移动组件系统模型的分析与描述*

魏峻，周桓

(中国科学院 软件研究所 计算机科学开放研究实验室，北京 100080)；
(中国科学院 软件研究所 对象技术中心，北京 100080)
E-mail：wj@ios.ac.cn；quan@otcaix.iscas.ac.cn

摘要：移动计算是新兴的分布式计算范型，其主要特征是计算组件与计算场所能动态改变绑定关系，表现出移动性，从而带来许多新的系统设计需求.从系统模型层次角度对位置、移动组件、移动和资源访问等移动计算核心概念，以及组件与位置之间的各种关系和关系变化刻画的移动范型进行了描述.通过使用集合论和操作语义的规约规则形式地表示这些概念、关系和移动机制，进而抽象出移动系统设计所需的语言结构，为移动系统设计和开发提供了分析基础.

关键词：移动组件系统模型；形式方法；移动性；位置；组件；组件关系
中图法分类号：TP311 **文献标识码**：A

现代网络技术，尤其是无线通信技术的发展，以及各种移动计算设备的不断出现，给人类带来了广泛而深入的信息访问，同时也刺激了软件系统的发展和转型.目前，移动计算普遍被认为是由软、硬件新技术支持的一种新计算范型，其主要特征是软、硬组件能与计算场所动态改变绑定，表现出移动性.正是由于其移动性带动了各种新兴网络应用的蓬勃发展.

目前，人们对移动还缺乏较完整和深入的认识，移动性带来的需求期待在新的分布式系统设计中得到满足.移动实体的自治与异质性使系统设计很难具备实现所需求的特征，如数据、代码、agent、操作环境等多种实体都可以移动；系统设计不得不面对如何描述移动实体，如何说明它们何时、怎样移动，以及移动带来的系统重构的问题.这些因素使得移动系统呈现出多样性与纷乱性.一些研究工作表明，从抽象层次运用形式方法能较好地认识移动系统的本质与特征，如 Ambient[1]，Joint[2]，Klaim[3]，Seal[4]演算和 Mobile Unity[5]等.它们有的是刻画 C/S 体系结构下的数据移动，有的是刻画如 Java applets 的代码即需即取(code on demand)式的代码移动，还有的是描述智能计算实体整体移动的 agent 移动，甚至可以刻画包含整个操作环境的硬件平台移动的一类系统[6].这些语言和演算为设计移动系统提供了规范说明和验证方法，但是由于数学基础、技术目标等因素的不同，从而各自勾画了不同的移动系统模型.

本文旨在从模型层次角度来描述移动计算核心概念和各种移动机制，分析移动系统设计所需的语言抽象结构，为移动系统的设计和开发提供分析基础.我们所讨论的实体是指能动态传送的代码、数据、执行状态、它们的闭包，甚至整个操作环境等多种抽象组件.

1 移动计算的特征与研究现状

移动计算关注两类计算活动，一类是在基于无线网络的移动设备(如膝上机、PDA 等)上进行

* 收稿日期：1999-06-28；修改日期：1999-11-23
基金项目：国家自然科学基金重点资助项目(69833030)；国家重点基础研究发展规划资助项目(G1998030404)
作者简介：魏峻(1970—)，男，湖北钟祥人，博士，主要研究领域为分布式对象计算，移动 agent 计算，软件形式化方法；周桓(1976—)，男，安徽芜湖人，博士生，主要研究领域为移动计算，分步计算系统.

的计算活动及相关问题;另一类是基于WEB的移动程序的计算,如Java applet,agent.多项研究表明,在设计层次可以用一致的视图抽象出这两类分布式计算[5,7,8].

1.1 移动计算特征

支持移动计算的基础设施,如WWW、无线通信网络、各种移动设备和Java一类程序设计语言等的特点,决定了移动计算范型具有不同于传统分布式计算的特征,主要表现为:

(1) 通信服务质量不稳定.WEB地理位置全球分布的自然特点以及经常出现的不可预料的通信拥塞,导致带宽波动或连接中断.而且,无线网络通信质量差,带宽低,频繁断连且等待响应时间长,移动设备的移动还会引起带宽波动.

(2) 位置因素不容忽视.网络通信质量差需要计算组件(软件或硬件)移动作为克服或解决网络带宽波动、连接不稳定、等待时间长等问题的有效方法.同时,网络区域化管理界定出逻辑上的边界范围,形成层次化的管理空间.不同粒度的管理域形成物理或逻辑上的位置,在其中(间)进行的计算无法忽视位置因素.

(3) 系统配置动态变化.移动计算的移动性造成移动组件系统配置的动态变化.全球开放的网络计算没有全局管理,通信链接和新计算实体的加入不会事先通知,计算实体还可以移动或消失.无论是在物理意义还是逻辑意义上,系统服务、可用资源和拓扑结构都在随时间发生变化.

