文章编号：1001-2486（2000）06-0047-05

移动智能体的形式化研究*

吴泉源，吴刚，王怀民

（国防科技大学计算机学院，湖南 长沙 410073）

摘 要：移动智能体是当前分布计算领域的研究热点之一，但是其理论研究还处于一个很不成熟的阶段。文中阐述了对移动智能体作形式化研究的必要性，介绍了当前的研究现状，分析了其中的不足，并进一步给出了对后续工作的展望。

关键词：移动智能体；并发模型；进程代数；π 演算

中图分类号：TP311 文献标识码：A

Research on Formal Model of Mobile Agent

WU Quan-Yuan，WU Gang，WANG Huai-Min

（College of Computer，National University of Defence Technology，Changsha 410073）

Abstract：Mobile agent receives much attention recently in the distributed computing field. But the research on its formal model is immature. The necessity of the formal research of mobile agent is analyzed firstly. Then we introduce several representative formal models，analyze their drawbacks，and present the prospects for the future work.

Key words：mobile agent；concurrent model；process algebra；π-calculus

20 世纪 90 年代以来，随着全球通信网与 Internet 的蓬勃发展，以及网络技术的逐渐成熟，分布式计算系统（Distributed Computing System）已经成为计算机科学领域的一项关键性主流技术和一种重要的计算范型。随着应用的不断深入，分布式系统也变得日趋复杂和庞大，提出一个合理的抽象概念模型来刻画分布式计算系统中的计算实体对于应用的设计与实现是极为重要的。于是人们引入了智能体（Agent）的概念。一个软件 Agent 可以定义为一个能够与环境交互并作出反应的自主软件实体，其主要特性包括自主性、交互性和反应性。

移动智能体（Mobile Agent）作为一种特殊的 Agent，也是作为一种新型的分布计算模式，近年来得到了广泛的关注与研究[1,2,3,4]。在移动智能体计算模式下，Agent 通过在网络中的自主移动与运行来完成任务，并使得大多数的交互行为在本地发生。人们普遍希望能借助移动智能体计算模型实现分布式应用从局域网平台向大规模（Large-scale）的网络计算平台的转移。

如图 1 所示，一个移动智能体是一个活动的、自主的计算实体，能从异构网络中的一台机器移动到另一台上执行。一般认为，智能体的迁移包括了相应代码、数据和运行时状态的移动。根据 Agent 的概念描述，一个移动智能体至少应包括自主性、交互性、反应性和移动性这些特性。自主性使得智能体可以不受外在的直接干预而独立运行；交互性使之能与包括人、设备和其他智能体在内的实体进行通讯；反应性使之能对环境的改变作出及时的反应；移动性使得智能体能在运行时实现异构网络上的自主迁移。

1 移动智能体形式化研究的必要性

分布计算系统的实现要求必须是可靠且易于维护的，这就需要相应的理论指导以及严格的形式化开发方法和工具的支持。随着分布计算技术的不断发展及其应用的不断深入，分布计算系统的开发面临着许多理论课题，主要体现在：如何获取、规范和分析分布计算系统的需求；如何基于某种抽象的

* 收稿日期：2000-06-20
基金项目：国家自然科学基金（69833030），国家 863 计划项目（863-306-ZD02-01-2）
作者简介：吴泉源（1942-），男，教授，博士生导师。

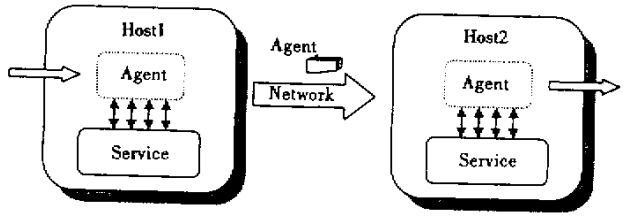

图1 Mobile agent 计算模式

Fig.1 Mobile agent computing model

计算模型来指导分布计算系统的设计。等等。

为了规范分布计算系统中的各个计算实体、描述计算实体之间的合作方式和交互行为，人们引入了 Agent 这样一个合理的抽象概念模型，并展开了对 Agent 的形式化研究，旨在获取分布计算系统的特性，以指导基于 Agent 的分布计算系统的设计与开发。

