ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.5, May 2008, pp.1186−1200
DOI: 10.3724/SP.J.1001.2008.01186

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

网构软件的资源自适应性的形式化分析与验证*

胡 军[1,3+], 黄志球[1], 曹 东[2], 徐丙凤[1]

[1](南京航空航天大学 信息科学与技术学院,江苏 南京 210016)
[2](南京航空航天大学 自动化学院,江苏 南京 210016)
[3](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

Formal Analysis and Verification of Resource Adaptability for Internetware

HU Jun[1,3+], HUANG Zhi-Qiu[1], CAO Dong[2], XU Bing-Feng[1]

[1](College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)
[2](College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)
[3](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)
+ Corresponding author: E-mail: hujunnju@ieee.org

Hu J, Huang ZQ, Cao D, Xu BF. Formal analysis and verification of resource adaptability for Internetware. *Journal of Software*, 2008,19(5):1186−1200. http://www.jos.org.cn/1000-9825/19/1186.htm

Abstract: This paper considers the formal analysis and verification methods for the environmental resource adaptability of component-based Interneware. Firstly, the resource interface automata is used to model the behaviors of component interfaces which contain the information of dynamical resource utilization, and the resource interface automaton networks are constructed to describe the compositional behaviors of the system. At the same time, the UML sequence diagrams are used to model the specifications of specified behaviors in the compositional system. Then, two typical problems are proposed and analyzed formally, one of them is to check whether all behaviors of a system are satisfied within the specified resource constraints, and the other is the verification of whether the specified behaviors are satisfied when the resources have been changed. Based on analyzing the compatible state space of the resource interface automata networks, a corresponding reachability graph is constructed, and finally several algorithms are developed for the above problems.

Key words: Internetware; component-based design; resource adaptive; formal verification; UML

摘 要: 针对基于构件的网构软件系统对环境资源变化的自适应性特征的可信分析与验证展开研究.具体工作包括:在网构软件的系统模型层次,使用带资源语义信息的接口自动机对软件构件的行为进行形式化建模,其包含了构件在完成特定功能的过程中对环境资源的使用特征;使用资源接口自动机网络来描述构件组装实体的组合行为;使用基于场景的UML顺序图模型来描述具有多功能的组合系统规约;分别研究了检验组合系统的所有行为是否都满足给定的资源约束以及检验指定的系统行为是否满足资源约束这两个具体问题;通过对资源自动机网络状态空间

* Supported by the Foundation of Aeronautics Science of China under Grant No.2007ZD52043 (航空科学基金); the Foundation of the Special Research Fund for the Doctoral Program of Higher Education of China under Grant No.20070287052 (高等学校博士学科点专项科研基金)

Received 2007-11-15; Accepted 2008-03-19

的分析,构造其相应的可达图,在此基础上给出了相应的检验资源可满足性、最小资源需求量以及检验指定功能合法性等算法.

关键词: 网构软件;构件式设计;资源自适应;形式化验证;统一建模语言

中图法分类号: TP311 **文献标识码:** A

由于计算机软件系统所基于的硬件平台正逐步从集中、静态和封闭的计算平台向开放、动态和多变的Internet平台转变,因此,软件系统也开始呈现具有自主适应、动态协同、在线演化、连续反应等形态特征.在文献[1,2]中,具有这些形态特征的软件系统称为网构软件(Internetware).网构软件的设计、分析、验证、部署与运行对目前的软件基本模型、开发技术、生命周期、体系结构以及生产环境等都带来了一系列的挑战.

对基于 Internet 环境的网构软件系统而言,其基本的计算单元应该是分布、自治和异构的软件构件实体.基于构件的软件系统具备系统可扩展性好、演化能力强、对已有的技术进行很强的重用能力等优点[3],因此,可以采用自底向上的基于软件构件的技术来构造在开放、动态的 Internet 环境下的网构软件系统.另一方面,这些构件实体以开放、自主的方式存在于 Internet 的各个节点上,通过各种协同方式与其他构件实体进行跨网络的协作和联盟,网络环境的开放与动态性以及用户使用方式的个性化要求决定了这样构成的网构软件应该能够感知外部环境的变化并随着这种变化按照功能、性能或可信度指标等进行合适的调整和演化.因此,网构软件对环境的自适应能力构成了其最基本的一个特征.

从软件生命周期的角度来看,在使用基于构件的技术来构造基于Internet的网构软件时,其中一个重要挑战是:如何在系统分析设计阶段以及系统在线演化阶段对由于环境资源的变化所导致的构件组合系统的行为变化进行有效的分析与验证,以提高系统在自适应性方面的可信度.

本文的工作在网构软件的系统模型层次,使用带资源语义信息的接口自动机对软件构件的行为进行形式化建模,其包含了构件在完成特定功能的过程中对环境资源的使用特征;使用资源接口自动机网络来描述构件组装实体的组合行为;使用基于场景的 UML 顺序图模型来描述具有多功能的组合系统规约;针对检验组合系统的所有行为是否都满足给定的资源约束和检验指定的系统行为是否满足资源约束这两个具体问题展开研究;通过对资源自动机网络状态空间的分析,构造其相应的可达图,并在此基础上给出了相应的检验资源可满足性、最小资源需求量以及检验指定功能合法性等算法.

本文第1节给出以上两个具体问题的描述以及对环境资源使用特征的假设前提说明.第2节给出资源接口自动机模型的非形式化描述和形式化定义,并给出作为构件组装实体的组合系统行为模型的资源接口自动机定义.第 3 节对在给定环境的资源约束下组合系统的所有行为是否都满足约束这个问题展开形式化分析,并给出检验资源可满足性、最小资源需求量两种算法.第4节使用基于场景的UML顺序图来描述组合系统接口行为的规约,对当环境资源发生变化时,组合系统指定功能是否仍然满足约束的问题展开形式化分析,并给出检验指定功能合法性的算法.最后是相关研究工作和结束语.

1 问题描述

网构软件的自适应性能力一般包括两方面的含义:其一是对外部环境的变化作出适当的反应;其二是对系统需求的变化作出适当的相应反应;从而使得系统所提供的服务功能或性能等维持在一个令人满意的、具有一定可信度的水平上[2].本文的研究工作主要是针对前一种考虑环境中资源发生变化的构件系统的自适应性以特征.

具体来说,本文在网构软件的系统模型层次,关注以下两个问题:

1) 检验构件组合系统的所有行为是否都满足给定的资源约束:在给定的资源约束下,构件组合系统在运行过程中,其所有的行为是否都没有违反资源使用的限制.
2) 检验构件组合系统中指定的功能行为是否满足资源约束:在给定的资源约束下,组合系统中我们所关心的一些指定功能是否一定满足资源限制.

上述第 1 个问题表明,在给定的资源限制条件下,需要检查组合系统中所有可能的行为是否都满足资源约束;只要在系统的状态空间中出现 1 个资源使用异常状态,这个问题的回答就是否定的.同时,我们可以根据检验结果,找出系统在哪些状态上是资源使用异常,以进一步检查系统是通过哪些行为到达这些异常状态.当检验的结果是肯定的时,我们认为系统所有的行为一定不会违反资源约束.显然,在这种情况下,系统运行的可信度是最高的.第 2 个问题则是在前一个问题的基础上进一步表达了在给定的资源约束下,即使组合系统中有可能出现某些异常的资源使用状态,我们所关心的一些比较重要或者基本的功能行为是否也能正常完成.事实上,这个问题表达了当系统运行的外界环境资源发生变化,导致系统的某些状态可能会出现资源使用异常,使得某些行为不能正常执行的时候,系统中另外一些我们所关心的重要或基本的功能行为仍然保留在那些资源使用正常的状态空间中,仍然可以得到正常的实现.这实际上就表达了系统具有较好的环境适应性.

在第 2 节中,我们将对接口自动机的基本模型进行资源语义信息方面的扩展,使得扩充的构件接口模型可以比较合理地表达系统对资源使用方面的性质,并且可以在系统模型这个抽象层次上进行分析.为此我们给出系统资源使用的几个基本假设如下:

1) 资源是可以量化的.这意味着在使用资源时,总是以某种基本单位的倍数来进行使用和归还.
2) 资源的使用是独占的.即一份资源在被使用的时候,不可以同时被另外一个构件使用,是属于非共享式的资源.
3) 资源是可以回收(归还)的,但回收的机制以及相应的代价暂不考虑.

在网构软件系统中通常所考虑的一些系统资源包括内存、消息队列、缓冲区等都满足以上基本假设,也可以包括传感器、感应器等受网构软件控制的外部设备.