(4) 系统组件松散而又机会性地组合.移动计算的物理约束决定了系统组件的自治与封装,以及组件相互之间的松耦合.即在系统崩溃、失败或有意断连等极端情况下仍要独自运作,但在可能重建连接时,又能即时动态绑定.

(5) 资源访问控制灵活而多样.网络的开放和移动系统配置的动态变化决定了资源访问策略需要灵活且形式多样,必须能够根据特定的应用领域动态地加以定制.

1.2 移动计算研究状况

许多移动计算的研究集中在移动硬件、移动和无线网络协议、移动数据访问等方面,还有一类是研究移动系统集成平台技术,其主要途径是利用与扩充传统分布式计算基础来使移动性透明,如改进分布式操作系统(文件系统),实现移动数据访问的Coda和Odyssey[9,10];改进程序设计语言,支持代码、对象和agent移动的语言系统[11~13];基于分布对象程序设计平台CORBA和DCOM,透明通信以支持代码与数据移动的ObjectSpace等.这些研究分别只针对了移动计算的某些特征.移动硬件针对的是设备的计算能力和通信接入能力;移动和无线网络协议解决通信中的广域、透明(IP)寻址;移动数据访问针对的是数据资源的访问与管理;集成平台实际上只注重数据与代码移动等形式,以及移动组件(设备)的位置管理与跟踪等方面.它们都忽视了从整体模型来认识移动计算系统,因而未能充分认识移动计算的核心概念和机制,发现移动系统设计所面临的基础问题.

但是,传统的刻画并发与分布式计算的形式方法在体现移动计算特性时又显得不够充分.如分布式协调计算模型LINDA[14]抽象了分布与通信,不适合描述实现;π演算[15]等进程代数可描述通信通道的移动,但缺乏位置概念,因而不能讨论分布,并且缺少表示agent自治性的标识概念;Obliq[12]的语义有分布式作用域的概念,它可以作为讨论分布式数据移动的框架,但却又不能表示像Telescript[11]那样的很多的移动机制;同样,用Telescript的Place-Agent模型描述Obilq的远程引用又非常不便利.

许多新的形式语言和演算被提出来用于描述和分析移动系统,如基于进程代数的Ambient,Joint,Klaim,Seal等演算,它们在处理移动与系统动态重配置方面显得从容自如.基于时序逻辑的Mobile Unity语言通过变量共享机制可以充分表现设备移动、操作断连等移动计算特征,而且使用

它可以很容易地进行从规范到实现的逐步求精以及系统性质验证.这些形式语言和演算赋予了移动计算核心概念和机制不同的语义和语言表现,表现出各具特色的移动系统模型.运用模型分析和形式刻画的方法,不仅能够更深刻地理解移动系统设计的基础问题,而且能在解释移动系统设计语言构造的精确语义方面起到重要作用,从而发现新的语言成分.

2 移动组件系统模型

移动系统涵盖了软件、硬件组件及相关的广泛且纵深的网络计算环境,移动计算的强位置依赖及其移动性需要移动计算系统的模型以层次的、松散的风格加以组织,无论是在具体的还是抽象的模型中,位置、移动组件、移动和资源访问都需要充分给予考虑.

2.1 系统模型概览

移动系统被看做是由多个站点组成的计算系统,每个站点(物理或逻辑上)可以包含多个计算位置,它们是组件的计算环境,同时也是各种资源的管理与控制中心,组件之间的交互协作可以是远程通信方式,也可以是组件移动到某位置进行局部交互,组件可以访问本地和远程资源.并且,计算环境(位置)也可以移动,从而带动其子位置和组件一起移动.

该模型是对广泛的移动计算系统模型的抽象,可对应于树型图1.图2是更直观的等价表示.若将图1中的Site对应于移动计算设备,将place对应于计算机上运行的解释执行环境,其中组件对应于Java或Telescript程序,net对应于由计算设备通过固定或无线网络连接成的计算机网络,那么就可以将该移动抽象模型很好地对应于实际运作的移动计算系统.

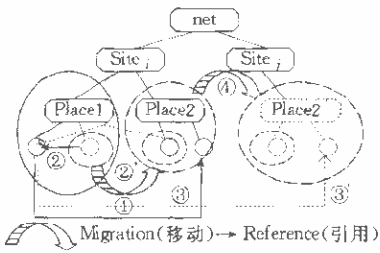

Fig. 1
图1

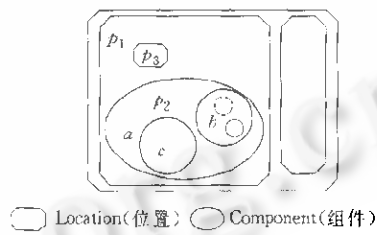

Fig. 2
图2

可以看出,该抽象模型实际上是关于组件、位置以及它们之间的关系及关系变化的模型.位置、组件、移动性、资源访问是移动系统的关键概念,其中,位置与组件是核心概念.