1993年，Yoav Shoham 在杂志《Artificial Intelligence》上发表了题为“Agent-Oriented Programming”的文章[5]，AOP 是一种以计算的社会观为基础的新型程序设计语言。Shoham 认为一个完整的 AOP 系统应包括以下三个主要成分：(1) 一种约束的形式化语言，具有清晰的语法和语义来定义 Agent 的认知状态；(2) 一种解释型的程序设计语言，用于定义 Agent 以及对 Agent 进行编程，并具有请求、通知等通讯原语，其语义应忠实于认知状态的语义；(3) 一个 Agent 的生成器（Agentifier)。

从理论的角度来看，并发模型被认为是分布式计算的形式化基础。研究人员在规约模型、逻辑模型、进程代数、Petri 网等不同的体系下分别对基于 Agent 的分布式计算进行了形式化研究，包括一阶分枝时序逻辑[6]、异步多体的 π 演算[7]等等。

移动智能体（Mobile Agent）作为分布式计算中的新生事物，其计算实体在网络环境下的可移动性为基于并发模型的形式化研究带来了新课题。但是这方面的理论研究还处于一个很不成熟的研究阶段，导致了已有的移动智能体系统缺乏理论的指导，各自在移动智能体系统的应用、安全、可用性、异构性等方面有着不同的假设前提，缺乏一个统一的术语规范和公共的方法学框架。

近期出现了许多支持移动应用的编程语言，包括 Java 和一些面向移动智能体计算模型的原型语言。Java 使得客户可以从其它站点上下载自包含的程序作本地执行，但并不能直接支持移动智能体的计算模型；而那些原型语言，包括 Facile[8]，Obliq[9]，CML[10]和 Telescript[11]等，虽然直接面向移动计算设计，但并没有坚实的形式化模型作基础，各自以不同的方式实现了迁移机制，使得各自的移动语义很难界定，相互之间很难比较与互模拟。

因而，我们需要一个抽象的语义框架，使得人们可以更好地理解和形式化面向移动计算的编程语言，并以此作为设计和实现分布式编程语言及移动计算应用的形式化基础。

2 移动智能体形式化研究的现状

作为分布式计算的形式化基础，并发模型也就当然地成为研究移动智能体理论模型的起点。已有的研究工作基本上是从某个较成熟的形式系统出发，针对移动智能体计算中计算实体的可移动特征作了相应的扩展，使之能够有效地表达并发对象的移动特性。大致可以将它们分成三类：

·归约模型。Yonezawa 实验室的 λdist 演算[12]是在传值（call-by-value）λ 演算基础上所作的分布式扩展。计算实体的语法被定义为

$$e ::= p \mid go \mid x \mid \lambda x.e \mid ee \mid < e >$$

其中 x 是一个变量；$\lambda x.e$ 是一个抽象；ee 是一个应用；p 表示运行环境，类似于 Telescript 中 Place 的概念；$< e >$ 表示一个 agent；而 agent 的迁移用 $go\ p$ 来表示，p 为移动的目的地，在执行了 go 表达

式之后，agent 的后续部分将移动到 p 继续执行。为了能够表达多个移动语言系统中的不同数据移动机制，λdist 演算还定义了六种数据移动类型：copy、resident、carry、proper、takeaway、ubiq。

λdist 演算能描述 Agent 移动机制和一些其它的分布计算机制，诸如同步、远程引用等。通过对 Obliq、Telescript、Java 和 FACILE 等移动智能体语言系统的形式化描述，λdist 演算还试图为它们之间的互模拟提供支持。但由于 λ 演算本身描述的是串行计算且又不支持面向对象的描述，为实现对异步通信和对象实体的描述，λdist 演算引入了许多复杂的结构，在一定程度上丧失了 λ 演算包括简洁性在内的许多特性。