2 资源接口自动机及自动机网络

软件构件对外表现的接口性质包括静态和动态两个方面.静态的是指接口参数的数量、类型的要求.动态的是指接口与外界环境之间交互的行为要求,可以看作是构件与构件环境之间某种形式的交互协议.接口自动机(interface automata,简称 IA)[4,5]是用来刻画软件构件接口交互行为时序特征的一种形式化语言.它较好地描述了使用构件时其对外界环境的输入假设和输出保证,即构件内方法被调用的先后次序以及构件向外环境输出调用信息或结果的次序.接口自动机及其组合的具体细节见文献[4,5].

本文中采用接口自动机作为基本模型对网构软件系统中构件实体的外部交互行为进行建模.由于在构件的组合中需要考虑外部环境的变化对组合行为的影响,因此,本文对接口自动机基本模型进行了资源语义信息的扩展,称其为资源接口自动机(resource interface automata,简称 RIA).从形式上看,在 IA 基本模型中至少需要添加两类资源语义信息,一类是资源使用的数量信息,一类就是资源的动态使用和归还信息.一般来说,我们认为系统停留在某一个状态上可能会需要占用一定数量的不同种类的资源.这里所谓的“停留在某个状态”的含义是指系统在这个状态上完成了某个我们所关心的特定的计算任务,比如经过计算得到了某个需要的输出结果或者改变了某些变量的值等等.由于系统的状态表示了不同粒度大小的计算单元(计算任务),因此我们认为,在 IA 模型中将资源使用信息作为系统状态的一种属性标识在系统的状态上比较合理,而没有将资源作为系统状态之间的转换能不能发生的条件之一.

2.1 资源接口自动机模型的非形式描述

首先,我们给出一个资源接口自动机(resource interface automata,简称 RIA)模型的例子.如图 1 所示,通过这个添加了资源使用信息的接口模型,构件 J 表达了它与环境之间的交互行为以及在这个交互过程中对环境资源使用的动态特征.在此模型中,构件 J 需要两类资源,分别表示为 R_1,R_2 我们在每一个状态上都标示一个有序数对 $\langle b_1,b_2\rangle$ (其中, b_1,b_2 都是非负整数),并称此有序数对为状态的资源特征向量.它表示系统在此状态上需要占用 b_1 个数量单位的资源 R_1 和 b_2 个数量单位的资源 R_2.在图 1 中的初始状态 0 上,构件的资源特征向量为 $\langle 0,0\rangle$,表示构件实体在初始状态的时候不占用这两种资源.但这并不是强制性的,有可能在初始状态时,构件需要某些数

量的资源来进行状态的初始化.然后构件通过输入动作 a 转换到状态 1,状态 1 上的资源特征向量为 $\langle 4,2\rangle$, 表明要使得构件维持在状态 1 上的正常运行,需要提供 4 个 R_1 资源和 2 个 R_2 资源.如果在此状态上,构件不能从环境中获得正常所需要的资源数量,那么它将无法维持在状态 1 上的正常运行.当出现这种情形时,我们称此状态为资源使用异常状态.当环境提供的资源满足需求时,构件就可以正常地完成在状态 1 上的计算任务,然后通过输出动作 d 的触发,构件状态转换到 2.此时,在状态 2 上的资源特征向量为 $\langle 1,1\rangle$,资源数量发生了变化,这表示在从状态 1 到状态 2 的转换过程中,构件归还了 3 个已经占用的 R_1 资源,1 个 R_2 资源.J 的后续行为都可以如此类推.这样,通过每个状态上的资源特征向量以及状态转换之间不同种类资源数量上的相应变化,资源接口自动机模型比较恰当地表达了构件在与环境之间的交互过程中的资源使用的动态信息.

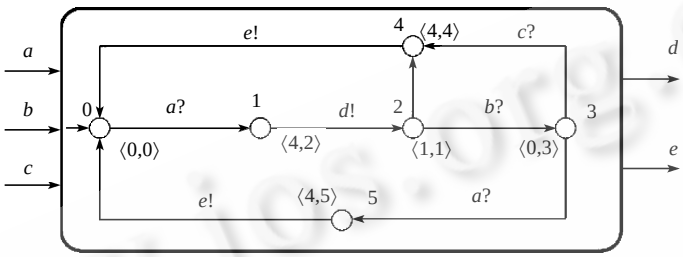


Fig.1 A resource interface automata model for a component J

图 1 构件 J 的资源接口自动机模型

2.2 资源接口自动机(RIA)的形式化定义

RIA 的形式化定义如下:

定义 1. 一个资源接口自动机是一个多元组: $P=(V_P,A_P,F_P,W_P,\Gamma_P)$, 其中:

- V_P 是一个有穷的状态集,其中每一个状态 $v_i\in V_P$,且 P 的初始状态表示为 v_P^{init}.
- A_P 是一个有穷的动作集,包括 Input,Output 和 Internal 三种动作集合,即 $A_P=A_P^I\cup A_P^O\cup A_P^H$ 且 A_P^I, A_P^O, A_P^H 互不相交.
- F_P 是一个有穷的映射集, $F_P=\{f_P^1,f_P^2,...,f_P^K\}$ (K 为正整数,表示系统中可使用资源的种类数);其中每一个映射 $f_P^i:V_P\mapsto R^+\cup\{0\}$ 是将 V_P 中任意一个状态都应设为一个非负整数.用 $F_P(v_i)$ 表示有序集 $\langle f_P^1(v_i),f_P^2(v_i),...,f_P^K(v_i)\rangle$.
- W_P 是一个有穷集,每一个元素形如: $w_i=\{(v_i,F_P(v_i))\,|\,v_i\in V_P\}$;简记为 $W_P=(V_P,F_P(V_P))$, 其中: $w_P^{Init}=\{(v_P^{Init},F_P(v_P^{Init}))\,|\,v_P^{Init}\in V_P\}$.
- Γ_P 是转换集, $\Gamma_P\subseteq W_P\times A_P\times W_P$.

我们称 W_P 中的每一个 w_i 为一个带资源约束的状态,简称为资源状态.有序数集 $F_P(v_i)=\langle f_P^1(v_i),f_P^2(v_i),...,f_P^K(v_i)\rangle$ 就是状态 v_i 上的资源特征向量,简称为资源向量.

下面给出 RIA 的行为的定义.

定义 2. 设一个 RIA 为 $P=(V_P,A_P,F_P,W_P,\Gamma_P)$,则 P 的一个行为表示为一个资源状态的转换序列 ρ,形如: $\rho=w_0\xrightarrow{a_0}w_1\xrightarrow{a_1}...\xrightarrow{a_{m-2}}w_{m-1}\xrightarrow{a_{m-1}}w_m$,其中: $w_i=(v_i,F_P^{v_i})$, $w_0=(v_P^{Init},F_P^{Init})$,且对于每一个 i,都有 $(w_i,a_i,w_{i+1})\in\Gamma_P$.

显然,RIA 中每一个资源状态转换序列中都包含了一个不带资源约束的状态转换序列,形如: $v_0\xrightarrow{a_0}v_1\xrightarrow{a_1}...\xrightarrow{a_{m-1}}v_m$.在本节中,我们将 RIA 中行为 ρ 的不带资源标记的状态转换序列称为 ρ 相应的功能行为,记为 $Func(\rho)$.