移动组件包括软组件,如软件agent,Java beans,applets,进程等,也包括硬组件,如Laptop,PDA,PalmPC,手机等移动计算设备,甚至包括移动基站.而位置是各种容纳组件(包含资源)的计算场所,更多地是指(物理或逻辑上的)空间,它可以是计算机或网络的物理位置,也可以是逻辑或概念上如逻辑网络节点、内存空间、进程空间、虚拟机等.其他概念是组件和位置概念或关系的衍生.移动是组件与位置各自的、相互之间绑定关系的变动;资源访问控制由位置界定出访问范围,由组件身份标识访问对象;系统配置的变化是由移动、组件创建与消亡等因素引起的.

因此,我们认为,在移动系统模型中必须清楚地区分组件与位置这两个概念,并且标识出它们之间的各种关系和可能的关系变化.这样就能刻画出移动计算的系统抽象模型.移动系统中的其他概念,如组件交互、安全控制等在移动系统设计与实现时非常重要,但在系统模型层次,它们不是认识和刻画移动系统本质特征的关键元素.

我们还注意到位置与组件的一种特殊关系——位置必须以某个组件作为其载体,例如,内存空

间存在于计算机设备中.位置这种特指空间的概念不能凭空存在,它需要与具体的实体相联系.这种将现实与计算世界综合的观点,能更清楚地确定位置、组件的关系.

下面,我们列举了移动系统中组件、位置各自的以及相互之间的各种关系.

(1) 组件之间的关系包括组合与引用关系.

组合关系说明,复杂组件由简单的组件复合而成.如图 2 中的复合组件 b 与子组件之间的关系.这种关系是紧耦合的,本文不考虑此类关系的动态变化(这是组件系统的动态扩充和自适应问题).

引用关系说明的是组件之间的引用关系,它是弱耦合的.表现最多的是计算组件引用数据资源,如图 1 中的②和③的引用关系.移动会引起引用关系的变化.如图 1 中②→②′和③→③′引用的变化.

(2) 位置之间的关系包括同辈和父子关系.

父子关系是指一个空间包含了另一个空间.如图 1 中的位置 Place1,Place2 与 Site; 之间的关系.同辈关系是指具有共同父空间的位置之间的关系.如图 1 中的位置 Place1 与 Place2.我们定义的系统模型不允许有相交的位置.

位置(计算环境)移动和具有寄宿关系的组件的移动,都会引起这两种位置关系的变化.

(3) 组件与位置的关系有包容与寄宿关系.

包容关系是移动系统中最重要的关系,现有的系统模型都有这种关系,它主要是从 Telescript 的 Agent-Place 模型[11]衍生而来.位置移动会带动所包容组件的移动.而且,各种移动都能改变包容关系,如图 1 所示.

寄宿关系表示给位置分配载体或确定位置的载体标识.寄宿关系一般较为稳定,其变化主要是由于组件的物理变动(这不应在计算活动中刻画)造成的.图 1 中左部表示的即是位置 Place1 寄宿于某组件.正是由于寄宿关系绑定,因此那些计算环境(位置)的移动则一般由载体组件的移动表现出来.

2.2 移动系统中的移动范型

移动系统中的移动本质上是系统配置(实体间的结构关系)随时间序列的演变.模型中可移动的实体有代码、数据、agent(包含计算代码、数据状态和/或执行状态)和计算环境,这些实体与位置之间的关系和可能的关系变化表现出多种移动范型.

(1) 数据移动是数据组件与位置包容关系的变化.若把图 1 中叶节点看做是数据组件,那么①的移动就代表数据移动.由于各种移动不可避免地包含数据传递,因此,数据移动是最基本的一种移动.数据移动包括多种形式(复制、驻留、携带、取走),下面的形式模型将给出各种数据移动的精确描述.

(2) 引用移动表示实体引用关系的变化.如图 1 中②→②′和③→③′的移动显示了两种引用关系的变化,它们分别是由组件移动和位置移动引起的.

(3) 代码移动是指可执行代码从某计算环境移动到另一个环境,它是代码组件与位置包容关系的一种变化.若将图 1 中叶节点看做是计算组件-代码,那么①的移动就代表代码移动.尽管代码可看做是数据,但代码是指加载进执行器的程序,代码移动不仅在语义上不同于数据移动,在实现中也有很大的差别.Java applet 下载就是代码移动,在实现中它使用了数据移动技术.

(4) agent 弱移动是指代码与相关数据状态一起移动,它可引起组件与位置包容关系的变化.如将图 1 中①的移动中的组件看做是封装了代码和数据状态的 agent,则①的移动即代表了 agent

弱移动.典型的例子是各种基于 Java 的移动 agent 系统中 agent 的迁移.

