002，25(7)：730-736(in Chinese)
(赵会群，王国仁，高远. 软件体系结构抽象模型. 计算机学报，2002，25(7)：730-736)

**ZHAO Hui-Qun**，born in 1960，Ph. D.，professor. His research interests focus on trustworthy software.

**SUN Jing**，M. S.，associate professor. Her research interests include software testing and decision support system.

**WEI Ying**，M. S. candidate. Her research interest is software testing.

**WANG Wen-Wen**，M. S. candidate. Her research interests focus on software architecture.

**GUO Feng**，Ph. D.，lecturer. His research interests focus on model checking.

**Background**

As different types of software paradigm the SOA，the Pervasive Computing，the Cloud Computing have a group of common characteristic. Globalization：each those is composed of autonomous computational entities where activities are not centrally controlled. Heterogeneity：those system are compose of heterogeneous devices (i. e. PDAs，laptops，mobile phones，etc.) that provide different configurations and functionalities. Mobility：each computational entity is mobile，due to the movement of the physical platforms or by movement of entities from one platform to another. User-Dependent：the end-user of a GCS is always the source of each change and a GCS must be able to adapt itself to make the user's task easier. Fault-Tolerance：GCSs provide mechanisms to guarantee that faults in the system do not interrupt a service delivery. Scalability：GCSs are able to start small and then expand over time in terms of size(i. e.，more number of users，devices and connections) and functionalities(i. e.，new service request) insuring the system availability.

Internetware is constructed by a set of autonomic software entities distributed over the Internet，and a set of connectors enabling the collaboration among these entities in various manners. Its objective are improving and fulfilling all above characteristics.

Exploded state space where an Internetware involved in is big challenge for searching a usable software entity. The authors get the judgment from the observation on Model Checking. In order to conquer the challenge a new method based on the algebraic model of Internet Ware for checking his reachability is proposed. With respect to the attributes of Internet Ware an algebraic model is proposed first，and then it is fully developed by introducing a series of new concept，for example Internet Ware Correlative and Internetware Space. By discovering the relationship between reachability of Internet ware software and Internet ware Operation Expression，converts the reachability of Internet ware software into the recursion of Internet ware Operation Expression. By mapping the Internet Ware Space to Linear Space translates his linear correlativity checking into solving his equations. All above effort can cut the volume of spaces down dramatically with proposed algorithm. Finally，the application aspects of checking method are discussed too.

This work was fully supported by National Natural Science Foundation of China (Grant Nos. 61070030，61111130121)，and Innovative Scientific Research team of Beijing Education Committee (Grant No. PHR201107107).