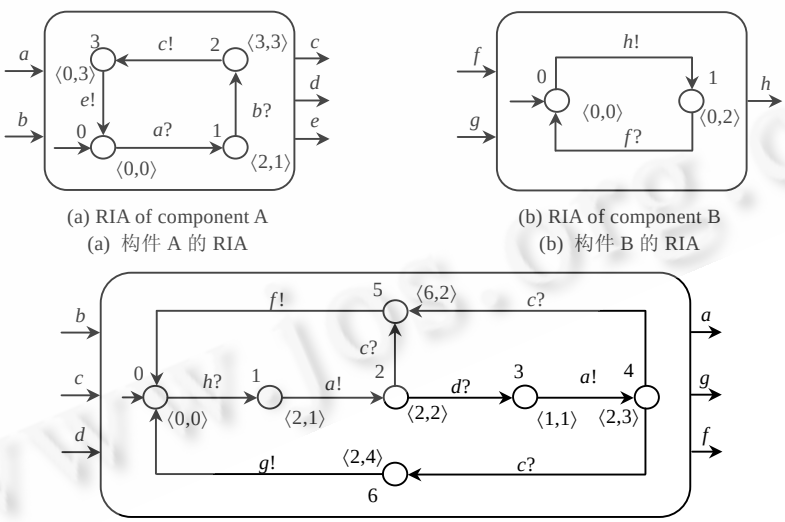
事实上,定义 2 中的资源状态的转换序列 ρ 表达了在构件接口行为的一个状态变化过程中,构件对环境资源

的动态使用信息.如:在上述的例子中,构件 J 的一个资源状态转换序列为 $\rho_1=(0,\langle 0,0\rangle)\xrightarrow{a?}(1,\langle 4,2\rangle)\xrightarrow{d!}(2,\langle 1,1\rangle)\xrightarrow{c?}(4,\langle 4,4\rangle)\xrightarrow{e!}(0,\langle 0,0\rangle)$, 它表明:在构件 J 的一个功能行为 $Func(\rho_1)=0\xrightarrow{a?}1\xrightarrow{d!}2\xrightarrow{c?}4\xrightarrow{e!}0$ 中,对资源 R_1 的使用情况是:构件在初始状态 0 上不需要任何 R_1 资源,在状态 2 上需要 1 个 R_1 资源,而在状态 1 和状态 4 上都需要占用 4 个 R_1;同时,转换 $1\xrightarrow{d!}2$ 和 $4\xrightarrow{e!}0$ 也表示当这两个转换发生时,构件将分别释放 3 个和 4 个所占用的 R_1 资源.对于资源 R_2 而言,我们也可以作出类似的语义解释.

基于以上的观察,我们可以从 P 的一个行为 ρ 中抽取每个状态上相应的资源特征向量,得到一个只包含资源使用变化信息的序列: $F_P(v_0)\hat{}F_P(v_1)\hat{}F_P(v_2)\hat{}...\hat{}F_P(v_m)$, 我们称此序列为行为 ρ 的资源轨迹,记为 λ_ρ. 例如,在前面给出的那个构件 J 的资源行为 ρ_1 中,其资源轨迹为 $\lambda_{\rho_1}=\langle 0,0\rangle\hat{}\langle 4,2\rangle\hat{}\langle 1,1\rangle\hat{}\langle 4,4\rangle\hat{}\langle 0,0\rangle$.然后,我们进一步将 λ_ρ 中的每一个资源向量展开为分量的形式,得到: $\langle f_P^1(v_0), f_P^2(v_0),..., f_P^k(v_0)\rangle\hat{}\langle f_P^1(v_1), f_P^2(v_1),..., f_P^k(v_1)\rangle\hat{}...\hat{}\langle f_P^1(v_m), f_P^2(v_m),..., f_P^k(v_m)\rangle$, 并将记录同一类资源信息的分量提取出来组成新的分量序列,一共有 K 条序列,形如: $f_P^i(v_0)\hat{}\ f_P^i(v_1)\hat{}...\hat{}\ f_P^i(v_m)$ (其中,$1\le i\le K$).显然,每一条分量序列 $f_P^i(v_0)\hat{}\ f_P^i(v_1)\hat{}...\hat{}\ f_P^i(v_m)$ 就表示资源 R_i 在行为 ρ 中的使用变化情况,我们称上述资源分量序列为行为 ρ 关于 R_i 的资源轨迹,记作 $\lambda_\rho(R_i)$.比如,上例中行为 ρ_1 关于资源 R_1 的轨迹为 $\lambda_{\rho_1}(R_1)=0\hat{}4\hat{}1\hat{}4\hat{}0$,关于资源 R_2 的轨迹为 $\lambda_{\rho_1}(R_2)=0\hat{}2\hat{}1\hat{}4\hat{}0$.显然,从这些关于各个资源分量的使用数量的序列中可以很容易地看出在行为 ρ_1 中,这两类资源是如何动态变化的信息.

2.3 资源接口自动机网络(RIA-Networks)

我们使用资源接口自动机网络(resource interface automaton networks,简称 RIA-Networks)作为网构软件系统中构件组合系统的带资源约束信息的行为模型.组合系统中的每一个构件的资源使用行为都使用一个相应的资源接口自动机来表示.整个构件组合系统由一个接口自动机集合组成,并通过构件接口之间交互的共享动作来进行同步组合;如图 2 中表示了由 3 个构件 A,B,C 的资源接口自动机所构成的组合系统的行为模型.



(a) RIA of component A
(a) 构件 A 的 RIA

(b) RIA of component B
(b) 构件 B 的 RIA

(c) RIA of component C
(c) 构件 C 的 RIA

Fig.2 RIA-Networks

图 2 资源接口自动机网络

定义 3. 一个资源接口自动机网络(RIA-Networks)是一个二元组 (Q,Z), 其中:

- $Q=\{P_1,P_2,...,P_n\}$ 为可组合的接口自动机集;
- $Z=\{shared(P_i,P_j)\,|\,1\le i,j\le n,i\ne j\}$, 为所有的共享动作集,其中任何 P_i 和 P_j 的共享动作集为 $shared(P_i,$

$P_j)=A_{P_i}\cap A_{P_j}=\left(A_{P_i}^O\cap A_{P_j}^I\right)\cup\left(A_{P_i}^I\cap A_{P_j}^O\right)(1\le i,j\le n,i\ne j).$

下面给出 RIA-Networks 的资源状态集和转换集.

定义 4. 设 $N=(Q,Z)$是一个 RIA-Networks,其中, $Q=\{P_1,P_2,...,P_n\}$,且 $P\ P_i=(V_{P_i},A_{P_i},W_{P_i},\Gamma_{P_i})(1\le i\le n)$, 则 N 中的组合状态集和动作集定义如下:

- N 的每一个不带资源向量的组合状态都形如: $\overline{v}=(v_1,v_2,...,v_n)$,其中, $v_i\in V_{P_i},1\le i\le n$.组合状态的集合为 $V_N=V_{P_1}\times V_{P_2}\times...\times V_{P_n}$, 其中,无资源向量的初始组合状态为 $\overline{v}_N^{init}=\left(v_{P_1}^{init},v_{P_2}^{init},...,v_{P_n}^{init}\right)$.
- N 中的每一个资源组合状态形如: $\overline{w}=(\overline{v},F_N(\overline{v}))$,其中: $F_N(\overline{v})$ 是一个资源向量标记: $\left\langle f_N^1(\overline{v}),f_N^2(\overline{v}),...,f_N^K(\overline{v})\right\rangle$, 对于每一个 $i(1\le i\le K)$, 有 $f_N^i(\overline{v})=\sum_{j=1}^n f_{P_j}^i(v_j)$; 初始的资源组合状态为 $\overline{w}_N^{Init}=(\overline{v}_N^{Init},F_N(\overline{v}_N^{Init}))$; N 中所有的资源组合状态集记为 W_N.
- N 的动作集为 $A_N=A_N^I\cup A_N^O\cup A_N^H$,其中:输入动作集为 $A_N^I=\left(\bigcup_{1\le i\le n}A_{P_i}^I\right)\Big/Z$, 输出动作集为 $A_N^O=\left(\bigcup_{1\le i\le n}A_{P_i}^O\right)\Big/Z$, 内部动作集为 $A_N^H=\left(\bigcup_{1\le i\le n}A_{P_i}^I\right)\cup Z$.

RIA-Networks 的转换集定义如下:

定义 5. 设 $N=(Q,Z)$是一个 RIA-Networks,其中: $Q=\{P_1,P_2,...,P_n\}$, 且 $P_i=(V_{P_i},A_{P_i},F_{P_i},W_{P_i},\Gamma_{P_i})(1\le i\le n)$; $\overline{w}=(\overline{v},F_N(\overline{v}))$ 和 $\overline{w}'=(\overline{v}',F_N(\overline{v}'))$ 是 N 中两个不同资源组合状态,其中: $\overline{v}=(v_1,v_2,...,v_n)$; $\overline{v}'=(v_1',v_2',...,v_n')$; $F_N(\overline{v})$ 与 $F_N(\overline{v}')$ 是相应的资源向量标记.当满足以下条件之一时,系统可以通过动作 a 从状态 $\overline{w}$ 到达状态 $\overline{w}'$, 用 $\overline{w}\xrightarrow{a}\overline{w}'$ 来表示:

- 对于一个动作 $a\notin Z$, 在 P_i 中存在一个转换 $(w_k,a,w_k')\in\Gamma_{P_i}$, 其中 $w_k=(v_k,F_{P_i}(v_k)),w_k'=(v_k',F_{P_i}(v_k'))$; 且对于任何其他的 $l(l\ne k,1\le l\le n)$,有 $w_l=w_l'$.
- 对于一个动作 $a\in shared(P_i,P_j)(1\le i,j\le n,i\ne j)$, 在 P_i 中存在一个转换 $(w_i,a!,w_i')\in\Gamma_{P_i}$ ($a!$表示 a 是输出动作);同时,在 P_j 中存在一个转换 $(w_j,a?,w_j')\in\Gamma_{P_j}$ ($a?$表示 a 是输入动作),且对于任何的 $k(k\ne i,j,1\le k\le n)$, 有 $w_k=w_k'$.