(5) agent 强移动是指在 agent 弱移动的基础上,加上 agent 的执行状态一起移动.它能实现 agent程序执行不间断且透明地多环境迁移.它也能引起组件与位置包容关系的变化.Telescript 实现了 agent 强移动.

(6) 闭包移动(closure mobility)是指 agent 弱移动加上资源引用移动,它能引起组件与位置包容关系的变化以及组件引用关系的变动.图 1 中①的移动加上②→②′的移动构成闭包移动.它是 Obliq 语言中的典型移动类型.

(7) 计算环境移动是指包含计算组件和资源组件的计算环境的移动.计算环境的移动会带来所包含的子环境和组件的一起移动.若其他位置的组件引用了其中的资源,则计算环境的移动不仅会影响父子位置关系,而且会影响组件的引用关系.图 1 中④的移动表示了计算环境的移动.这类移动的代表:若将嵌入 Java applet 或 activeX 控件的 HTML 页面看做计算环境,则 HTML 页面的下载代表了"软"计算环境移动;将在无线网络环境进行计算活动的便携机移动到另一个子网,则是"硬"计算环境移动.

3 移动组件系统的形式模型

下面,我们使用集合论来定义移动系统中的基本成分和它们之间的各种关系,使用操作性的规约规则刻画关系变化表现的各种移动范型,以这种抽象但精确的方式来描述移动组件系统模型.

这里描述的实体有数据、代码、agent、位置和系统,实体定义在一个名字空间 N 中,c 统一代表组件,p 代表位置,$d(D)$代表数据组件,ag 代表 agent,$v(V)$代表数据值.

定义 1. 移动组件 c_i(包括数据、代码和 agent)包含下面基本成分:名字 $c_i.\text{name}=c$,所在位置 $c_i.p$,标识 $c_i.id$,类型 $c_i.t\in\{\text{DATA},\text{CODE},\text{AGENT}\}$和命令请求域 $c_i.\text{comd}$.

定义 2. 若 $c_i.t=\text{DATA}$,则移动数据组件还包含相关值 $c_i.\text{valuc}$;请求域 $c_i.\text{comd}\in\{\text{COPY},\text{CARRY},\text{RESIDENT},\text{TAKEAWAY}\}$.

定义 3. 若 $c_i.t=\text{CODE}$,则移动代码组件 $c_i.\text{comd}\in\{\text{MOVETO}\}$.

定义 4. 若 $c_i.t=\text{AGENT}$,则移动 agent 组件包含状态 $c_i.\omega\in\{\text{ACTIVE},\text{INACTIVE}\}$;子组件列表 $c_i.\text{comp}$;引用资源列表 $c_i.\text{reflist}$;命令请求域 $c_i.\text{comd}\in\{\text{MOVETO},\text{CLONE},\text{UPDATE},\text{REF},\text{DEREF},\text{SEND},\text{RECEIVE},\ldots\}$.

定义 5. 位置 p_i 抽象包含下面基本成分:名字 $p_i.\text{name}=p_i$,标识 $p_i.id$,寄宿组件标识 $p_i.\text{lodgeComp}$,子位置列表 $p_i.\text{childlist}$,包容组件列表 $p_i.\text{complist}$.

定义 6. 设组件集合为 $CompID$,位置集合为 $LocID$,则组件、位置各自之间以及相互之间的关系为:

(1) 组件组合关系由映射 subcomp:$CompID\alpha 2^{CompID}$表示.

(2) 组件引用关系由映射 refcomp:$CompID\alpha 2^{CompID}$表示.

(3) 位置的父子关系由映射 subloc:$LocID\alpha 2^{LocID}$表示.

(4) 位置与组件的包容关系由映射 contain:$LocID\alpha 2^{CompID}$表示.

(5) 位置与组件的寄宿关系由映射 lodge:$CompID\alpha 2^{LocID}$表示.

定理. 在移动系统模型 Mod 中,若位置 $p_2\in\text{subloc}(p_1)$,组件 $b\in\text{subcomp}(a)$,且组件 $a\in\text{contain}(p_1)$和 $b_1\in\text{contain}(p_2)$,则位置 $p_2\in\text{lodge}(a)$.

证明:由定义 6 和位置不相交规定易得.其直接表示如图 2 所示.

由这些关系映射及定理可以得到,定义4中移动agent组件的$c_i.\mathrm{comp}=\mathrm{subcomp}(c_i.id)$,$c_i.\mathrm{reflist}=\mathrm{refcomp}(c_i.id)$以及定义5中位置抽象的$p_i.\mathrm{childlist}=\mathrm{subloc}(p_i.id)$,$p_i.\mathrm{complist}=\mathrm{contain}(p_i.id)$,$p_i.\mathrm{lodgeComp}=\mathrm{lodge}^{-1}(p_i.id)$.

移动系统是由分散在多个计算环境中的自治计算组件通过交互、移动进行协作和资源访问而构成的分布式系统.若将计算组件看做是独立运行体,则移动系统表示为分散在各个计算位置上的组件的并行松耦合:

$$\mathrm{MobiSys}\overset{\mathrm{def}}{=}\|_{i,k}p_k\uparrow ag_i,$$

其中$p_k\uparrow ag_i$表示在位置p_k的计算实体ag_i.上式也可以写为$\mathrm{MobiSys}\overset{\mathrm{def}}{=}\|_k p_k\uparrow(\|_i ag_i)$.

计算组件agent的行为由其操作表示,在这里定义为各种计算原语,其种类见定义4中的命令请求域.它们分别代表agent的一段计算$ag@(\mathrm{primitive},\mathrm{param})$.在计算过程中,这些原语的执行会使agent演化为agent系统(通过创建、克隆、迁移等),且分散到各个站点.因此,关于原语的操作语义,不应简单地描述为源码化简,而应表示为agent演化系统可能状态之间的规约.

定义7. 设γ为计算组件所在位置的计算上下文.其中包括与组件相关的信息和局部资源引用情况,表示为有限映射集合$\{a\alpha v,\ldots\}$.γ与组件计算体ag构成的偶对(γ,ag),称为agent配置.在位置p的agent配置表示为$(p\uparrow\gamma,ag)$.

Agent计算原语的语义表示为配置规约$(\gamma,ag@(\mathrm{primitive},\mathrm{param}))\longrightarrow(\gamma',ag')$,$\gamma'=\Phi(\gamma)$,$ag'=\Psi(ag)$,$\Phi,\Psi$分别是对计算上下文和agent体的作用.对于不同的agent原语,Φ,Ψ各不相同.这种简单的规约规则适用于agent中只涉及局部位置的原语.而涉及多个计算位置的原语需要用较全局的配置变化来刻画,随后我们将给出这种原语.

首先给出关于局部位置原语的规约规则.它们分别是:(UPDATE,a,v)更新地址a的值为v;(REF,d.value)为数据资源值d.value分配新的地址建立引用;(DEREF,a)除去对地址a的引用;(SEND,ag,comd)向agent ag发送消息comd;(RECEIVE,comd)接收消息,将消息放到ag.comd.

$(p_k\uparrow\gamma,ag@(\mathrm{UPDATE},a,v))\rightarrow(p_k\uparrow\gamma[a\alpha v],ag)$,若$a\alpha v'$,

$(p_k\uparrow\gamma,ag@(\mathrm{REF},d.\mathrm{value}))\longrightarrow(p_k\uparrow\gamma\cup\{a\alpha d.\mathrm{value}\},ag[\mathrm{reflist}\cup\{a\}])a\notin\mathrm{Dom}(p_k\uparrow\gamma)$,

$(p_k\uparrow\gamma,ag@(\mathrm{DEREF},a))\longrightarrow(p_k\uparrow\gamma\backslash\{a\alpha v\},ag[\mathrm{reflist}\backslash\{a\}])$,若$a\alpha v\in(p_k\uparrow\gamma)$,

$(p_k\uparrow\gamma,ag@(\mathrm{SEND},ag,\mathrm{msg}))\longrightarrow(p_k\uparrow\gamma,ag)$,

$(p_k\uparrow\gamma,ag@(\mathrm{RECEIVE},\mathrm{comd}))\longrightarrow(p_k\uparrow\gamma,ag[\mathrm{comd}/\mathrm{msg}])$,

$A[x/v]$表示对A体某部分的替换.

agent涉及移动的计算原语分别是:(MOVETO,p),将agent迁移到位置p;(CLONE,p),将agent克隆到位置p.设Γ为agent计算原语可能涉及的多个位置的计算上下文组成的元组,$\Gamma=\langle p_k\uparrow\gamma_i,\ldots,p_l\uparrow\gamma_j\rangle$,则它们的规约规则描述为一般形式:

$$\Gamma,q\uparrow ag@(\mathrm{Primitive},\mathrm{Param})\longrightarrow\Gamma'=\langle\Phi^*(q\uparrow\gamma_i),\Omega^*(p\uparrow\gamma_j)\rangle,ag'=\Psi^*(q\uparrow ag).$$

$\Gamma=\langle q\uparrow\gamma_i,p\uparrow\gamma_j\rangle$,$\Phi,\Omega$分别是对源与目标位置计算上下文的作用,$\Psi$是对agent体的作用,$\Phi^*,\Omega^*,\Psi^*$代表多次作用.