基于以上组合状态和转换的定义,下面给出 RIA-Networks 的行为定义.

定义 6. N 为一个资源接口自动机网络 (RIA-Networks), N 的一个行为就是一个资源状态转换序列 $\overline{w}_0\xrightarrow{a_0}\overline{w}_1\xrightarrow{a_1}...\xrightarrow{a_{m-1}}\overline{w}_m$,其满足:

- $\overline{w}_0=\overline{w}_N^{Init}$;
- 对于每一个 $i(0\le i\le m)$, 有 $(\overline{w}_i,a_i,\overline{w}_{i+1})\in\Gamma_N$.

由接口自动机基本模型中关于非法状态的定义[4,5],在 RIA-Networks 中同样会存在交互次序上的非法状态问题.下面我们同样给出带资源约束的非法交互组合状态的定义:

定义 7. 设 $N=(Q,Z)$是一个 RIA-Networks,其中 $Q=\{P_1,P_2,...,P_n\}$,其非法交互状态为

$$illegal(N)=\left\{(\overline{v},F_N(\overline{v}))\in N\left|\begin{array}{l}\exists(v_i,v_j)(i\ne j;1\le i,j\le n),\exists a\in shared(P_i,P_j),\\ \left(\begin{array}{l}a\in A_{P_i}^O(w_i)\wedge a\notin A_{P_j}^I(w_j)\\ \vee\\ a\in A_{P_j}^O(w_j)\wedge a\notin A_{P_i}^I(w_i)\end{array}\right)\end{array}\right.\right\},$$

其中, $\overline{v}=(v_1,v_2,...,v_n)$.

使用文献[4]中给出的逆向可达性算法,我们可以构造出只包含可达的兼容组合状态的资源接口自动机网络,用 $com(N)$来表示.当 $com(N)$中的状态集为空时,则表示不存在合法的环境使得整个组合系统可以正常工作.在本节以下的讨论中将只考虑 $com(N)$不为空的情形,即后面所讨论的组合系统行为都是在可兼容的状态空间中的资源使用行为.

3 检验组合系统的所有行为是否都满足给定的资源约束

首先,我们给出构件组合系统的环境资源约束条件,以资源约束向量的形式给出 $\vec{B}=\langle b_1,b_2,...,b_K\rangle$;其中的每一个分量 $b_i(1\le i\le K)$ 表示当前环境中可利用资源 R_i 的最大可使用量.以下给出组合系统中对资源使用的异常状态的定义.

定义 8. 设 N 为一个 RIA-Networks,$N=(Q,Z)$,其中:$Q=\{P_1,P_2,...,P_n\}$,且 $P_j=(V_{P_j},A_{P_j},F_{P_j},W_{P_j},\Gamma_{P_j})$ $(1\le j\le n)$;若 N 中的某一个组合状态 $\overline{w}$ (形如:$(\overline{v},F_N(\overline{v}))$,其中 $\overline{v}=(v_1,v_2,...,v_n)$ 满足以下条件时,则称 $\overline{w}$ 为资源使用异常状态,简称为异常状态:

- 相对于资源约束条件 $\vec{B}$ 而言,存在一个 $i(1\le i\le K)$ 使得 $\sum_{j=1}^{n}f_{P_j}^{i}(v_j)>b_i$.
- 基于 $com(N)$,我们可以构造一个相应的可达图 $N=(S,L)$,构造方法如下:
- S 是有穷节点集,$com(N)$中每一个资源组合状态 $\overline{w}_i$ 就对应于 S 中的一个节点 s_i;起始节点 s_0 就对应于 $com(N)$中的初始状态 $\overline{w}_N^{Init}$;
- L 是有穷的边集,$com(N)$中的每一个转换 $(\overline{w}_i,a_i,\overline{w}_{i+1})\in\Gamma_{com(N)}$ 就对应到图 G 中一条从节点 s_i 到 s_{i+1} 的转换边 $l_i=(s_i,a_i,s_{i+1})$.

由以上的构造方法可知,可达图 G 中的每一个节点 s_i 都有与相应组合状态中一致的资源向量属性,我们用 $\overrightarrow{X_{s_i}}$ 来表示. $\overrightarrow{X_{s_i}}$ 也是一个长度为 K 的非负整数向量,形如:$\left\langle x_{s_i}^1,x_{s_i}^2,...,x_{s_i}^K\right\rangle$. 注意到,由于我们对一个 RIA-Networks 中的非法交互状态和资源使用异常状态是分别定义的,因此即使是在 $com(N)$中,仍然有可能出现异常状态.那么,当可达图中某个节点 s_j 所对应的资源组合状态 w_i 是一个异常状态时,即相对于资源约束条件 $\vec{B}$ 而言,至少存在 1 个 $\overrightarrow{X}_{s_j}[i]$ 满足 $\overrightarrow{X}_{s_j}[i]>b_i(1\le i\le K)$ 时,称节点 s_j 是一个异常节点.当可达图 G 中不存在异常节点时,我们称满足这种性质的可达图为规范的可达图 RG(regular reachability graph).

图 3 所示即为上一节中由 3 个构件 A,B,C 组合成的可兼容组合状态空间 $com(A\otimes B\otimes C)$ 所构造的可达图,有 s_0~s_6 共 7 个节点,在每个节点上给出了相应的资源向量标记.当给定的资源约束为 $\vec{B}=\langle 7,8\rangle$ 时,可以判断出可达图中每一个节点都满足资源的约束,因此,这是一个规范的可达图.但是,当环境资源发生变化,调整为 $\vec{B}=\langle 5,8\rangle$ 时,我们可以发现,在节点 s_6 上的资源向量满足 $\overrightarrow{X}_{s_0}[1]=6>5$,即满足定义 8,此时我们说 s_6 是一个异常的节点,图 3 也不是一个规范的可达图.

基于以上讨论,下面我们给出检验组合系统的资源使用可满足性的定理.

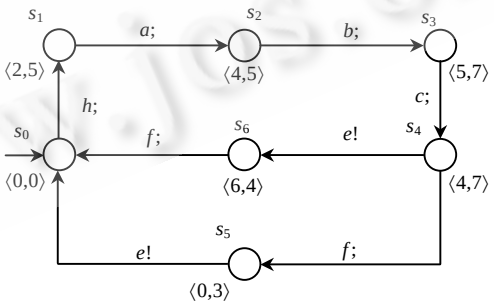


Fig.3 A reachability graph G of $com(A\otimes B\otimes C)$

图 3 由 $com(A\otimes B\otimes C)$ 构造的可达图 G

定理 1. N 是一个 RIA-Networks,其满足某个资源使用约束 $\vec{B}$ 当且仅当其兼容状态空间的可达图 G 是一个规范的可达图.

定理证明见附录.基于定理 1,可以构造一个深度优先的算法来判断一个可达图 G 是否成为规范的可达图.

我们只需要遍历可达图 G 中的每一个节点,检查其上的资源向量标记就可以了.这只需要常规的深度优先搜索框架就可以做到,具体内容见算法 1.算法的输入为可达图 G 和资源约束向量 $\vec{B}=\langle b_1,b_2,...,b_K\rangle$,输出结果为 G 是否为一个规范的可达图以及当 G 不是一个规范的可达图时,图中的异常节点集合.在算法 1 中,*current_path* 中保存的是当前对图 G 的遍历路径,$\vec{B}$ 中存放长度为 K 的资源约束向量; 我们使用 $\vec{X}(node)$ 表示当前访问节点 *node* 上的资源向量,*abnormal* 集中存放图 G 中找到的异常节点,*satisfied* 是一个布尔变量,当图 G 中不存在异常节点时,*satisfied* 为“true”,即图 G 是一个规范的可达图,否则 *satisfied* 为“false”.

算法 1. 检验资源可满足性算法.