CLONE原语

$\Gamma,q\uparrow ag@(\mathrm{CLONE},p)\longrightarrow\Gamma'=\langle\Phi(q\uparrow\gamma_i),\Omega(p\uparrow\gamma_j)\rangle,ag'=\Psi(q\uparrow ag)$,

$\Phi(q\uparrow\gamma_i)=q\uparrow\gamma_i$,

$\Psi(ag)=\{q\uparrow ag,p\uparrow ag'[\mathrm{reflist}\alpha\mathrm{reflist}',id\alpha id']\}$,

$\Omega(p\uparrow\gamma_j)=p\uparrow\gamma_j[\text{complist}\cup\{ag.id'\}]\cup\{p\uparrow ag.\text{reflist}'\alpha D.V'\}$,若 $q\uparrow ag.\text{reflist}\alpha D.V$.

MOVETO 原语

$\Gamma,q\uparrow ag@(\text{MOVETO},p)\longrightarrow\Gamma'-\langle\Phi^*(q\uparrow\gamma_i),\Omega^*(p\uparrow\gamma_j)\rangle,ag'=\Psi^*(q\uparrow ag)$.

计算组件移动的类型有代码移动、agent 移动、闭包移动和计算环境移动.由前面非形式模型中的描述可知,它们的特征各不相同,因而各种作用都不尽相同.我们分别给出不同范型的计算组件移动的规约式:

(1) 代码移动

若 $q\uparrow ag.t=\text{CODE}$,则

$(\Gamma,q\uparrow ag@(\text{MOVETO},p))\longrightarrow(\Gamma'=\langle\Phi^{CD}(q\uparrow\gamma_i),\Omega^{CD}(p\uparrow\gamma_j)\rangle,ag'=\Psi^{CD}(q\uparrow ag))$,

$\Phi^{CD}(q\uparrow\gamma_i)=q\uparrow\gamma_i[\text{complist}\backslash\{ag.id\}]$,

$\Omega^{CD}(p\uparrow\gamma_j)=p\uparrow\gamma_j[\text{complist}\cup\{ag.id\}]$,

$\Psi^{CD}(q\uparrow ag)=p\uparrow ag$.

(2) Agent 移动

若 $q\uparrow ag.t=\text{AGENT},\text{lodge}(q\uparrow ag.id)=\varnothing$,则

$(\Gamma,q\uparrow ag@(\text{MOVETO},p))\longrightarrow(\Gamma'=\langle\Phi^{AG}(q\uparrow\gamma_i),\Omega^{AG}(p\uparrow\gamma_j)\rangle,ag'=\Psi^{AG}(q\uparrow ag))$,

其中

(i) $\Phi_1(q\uparrow\gamma_i)=q\uparrow\gamma_i[\text{complist}\backslash\{ag.id\}]\backslash\{\text{reflist}\alpha V\}$,

$\Psi_1(q\uparrow ag)=p\uparrow ag[\text{reflist}=\varnothing,\omega=\text{INACTIVE}]$,

$\Omega_1(p\uparrow\gamma_j)=p\uparrow\gamma_j$;

(ii) $\Phi_2(q\uparrow\gamma_i)=q\uparrow\gamma_i$,

$\Psi_2(p\uparrow ag)=p\uparrow ag[\text{reflist}=\text{Add},\omega=\text{ACTIVE}]$,

$\Omega_2(p\uparrow\gamma_j)=p\uparrow\gamma_j[\text{complist}\cup\{ag.id\}]\cup\{\text{Add}\alpha V'\}$,

$\Phi^{AG}(q\uparrow\gamma_i)=\Phi_2\Phi_1(q\uparrow\gamma_j),\Omega^{AG}(p\uparrow\gamma_j)=\Omega_2\Omega_1(p\uparrow\gamma_j),\Psi^{AG}(ag)=\Psi_2\Psi_1(q\uparrow ag)$.

(3) 闭包移动

若 $q\uparrow ag.t=\text{AGENT},\text{lodge}(q\uparrow ag.id)=\varnothing$,则

$(\Gamma,q\uparrow ag@(\text{MOVETO},p))\longrightarrow(\Gamma'=\langle\Phi^{CL}(q\uparrow\gamma_i),\Omega^{CL}(p\uparrow\gamma_j)\rangle,ag'=\Psi^{CL}(q\uparrow ag))$.

闭包移动与 agent 移动的不同在于对资源引用的处理是不同的.并且,闭包移动中引用数据可有 4 类不同的移动,因此 Φ^{CL},Ω^{CL}和 Ψ^{CL}的作用各不相同.

(i) $D.t=\text{COPY}$

$\Phi^{CL}(q\uparrow\gamma_i)=q\uparrow\gamma_i[\text{complist}\backslash\{ag.id\}]$

$\Psi^{CL}(q\uparrow ag)=p\uparrow ag[\text{reflist}\alpha\text{reflist}']$

$\Omega^{CL}(p\uparrow\gamma_j)=p\uparrow\gamma_j[\text{complist}\cup\{ag.id\}]\cup\{\text{reflist}'\alpha D.V'\}$

(ii) $D.t=\text{RESIDENT}$

$\Phi^{CL}(q\uparrow\gamma_i)=q\uparrow\gamma_i[\text{complist}\backslash\{ag.id\}]$

$\Psi^{CL}(q\uparrow ag)=p\uparrow ag[\text{reflist}\alpha p\uparrow\text{reflist}]$

$\Omega^{CL}(p\uparrow\gamma_j)=p\uparrow\gamma_j[\text{complist}\cup\{ag.id\}]$

(iii) $D.t=\text{CARRY}$

$\Phi^{CL}(q\uparrow\gamma_i)=q\uparrow\gamma_i[\text{complist}\backslash\{ag.id\},\text{reflist}\alpha p\uparrow\text{reflist}']$

$\Psi^{CL}(q\uparrow ag)=p\uparrow ag[\text{reflist}\alpha\text{reflist}']$

$\Omega^{CL}(p \uparrow \gamma_j) = p \uparrow \gamma_j[\text{complist} \cup \{ag.id\}] \cup \{\text{reflist}@D.V'\}$

(iv) $D.t = \text{TAKEAWAY}$

$\Phi^{CL}(q \uparrow \gamma_i) = q \uparrow \gamma_i[\text{complist} \setminus \{ag.id\}, \text{reflist}@\perp]$

$\Psi^{CL}(q \uparrow ag) = p \uparrow ag[\text{reflist}@\text{reflist}']$

$\Omega^{CL}(p \uparrow \gamma_j) = p \uparrow \gamma_j[\text{complist} \cup \{ag.id\}] \cup \{\text{reflist}@D.V'\}$

(4) 计算环境移动

若 $q \uparrow ag.t = \text{AGENT}, \text{lodge}(q \uparrow ag.id) \neq \varnothing$,则

$(\Gamma, q \uparrow ag@(\text{MOVETO}, p)) \longrightarrow (\Gamma' = (\Phi^{ENV}(q \uparrow \gamma_i), \Omega^{ENV}(p \uparrow \gamma_j)), ag' = \Psi^{ENV}(q \uparrow ag))$.

计算环境移动是由其载体组件的移动表现的.因此,Φ^{ENV},Ω^{ENV},Ψ^{ENV}是在 agent 移动和闭包移动的基础上,增加考虑了位置关系的变化.

$\Phi^1(q \uparrow \gamma_i) = q \uparrow \gamma_i[\text{childlist} \setminus \{q.id\}]$,

$\Omega_1(p \uparrow \gamma_j) = p \uparrow \gamma_j[\text{childlist} \cup \{q.id\}]$,

$\Psi_1(ag) = ag$,

因此,$\Phi^{ENV}(q \uparrow \gamma_i) = \Phi_1\Phi^{AG}(q \uparrow \gamma_i)$或 $\Phi_1\Phi^{CL}(q \uparrow \gamma_i)$,

$\Omega^{ENV}(p \uparrow \gamma_i) = \Omega_1\Omega^{AG}(p \uparrow \gamma_i)$或 $\Omega_1\Omega^{CL}(p \uparrow \gamma_i)$,

$\Psi^{ENV}(q \uparrow ag) = \Psi_1\Psi^{AG}(q \uparrow ag)$或 $\Psi_1\Psi^{CL}(q \uparrow ag)$.

4 相关工作比较与总结

在移动系统形式化研究领域中,已有一些理论上较完整的形式语言和演算.如扩展了 π 演算的分布式 Join 演算,它引入位置和分布式失败来刻画具有移动性的分布式系统,但其静态全局作用域和分布失败检测,在广域移动系统设计中是要避免的.Klaim 也扩充了位置概念,可以刻画移动 agent 系统,但它不支持层次位置,因而无法描述广泛的实体移动(如移动设备移动).Ambient 演算既抽象位置又抽象 agent、进程等概念,该演算强调 ambient 的层次组织、操纵和访问,但没有从根本上区分开组件与位置,因而未能显式地描述多种实体移动;Mobile Unity 是基于状态转换的移动系统描述语言,它扩充 UNITY[16]语言结构和逻辑系统,以变量共享机制刻画各种移动系统(包括硬件和软件)的暂态交互和组合,使用特殊变量显式地表示位置概念,它可以表示多种范型[17]和多种粒度的移动[18].该语言也没有固有的层次位置,不能表示实体的动态创建,因而系统的动态重配置只是表现在组件交互连接约束条件的变化上[8].