$current\_path:=\{s_0\};\vec{B}:=\langle b_1,b_2,...,b_K\rangle;$

$\vec{X}:=\langle 0,0,...,0\rangle;abnormal:=\varnothing;satified:=\text{true};$

repeat

node:=取 *current_path* 中最后一个节点;

if *node* 的后继节点均已访问过 **then** 删除 *current_path* 中的最后一个节点;

else begin *node*:=取 *node* 的一个未访问过的后继节点;

$\vec{X}:=\vec{X}(node);$

for i=0 **to** K–1

if $\vec{X}[i]>b_i$ **then** *satisfied*:=false;

if *satisfied* = false **then** 将 *node* 加入到 *abnormal* 集中;

将 *node* 加入到 *current_path* 中;

end

until *current_path*={};

if *abnormal*:=∅ **then return** true **else return** false.

就图 3 中的可达图而言,当资源约束向量为 $\vec{B}=\langle 5,8\rangle$ 时,节点 s_6 为异常的节点,检验的结果为 false,集合 *abnormal* 中存放的就是节点 s_6.事实上,与上述给定资源限制下组合系统所有行为合法性验证问题相对应的是一个组合系统资源最少需求量的设计问题,即给定一个由 n 个构件组合而成的系统,需要 K 中不同种类资源,要使得这个组合系统可以正常运行,完整地体现所有的组合功能,它对 K 种资源各自的最少需求量是多少.在本文中,n 个构件各自的资源使用行为特征是用相应的资源接口自动机来描述的,组合系统的行为就是一个资源接口自动机网络.基于前面的讨论,上述系统资源最少需求量的计算问题实质上就是找出 $com(N)$的可达图 G 中各个节点上的不同种类资源使用量的最大值.同样,我们可以通过对可达图 G 进行深度优先遍历,查找每一个节点并进行比较来实现.算法框架见算法 2.算法的输入是可达图 G,输出是资源使用量 $\vec{B}$,也是一个长度为 K 的向量.如:在可达图 3 中使用上述算法,可以求出系统中两类资源各自的至少使用需求量为 $\vec{B}=\langle 6,7\rangle$.这两个算法的复杂度与通用的图深度优先搜索算法框架的复杂度相同.

算法 2. 计算系统资源最少需求量.

$current\_path:=\{s_0\};\vec{B}:=\langle 0,0,...,0\rangle;\vec{X}=\langle 0,0,...,0\rangle;$

repeat

node:=取 *current_path* 中最后一个节点;

if *node* 的后继节点均已访问过 **then** 删除 *current_path* 中的最后一个节点

else begin *node*:=取 *node* 的一个未访问过的后继节点;

$\vec{X}:=\vec{X}(node);$

for i=0 **to** K–1

if $\vec{X}[i]>b_i$ then $b_i:=\vec{X}[i];$

将 *node* 加入到 *current_path* 中;

end

until *current_path*={};

4 基于场景的自适应性分析与验证

在上一节中,如果检验算法(算法 1)给出的结果为 false,则表明在组合系统 RIA-Networks 所有可能的行为中存在某些行为违反了给定的资源约束条件.在通常的构件系统中,当出现这种情况时,则认为这个组合系统在资源的使用上是不合法的,构件组合的结果是无效的、失败的,需要重新选择构件进行新的组合.我们认为这是一种比较“悲观”的处理方式.实际上,在网构软件中应当采用更加“乐观”的处理方式;当环境资源发生变化时,虽然组合系统的所有行为不一定能够全部满足资源的约束条件,但我们并不是简单地否定或放弃这个组合系统,而是希望组合系统的某些特定功能或基本功能在资源不足的环境中仍然能够得到正常的执行.这其实就是表达了网构软件具有一定的环境自适应能力.在这种情况下,虽然组合系统的整体行为的可信度降低了,但是系统的那些特定功能或基本功能的可信度仍然必须得到严格的保证.对于我们关注的这些功能行为,可以使用组合系统中构件间的基于交互场景的 UML 顺序图模型[6]来给出规约说明.显然,如果这些行为场景都不能在变化后的资源环境下得到满足,那么这个构件组合系统就应当可以认为已经完全失效了.这时,我们需要重新选择构件,考虑进行其他形式的组合,并再一次进行相应的分析与验证.

4.1 UML顺序图模型及其形式化说明

构件式系统中可以使用 UML 顺序模型描述系统运行过程中可能出现的一个行为场景.它通过在多个构件之间的消息发送和接收的序列来表达在一个场景中这些构件是如何进行交互,从而达到某一个合作的目标的.图 4 所示为一个简单的顺序图模型.本文中暂不考虑带循环和选择分支的顺序图.

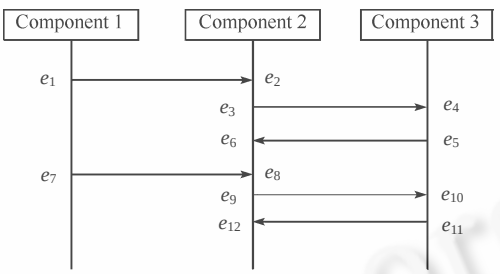


Fig.4 UML sequence diagram model

图 4 UML 顺序图模型

图中的消息发送以及消息的接收使用事件来表示.不失一般性,假定每一个顺序图对应于 1 个或多个可视序列.每个可视序列中的元素都是一个事件对(e_1,e_2);其中(e_1,e_2)表示事件 e_1 在事件 e_2 之前发生.在图中只要当出现满足下述关系时,事件 e_1 就会被视为在事件 e_2 之前发生[7]:

- 因果关系:e_1 为消息发送事件,e_2 为此消息对应的接收事件.
- 控制关系:在同一个构件的生命线轴上,事件 e_1 出现于事件 e_2 的上方,且 e_2 是消息发送事件.这个事件顺序表明一个发送事件可以等待其他事件的发生.但是一个构件只能控制自身发送消息事件的时间先后,对于接收消息事件,由于系统的底层结构和消息传送的媒介是未知的,因此接收消息的时间也是不确定的,所以对消息接收事件发生的顺序没有作严格的限定.
- FIFO 顺序关系:在同一个构件的生命线轴上,接收事件 e_1 出现于接收事件 e_2 的上方,并且相对应的发送事件 e_1' 和 e_2' 同时出现在另外一个构件的生命线轴上,且 e_1' 位置高于 e_2'.

对于顺序图中的任何消息,可能是同步的,也可能是异步的.异步的消息会消耗时间;同步的消息则在其发送和接收事件之间不会有时间延迟.本文中假定顺序图中的消息都是同步消息.我们使用这种构件接口之间消

息交互的顺序图作为构件组合系统的某个特定功能或基本功能场景的规约说明.

定义 9. 一个顺序图是五元组 $D=(C,E,M,L,V)$, 其中:

- C:有穷的构件集;
- E:有穷的事件集;
- M:有穷的消息集;对每一个消息 $m\in M$, 分别使用 $m!$和 $m?$来表示消息 m 的发送和接收事件;
- $L:E\to C$是一个标记函数,将每一个事件 $e\in E$ 分配给一个构件 $L(e)\in C$;
- V:元素形式为 (e,e') 的有穷集,其中 $e,e'\in E$,且$e\neq e'$, 表示 D 中的每一个可视序列对.

任意一个事件 $e\in E$, 都对应一个消息 $m\in M$ 的发送和接收事件,即 $\tau(e)=m!$ 或者 $\tau(e)=m?$. 顺序图刻画了系统运行的一个场景,其运行过程就表现为一个事件的序列 $e_0\hat{}e_1\hat{}...\hat{}e_m$, 其中事件 e_{i+1} 在事件 e_i 之后发生 $(1\le i\le m-1)$. 由于在控制关系中可能存在不确定的接收消息的先后次序,因此一个顺序图可能允许多个事件序列.

定义 10. 对于任意一个顺序图 $D=(C,E,M,L,V)$,事件序列 $e_0\hat{}e_1\hat{}...\hat{}e_n$ 是 D 的一个有效事件序列当且仅当以下条件满足:

- E 中所有的事件都在这个序列中出现,而且每个事件仅发生 1 次,即 $\{e_0,e_1,...,e_n\}=E$,且对于任意 $i,j(i\neq j,0\le i,j\le n)$,有 $e_i\neq e_j$.
- $e_0,e_1,...,e_n$ 满足 V 定义的可视序列,即对于任意 e_i 和 e_j,当 $(e_i,e_j)\in V$ 时,一定有 $0\le i<j\le n$.

4.2 检验构件组合系统中指定的功能行为是否满足资源约束

在以下的讨论中,我们假定对于所关心的组合系统中某个指定的功能行为而言,已经通过顺序图规约给出了其相应的事件序列集合 D,然后取集合中任意一条事件序列加以讨论即可.不失一般性,设 D 中任意一条事件序列形如: $d=f_0\hat{}f_1\hat{}...\hat{}f_u$.以下先给出 RIA-Networks 中行为与给定的事件序列之间的投影关系.

设 $com(N)$是一个兼容的 RIA-Networks 系统.系统由 n 个构件组合而成,有 K 种不同种类的资源.取 $com(N)$ 中的任意一个行为ρ,形如: $\rho=w_0\xrightarrow{a_0}w_1\xrightarrow{a_1}...\xrightarrow{a_{m-2}}w_{m-1}\xrightarrow{a_{m-1}}w_m$. 每一个资源组合状态表示为 $\overline{w}_i=(\overline{v}_i,F_N(\overline{v}_i))$, 其中每个组合状态形如: $\overline{v}_i=(v_{i_0},v_{i_1},...,v_{i_n})$. 抽取行为$\rho$中的动作序列 $a_0\hat{}a_1\hat{}...\hat{}a_{m-1}$, 替换每一个内部动作 $a_i(0\le i\le m-1)$ 为一对输入和输出动作 $a_i?\hat{}a_i!$,其余的保持不变.同样,将每一个输出(输入)动作 a 视为一个发送(接受)事件 e,可以得到一个相应的事件序列,形如: $e_0\hat{}e_1\hat{}...\hat{}e_r(r\ge m-1)$, 称其为行为$\rho$的轨迹,用$\sigma$表示.

设σ_1 是σ的一个子串,形如: $e_p\hat{}e_{p+1}\hat{}...\hat{}e_q$ $(0\le p\le r-q,0\le q\le r)$,对于上述给定的事件序列 $d=f_0\hat{}f_1\hat{}...\hat{}f_u$ 而言,如果存在 $e_{k_0},e_{k_1},...,e_{k_u}$ 同时满足以下条件,则称子串σ_1 是σ中关于事件序列 d 的一个合法投影:

- $p=k_0<k_1<...<k_u=q$;
- 对于任意 $i(0\le i\le u)$,都有 $f_i=e_{k_i}$;
- 对于任意 $i(0\le i\le u)$,任意 $j\neq k_v(p\le j\le q,0\le v\le u)$, 有 $f_i\neq e_j$.

在这种情况下, σ_1 事实上就正好表述了 d 在行为σ中的一个发生.

抽取行为σ中的资源轨迹 $\lambda_\rho=F_N(\overline{v}_0)\hat{}F_N(\overline{v}_1)\hat{}F_N(\overline{v}_2)\hat{}...\hat{}F_N(\overline{v}_m)$; 展开为分量表示形式为 $\langle f_N^1(\overline{v}_0)f_N^2(\overline{v}_0),...,f_N^K(\overline{v}_0)\rangle\hat{}\langle f_N^1(\overline{v}_1),f_N^2(\overline{v}_1),...,f_N^K(\overline{v}_1)\rangle\hat{}...\hat{}\langle f_N^1(\overline{v}_m)f_N^2(\overline{v}_m),...,f_N^K(\overline{v}_m)\rangle$. 继续针对每一类资源提取相应的分量,我们可以分别获得 K 个资源使用序列,即对任一类资源 $R_i(1\le i\le K)$, 都有一个序列: $\lambda_\rho(R_i)=f_N^i(\overline{v}_0)\hat{}f_N^i(\overline{v}_1)\hat{}...\hat{}f_N^i(\overline{v}_m)$, 其中, $f_N^i(\overline{v}_h)=\sum_{j=1}^n f_{P_j}^i(\overline{v}_{hj})$ $(0\le h\le m)$. 我们用 $\mathrm{Max}[\lambda_\rho(R_i)]$ 表示上述序列中的最大值.

在上面讨论的基础上,下面我们给出给定功能的资源使用合法性问题的形式描述.设系统资源约束为 $\vec{B}=\langle b_1,b_2,...,b_K\rangle$ (每一个 b_i 都是一个非负整数);组合系统 $com(N)$满足给定功能的事件序列 d 是指在 $com(N)$中存在一个行为σ,形如: $\rho=\overline{w}_0\xrightarrow{a_0}\overline{w}_1\xrightarrow{a_1}...\xrightarrow{a_{m-1}}\overline{w}_m$,满足以下条件:

- 子序列 $\overline{w}_j\xrightarrow{a_j}\overline{w}_{j+1}\xrightarrow{a_{j+1}}...\xrightarrow{a_{m-1}}\overline{w}_m$ 的轨迹σ_1 是关于 d 的一个合法投影;
- 对于每一类资源 R_i,都有 $\mathrm{Max}[\lambda_\rho(R_i)]\le b_i$ $(1\le i\le K)$ 成立.

由上一节中可达图 G 的构造方法可知,$com(N)$的任意一个行为对应于 G 中的一条路径.不妨设 $\rho = l_0 \hat{} l_1 \hat{} ... \hat{} l_m (l_i : s_i \xrightarrow{a_i} s_{i+1})$ 是 G 中的任意一条路径,其中 $l_i \hat{} l_{i+1} \hat{} ... \hat{} l_j$ $(0 \le i \le m-j)$ 为路径ρ的子路径.采用前面同样的方法,我们对每条边上的动作标记序列 $a_0 \hat{} a_1 \hat{} ... \hat{} a_{m-1}$ 进行替换处理,可以得到一个相应的事件序列 $e_0 \hat{} e_1 \hat{} ... \hat{} e_s$ $(s \ge m)$,称为路径ρ的轨迹 σ_ρ, 子路径 $l_i \hat{} l_{i+1} \hat{} ... \hat{} l_{i+j}$ 的轨迹我们用 $\sigma(l_i, l_j)$ 来表示,用 $\psi(l_k)$ 来表示边 l_k 上对应的事件对.显然,对于 $com(N)$ 中任意一个行为的轨迹 σ 而言,可达图 G 中一定存在一条路径,其轨迹也正好是σ.因此,对于以上定义的指定功能资源使用合法性验证问题,我们可以通过对可达图 G 进行相关性质的检查来实现.

考虑到在第 3 节中已经给出了一个验证整个组合系统对资源使用合法性的算法,算法(算法 1)的输出结果中有一个可达图 G 的异常节点集 $abnormal$.当 $abnormal$ 为空时,表明 G 就是一个规范可达图,系统中的每一个行为都满足资源约束条件 $\vec{B}$,这样,上述验证问题的结果为真.当 $abnormal$ 为非空时,表明 $com(N)$不满足其中的所有行为都符合给定的资源约束 $\vec{B}$.此时,我们将 $abnormal$ 集中的这些异常节点从可达图 G 中去掉,并且将与这些节点相关联的边也同时删除,这样所得到的一个可达图就是一个满足当前资源约束 $\vec{B}$ 的规范可达图 G' (称为修正的可达图(revised reachability graph)),然后,我们只需在 G' 中查找是否存在一条有穷路径,它满足其某一条子路径的轨迹是关于给定功能行为的事件序列 d 的合法投影.如果存在这样的路径,就意味着在当前的资源约束 $\vec{B}$ 下,组合系统 $com(N)$即使存在某些资源使用异常的状态,我们所关心的那个功能行为也可以正常实现.比如,在上一节中,我们已经判断出当给定的资源约束为 $\vec{B} = \langle 5,8 \rangle$ i 时,组合系统 $com(A \otimes B \otimes C)$ 的可达图中存在使用资源异常的节点 s_6,其可达图也不是一个规范的可达图.现在,我们希望在当前环境资源可用性发生变化时,组合系统中仍然有功能序列 $h \to a \to b \to c \to f \to e$ 能够保留下来,可以正常工作.如图 5 所示,我们将图 3 中的异常节点 s_6 和相关的边去掉.图 5 中虚线表示去掉的边,异常节点 s_6 也用虚线框标示出来了.由于剩下的节点都满足当前的资源约束,因此,得到的是一个修正的规范可达图;并且我们发现,在此图中,我们所希望保留的功能序列 $h \to a \to b \to c \to f \to e$ 确实被保留下来了,这就意味着在当前的资源约束条件下,组合系统仍然可以完成这个特定的功能.