我们没有像这些形式化方法一样,直接定义某种精确的语言或演算,而是通过分析和定义移动系统中的基本概念和它们之间的关系,使用操作语义的规约规则描述各种移动范型,以这样的方法刻画移动系统模型.我们充分吸取了 Ambient 模型的特点,刻画了层次且松散组织的移动系统模型,但同时又强调组件与位置概念的区别,形式地描述了关系变化表现的各种移动范型,尤其是借助寄宿关系,以组件移动表现出寄宿位置的移动.分析结果标识了移动系统模型设计的核心概念和问题,而形式描述则可作为语言构造设计的基础.进一步的工作是基于模态逻辑,设计一个充分支持移动系统模型特点的移动组件系统规范描述语言.

References:

[1] Cardelli, L., Gordon, A. D. Mobile ambients. In: Nivat, M., ed. Foundations of Software Science and Computational Structures, LNCS 1378. Berlin: Springer-Verlag, 1998. 140~155.

[2] Fournet, C., Gonthier, G., Levy, J. J., *et al*. A calculus of mobile agents. In: Montanari, U., Sassone, V., eds.

CONCUR '96: Concurrency Theory, LNCS 1119. Berlin, Heidelberg, New York: Springer-Verlag, 1996. 1~17.
[3] De Nicola, R., Ferrari, G., Pugliese, R. KLAIM: a kernel language for agents interaction and mobility. IEEE Transactions on Software Engineering, 1998,24(5):315~330.
[4] Vitek, J., Castagna, G. Seal: a framework for secure mobile computations. In: Bal, H. E., Belkhouche, B., Cardelli, L., eds. Internet Programming Languages, LNCS 1686. Berlin: Springer-Verlag, 1999. 47~77.
[5] McCann, P. J., Roman, G. C. Compositional programming abstractions for mobile computing. IEEE Transcations on Software Engineering, 1998,24(2):97~110.
[6] Cardelli, L. Abstractions for mobile computations. Microsoft Research, MSR-TR-98-34, 1997.
[7] Mobile Agents. In: Rothermel, K., Popescu-Zeletin, R., eds. Proceedings of the 1st International Workshop (MA'97), LNCS 1219. Berlin, Heidelberg, New York: Springer-Verlag, 1997.
[8] Wei, Jun, Feng, Yu-lin. Analysis of formal models and methods on mobile computing. Journal of Computer Research and Development, 2000,37(2):129~139 (in Chinese).
[9] Satyanarayanan, M. Mobile information access. IEEE Personal Communications, 1996,3(1):20~33.
[10] Satyanarayanan, M., Noble, B., Kumar, P., *et al*. Application-Aware adaptation for mobile computing. Operating Systems Review, 1995,29(1):52~55.
[11] White, J. E. Mobile agents. In: Bradshaw, J., ed. Software Agents. AAAI Press/The MIT Press, 1996.
[12] Cardelli, L. A language with distributed scope. In: Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. San Francisco, CA: ACM Press, 1995. 286~297.
[13] Kiniry, J., Zimmerman, D. A hands-on look at Java mobile agents. IEEE Internet Computing, 1997,1(4):21~30.
[14] Carriero, N., Gelernter, D. Linda in context. CACM, 1989,32(4):444~458.
[15] Milner, R., Parrow, J., Walker, D. A calculus of mobile processes I. Information and Computation, 1992,100(1):1~77.
[16] Chandy, K. M., Misra, J. Parallel Program Design: a Foundation. New York, NY: Addison-Wesley, 1988.
[17] Picco, G. P., Roman, G. C., McCann, P. Expressing code mobility in mobile UNITY. In: Jazayeri, M., Schauer, H., eds. Proceedings of the 6th European Software Engineering Conference (ESEC/FSE'97), LNCS 1301. Berlin: Springer-Verlag, 1997. 500~518.
[18] Mascolo, C., Picco, G. P., Roman, G. C. A fine-grained model for code mobility. Washington University Technique Report, WUCS-99-07, 1999.

附中文参考文献:

[8] 魏峻,冯玉琳. 移动计算形式理论分析与研究. 计算机研究与发展,2000,37(2):129~139.

Specifying and Analyzing Model for Mobile Component Systems

WEI Jun, ZHOU Huan

(*Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*);
(*Object Technology Center, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)
E-mail: wj@ox.ios.ac.cn; quan@otcaix.iscas.ac.cn

Received June 28, 1999; accepted November 23, 1999

Abstract: Mobile computing is a new computing paradigm. A main feature of this paradigm is dynamically changing the binding between computational components and computational locations, which brings new requirements to the design of distributed systems. In this paper, an abstract model for mobile component systems is proposed. Not only Location, Component, Mobility and Resource Access as the basic elements of mobile systems are identified in the abstract model, but modeling mobile systems by basic elements, relationships among elements and relationship changes is also characterized. Based on set theory and operational reduction rules, these elements, relationships and mobility mechanisms are specified in a formal manner. The analysis and formal specifications can be applied to the design basis of mobile systems and their specification languages.

Key words: model for mobile component system; formal method; mobility; location; component; component relationship