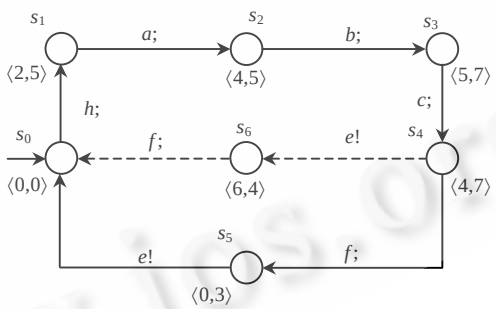


Fig.5 A revised reachability graph G'

图 5 修正的可达图 G'

一般来说,由于在 G' 中可能存在环路,因此我们采用文献[8]中类似的处理方法,首先引入 G' 中关于 d 的投影路径的概念.给定事件序列 d, G' 中的任一条路径 $\rho = l_0 \hat{} l_1 \hat{} ... \hat{} l_m$,若其同时满足以下条件,则称 ρ 为关于 d 的投影路径:

- 存在一个子路径的轨迹 $\sigma(l_i, l_m)(0 \le i \le m)$ 是 $\sigma(l_0, l_m)$ 关于 d 的一个合法投影;
- 对于任意 $j,k(0 \le j < k < i)$, ρ 中有 $l_j \ne l_k$;
- 对任何相邻的 $j,k\,(i \le j < k \le m, \psi(l_j) \in d, \psi(l_k) \in d)$, ρ 中有 $l_g \ne l_h\,(j < g < h < k)$.

第 1 个条件表示路径 ρ 的末端部分恰好包含 d 的事件序列;第 2 个和第 3 个条件则对路径 ρ 中前后两段中可能出现的环路作了相应的限制.显然, G' 中的每一条投影路径都是有穷的, G' 中投影路径的数量也是有穷的.

基于以上讨论,我们就可以给出以下验证定理:

定理 2. 在资源约束 $\vec{B}$ 下,$com(N)$仍然满足给定功能的事件序列 d 当且仅当其修正的可达图 G' 中存在一条关于 d 的投影路径.

定理证明见附录.基于上述定理,我们可以构造一个基于深度优先搜索的验证算法,见算法 3.算法的输入是修正的可达图 G' 和给定功能行为的事件序列 d,算法的输出就是检验结果满足还是不满足.算法的复杂度与 G' 中关于 d 的投影路径的长度和数量成正比.

算法 3. 检验指定功能的资源使用合法性算法.

$current\_path := \{s_0\}$; $satisfied :=$ false;

repeat

node:=取 *current_path* 中最后一个节点;

if *node* 的后继节点均已访问过 **then** 删除 *current_path* 中的最后一个节点

else begin *node*:=取 *node* 的一个未访问过的后继节点;

if 路径 *current_path*^*node* 是关于 *d* 的投影路径

then satisfied:=true;

if 路径 *current_path*^*node* 是一条投影路径的前缀

then *node* 加入到 *current_path* 中;

end

until *current_path*={};

if *satisfied* **then return** true **else return** false.

5 相关研究工作

目前,在基于软件构件技术的工程实践领域,存在一些有代表性的途径与方法,如 OMG 组织的 CORBA/CCM[9],Microsoft 公司给出的 COM/DCOM/COM+[10],Sun 公司的 J2EE/EJB[11]等.这些技术在应用服务器以及复合文档处理方面应用较为成功;主要是在语法层面上对接口数据类型的匹配与检查以及接口使用协议的匹配与检查,还没有专门针对使用构件的资源约束方面的描述和检查机制.其原因可能主要是对于上述的应用领域而言,构件的运行框架一般都默认提供了足够的运行资源,并且环境资源的变化对于这些应用的影响后果并不严重;因此,在这些应用领域中对软件构件的使用基本上可以不考虑资源的问题.但是,目前的这些技术还不足以支撑网构软件在自适应性质方面的能力.

与系统资源使用特征建模、分析与验证相关的研究工作主要是与嵌入式系统相关.由于在嵌入式系统中各类系统资源严格受限,因此必须在嵌入式软件的设计中考虑与资源使用相关的非功能方面的性质.这方面的工作较多,包括:根据应用背景设计特定的框架平台来支持资源的分析和检查,如文献[12,13];对系统特定资源的使用与系统性能的影响进行预测和优化等,如文献[14,15];设计相关的形式化或半形式化的非功能性建模语言来描述资源特征,如文献[16,17]等.其中,文献[17]给出了一种进程代数的形式系统,以描述和管理在并发系统中的资源使用.文献[16]则是在接口自动机的基础上扩充了资源约束条件,以对构件化嵌入式软件系统的资源和能耗进行分析验证;由于其资源接口(resource interface)只是简单地添加了整型数来表达资源量,模型的表达能力还比较有限.这些相关工作的共同特点是在嵌入式系统这个相对封闭的计算环境中展开资源与非功能性质方面的研究,而不是基于 Internet 这个开放、动态的平台.

在网构软件的自适应性方面的研究包括:通过构建构件实体之间的协同模式以及演化的方式来动态地适应外部环境变化[18,19],这里所指的环境变化是广义上的,包含了资源变化与需求变化的语义.文献[18]是这方面的代表工作,提出了一种以服务实体与协同部分分离、基于第三方服务实体的协同聚合、协同模式的设计与演化适应环境变化为特征的开放协同软件模型作为网构软件的基础模型.使用智能 Agent,从软件实体协同方式的动态调整角度来构建网构软件的自适应性质和自演化功能.另外一类重要工作是基于软件体系结构的网构软件自适应研究.其中,文献[19]是这方面的代表工作,设计了一个基于构件的软件体系结构开发框架 ABC,用于

网构软件的设计、开发、部署与运行等全部的生命周期.在这个基于构件的软件体系结构开发框架下,可以通过对构件实体的主体化,使用规则集加规划集的方法来支持网构软件在运行过程中构件的动态调整.但是总的来看,目前对于网构系统在自适应、自演化过程中出现的实体行为、协同模式等方面的不一致性问题,暂时还未形成系统的形式化分析和验证方法.

与以上的研究工作相比,本文的工作主要是在基于构件的网构系统模型层次,通过对构件接口组合行为的形式化分析来提高对网构软件的环境自适应性的可信度.这样,在网构软件部署和运行时,当环境资源发生变化时,可以根据系统模型分析和验证的结果对网构软件的行为作出合理和可信的预期.此外,随着多核硬件平台的广泛应用,这种模型验证的方法还可以在运行时对系统在线演化的行为进行形式化分析与验证,从而进一步提高系统在自适应性方面的可信度.比如,在构件组合体中的某一个构件进行了自主的更新或被动的替换之后,可能相应的接口信息会发生变化或者系统资源的变化超出了限定范围,进而会影响到整个组合体的功能时,就必须进行重新分析与验证.

6 结束语和进一步工作

本文的工作在网构软件的系统模型层次,使用带资源语义信息的接口自动机对软件构件的行为进行形式化建模;使用资源接口自动机网络来描述构件组装实体的组合行为;使用基于场景的 UML 顺序图模型来描述具有多功能的组合系统规约;针对检验组合系统的所有行为是否都满足给定的资源约束和检验指定的系统行为是否满足资源约束这两个具体的问题展开研究;通过对资源自动机网络状态空间的分析,构造其相应的可达图,并在此基础上给出了相应的检验资源可满足性、最小资源需求量以及检验指定功能合法性等算法.

本文的工作主要关注于网构软件在环境资源自适应性质方面可信方法的理论研究,目前正在进行的工作包括实现文中所给出的算法,构造相应的分析与验证原型工具,并对典型的网构软件系统进行实例研究.另外,本文中所考虑的资源使用的特征是有一定的假设前提的,并且给出的数值向量形式的环境资源约束较为简单,这方面的进一步工作将包括对 Internet 环境下的资源的变化特征进行分析,建立更加合理的资源模型.比如,由于 Internet 上的资源变化是不确定、随机的,因此,考虑带有概率信息的资源语义模型可能更加适合于网构软件自适应性质的分析与验证.最后,本文所研究的自适应性特征主要是考虑软件构件组合系统在环境资源变化时所展现出来的一种被动的、可调节的柔性特征,其中,基于场景的特定构件组合行为的规约是预先设定的.进一步的工作应当包括研究软件需求变化所带来的自适应性行为,这表现的是系统一种主动的适应和调节能力,可以根据运行阶段软件系统需求的变化来实时、动态地构造合适的构件行为交互场景规约,并进一步根据这些场景来分析和验证构件组合系统的自适应性质,增加我们对构件组合系统的自适应能力的可信度.

References:

[1] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[2] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and progress on Internetware. Science in China (Series E), 2006,36(10):1037−1080 (in Chinese with English abstract).

[3] Szyperski C. Component Software: Beyond Object-Oriented Programming. 2nd ed., Addison Wesley Professional, 2002.

[4] de Alfaro L, Henzinger TA. Interface automata. In: Proc. of the Joint 8th European Software Engineering Conf. and 9th ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (ESEC/FSE 2001). ACM Press, 2001. 109−120.

[5] de Alfaro L, Henzinger TA. Interface theories for component-based design. In: Henzinger TA, Kirsch CM, eds. Proc. of the EMSOFT 2001. LNCS 2211, Springer-Verlag, 2001. 148−165.

[6] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 1999.

[7] Peled DA. Software Reliability Methods. Springer-Verlag, 2001.

[8] Li XD, Hu J, Bu L, Zhao JH, Zheng GL. Consistency checking of concurrent models for scenario-based specifications. In: Prinz A, Reed R, Reed J, eds. Proc. of the 12th Int'l SDL Forum (SDL2005). LNCS 3530, 2005. 298−312.

[9] OMG. Common object request broker architecture: Core specification. Version 3.0, 2004. http://www.omg.org/technology/documents/corba_spec_catalog.htm

[10] Eddon G, Eddon H. Inside COM+ Base Services. Redmond: Microsoft Press, 2000.

[11] Burke B, Monson-Haefel R. Enterprise Javabeans 3.0. O'Reilly, 2006.

[12] Burmester S, Gehrke M, Giese H, OberthÄur S. Making mechatronic agents resource-aware in order to enable safe dynamic resource allocation. In: Buttazzo GC, ed. Proc. of the 4th ACM Int'l Conf. on Embedded Software, EMSOFT 2004. ACM, 2004. 175−183.

[13] Mikic-Rakic M, Medvidovic N. Architecture-Level support for software component deployment in resource constrained environments. In: Bishop JM, ed. Component Deployment, IFIP/ACM Working Conf., CD 2002. LNCS 2370, Springer-Verlag, 2002. 31−50.

[14] Fredriksson J, SandstrÄom K, Akerholm M. Optimizing resource usage in component based real-time systems. In: Heineman GT, Crnkovic I, Schmidt HW, Stafford JA, Szyperski CA, Wallnau KC, eds. Component- Based Software Engineering, the 8th Int'l Symp., CBSE 2005. LNCS 3489, Springer-Verlag, 2005. 49−65.

[15] Fredriksson J, Tivoli M, Crnkovic I. A component-based development framework for supporting functional and non-functional analysis in control system design. In: Redmiles DF, Ellman T, Zisman A, eds. Proc. of the 20th IEEE/ACM Int'l Conf. of Aumated Software Engineering. ACM Press, 2005. 368−371.

[16] Chakrabarti A, de Alfaro L, Henzinger TA, Stoelinga M. Resource interfaces. In: Alur R, Lee I, eds. Embedded Software, the 3rd Int'l Conf., EMSOFT 2003. LNCS 2855, Springer-Verlag, 2003. 117−133.

[17] Mousavi M, Reniers M, Basten T, Chaudron M. PARS: A process algebra with resources and schedules. In: Larsen KG, Niebert P, eds. Proc. of the Int'l Conf. on Formal Modeling and Analysis of Timed Systems. LNCS 2791, Springer-Verlag, 2004. 134−150.

[18] Lü J, Tao XP, Ma XX, Hu H, Xu F, Cao C. A study of Agent-based Internetware model. Science in China (Series E), 2005,35(12):1233−1253 (in Chinese with English abstract).

[19] Mei H, Huang G, Zhao HY, Jiao WP. A software architecture centric engineering approach for Internetware. Science in China (Series E), 2006,36(10):1100−1126 (in Chinese with English abstract).

附中文参考文献:

[1] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[2] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑),2006,36(10):1037−1080.

[18] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究.中国科学(E 辑),2005,35(12):1233−1253.

[19] 梅宏,黄罡,赵海燕,焦文品.一种以软件体系结构为中心的网构软件开发方法.中国科学(E 辑),2006,36(10):1100−1126.

附录. 定理证明.

定理 1. N 是一个 RIA-Networks,其满足某个资源使用约束 $\vec{B}$ 当且仅当其兼容状态空间的可达图 G 是一个规范的可达图.

证明:充分条件. 设 N 是一个 RIA-Networks,其满足某个资源使用约束 $\vec{B}$ 意味着对于给定的约束 $\vec{B}$,系统 N 在运行过程中,其任何行为中的任意一个组合状态 $\overline{w}_i$ 都没有违反资源使用的限制,即均不满足定义 8.由于可兼容的系统状态空间 $com(N)$是排除掉非法交互状态之后的 N 的一个可达状态子集,因此,基于 $com(N)$所构造的可达图 G 中一定不会出现异常节点,即 G 是一个规范的可达图.

必要条件.设 N 是 RIA-Networks,当可兼容状态空间 $com(N)$的可达图 G 是一个规范的可达图时,意味着在 $com(N)$中不存在违反资源约束 $\vec{B}$ 的组合状态.但是,由接口自动机的定义和计算兼容状态空间的逆向可达性算法可知,在原系统 N 中,仍然可能存在某些不兼容的组合状态 $\overline{w}_j$ 或者不可达的组合状态 $\overline{w}_k$,这些状态上的资源使用有可能超出给定的约束 $\vec{B}$.当出现第 1 种情况时,由于 N 总是在某个合法环境 R 下运行,相对于环境 R 而言,状态 $\overline{w}_j$ 一定是不可达的.因为若相对于某个合法环境 R' 而言,$\overline{w}_j$ 成为了可达状态,则意味着在所构造出的可达图 G 中一定存在某个异常节点,G 也一定不是规范的可达图,这与假设前提矛盾.当出现第 2 种情况时,意味着无论 N 在何种合法环境下运行,$\overline{w}_k$ 永远是不可达的.因此,由上可知,在某个合法环境下,N 的所有可能的行为

一定是满足资源约束 $\vec{B}$ 的. □

定理 2. 在资源约束 $\vec{B}$ 下,$com(N)$仍然满足给定功能的事件序列 d 当且仅当其修正的可达图 G' 中存在一条关于 d 的投影路径.

证明:充分条件.当 $com(N)$满足 d 时,由可达图 G 的构造方法可知,G 中存在一条路径 $\rho=l_0\hat{}l_1\hat{}...\hat{}l_j\hat{}...\hat{}l_k\hat{}...\hat{}l_n$, 满足: $\sigma(l_j,l_k)$ 是关于 d 的合法投影.取路径 $\rho=l_0\hat{}l_1\hat{}...\hat{}l_j\hat{}...\hat{}l_k$ 考虑,若子路径 $l_0\hat{}l_1\hat{}...\hat{}l_{j-1}$ 中存在片段 $l_g\hat{}l_{g+1}\hat{}...\hat{}l_h(0\le g,h\le j-1)$ 满足 $l_g=l_n$,则在原路径中用 l_g 替换掉片段 $l_g\hat{}l_{g+1}\hat{}...\hat{}l_h$. 反复进行此环路替换操作,直至 $l_0\hat{}l_1\hat{}...\hat{}l_{j-1}$ 中每一个 $l_k(0\le k\le j-1)$ 都不相同.然后,考虑子路径 $l_j\hat{}l_{j+1}\hat{}...\hat{}l_k$, 若其中存在片段 $l_u\hat{}l_{u+1}\hat{}...\hat{}l_v$ $(j<u,v<k)$, 满足:(1) 不存在一个 $\psi(l_i)\in E(u\le i\le v)$ (E 是 d 中的事件集);(2) $l_u=l_v$, 则用 l_u 替换掉原来路径中的片段 $l_u\hat{}l_{u+1}\hat{}...\hat{}l_v$.因为 $\sigma(l_j,l_k)$ 是关于 D 的合法投影,由上述构造方法可知,经过替换操作所得到的路径 ρ' 是一个满足 D 的投影路径.

必要条件. 由可达图 G 的构造和投影路径的定义可直接证明. □

胡军(1973－),男,湖北黄冈人,博士,讲师,CCF 会员,主要研究领域为软件工程,形式化方法,软件验证.

黄志球(1965－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,可信计算,数据库工程.

曹东(1972－),男,博士,助理研究员,主要研究领域为航空控制工程,软件工程,嵌入式软件.

徐丙凤(1986－),女,硕士,CCF 学生会员,主要研究领域为软件工程,基于模型的测试.