·时序逻辑模型。UNITY[13]是一种以时序逻辑为基础的基于状态的形式化方法，但标准的 UNITY 由于只能表达计算实体的静态结构，并不适合于处理移动计算系统中的问题。McCann、Roman 等人便在 UNITY 的基础上作了相应扩展，以支持可动态重配置的分布式系统的分析与建模，并称之为 Mobile UNITY[14]。Mobile UNITY 提供了一组形式化的符号，以捕获移动行为和移动结点之间的瞬时交互，并包含有一个断言式的验证逻辑，用于推理移动计算系统的特性。

一个用 Mobile UNITY 描述的系统由三个部分组成[15]：Declare 部分、Component 部分和 Interaction 部分。Declare 部分描述了程序组件中的变量及其类型，包括刻画位置的变量；Component 部分对程序中变量作实例化操作；Interaction 部分给出了程序组件实例的瞬时交互描述。Mobile UNITY 描述的移动计算系统是在程序组件之间的瞬时连接条件下进行操作的，连接由当前的位置决定，因此位置便成为该模型中的重要概念。但是模型并没有对位置变量做类型限制，它可以是离散或连续的，也可以是移动平台的物理坐标或移动 agent 所在的网络。进程通过设置位置变量非显式地刻划移动，形如 λ：= NewLoc（λ），该函数将返回一个新位置，并表示进程将迁移到此新位置继续执行。

然而，Mobile UNITY 研究的初衷是为了解决移动通讯环境下由于硬件设备的移动而引发的计算组件交互关系的重配置，以及由于移动网络链路的易断接而引发的交互重建等问题。并不是针对移动智能体这样的软件实体移动而设计的。Picco 在文献［16］中研究了将 Mobile UNITY 用于移动智能体形式化表示的可行性，认为 Mobile UNITY 作为一个形式系统，并没有限制移动实体的具体定位，也并不排除对位置空间这一概念更为抽象的理解。因而，可以用之来描述移动智能体计算模型，并能利用 Mobile UNITY 的验证逻辑来证明移动智能体应用程序的正确性。

·进程代数。移动智能体的形式化研究中最活跃的一组，是从进程代数入手的。包括 Dπ* 演算[17]、分布式 Join 演算[18]、Dπ 演算[19]、KLAIM[20]及 Amadio 的一些工作[21,22]等。

π 演算[23]作为一种并发计算的理论模型，通过少量的结构为并发计算提供了强大的描述能力和简洁的语义说明。但是并发计算与分布式计算之间有着许多不同，尤其对于分布式移动计算系统而言，π 演算存在许多不足。例如移动智能体系统中计算实体的移动是一个异步的过程，而典型 π 演算中原子性的非本地交互是用基于会合的同步通信来描述的；再例如 π 演算中没有显式的位置概念，不便于描述运行实体的自主移动，等等。因而，许多研究工作都通过对 π 演算的分布式扩展来实现了对移动计算的有效表述。（1）Hennessy 的 Dπ 演算。在通用的 Polyadicπ 演算[7]的基础上，他引入了显式的位置概念，并扩展了位置之间的移动原语及新位置的创建原语。在 Dπ 演算中，异步的消息传递与 Agent 的显式移动被统一为一种形式，任何异步的消息传递均通过子 Agent 的移动加上本地的同步通讯来实现。（2）分布式 Join 演算。Join 演算是 Fournet 等人提出的一个 π 演算的异步化变种，它不仅有着和 π 演算相同的表达能力，而且具有更好的位置表达能力和静态限域规则，适合于对分布式系统的描述。在此基础上，Fournet 又通过扩展显式的位置概念和移动原语，得到了分布式 Join 演算，使之有能力表达 Mobile Agent 在物理站点之间的移动行为，并试图以此作为分布式编程语言的基础。其中，Mobile Agent 用位置来表示，并利用树型结构的子位置描述了子 Agent 的概念。在分布式 Join 演算中，异步消息与 Agent 的移动是通过不同的原语操作来实现的，它还提供了一个简单的失败模型，使得一个位置的失败能被其它正在运行的位置检测到，并允许失败的恢复过程。（3）Dπ* 演算（分布式 Blue 演算）。Blue 演算（π*）[24]是一个结合了 λ 演算特点的 π 演算异步扩展。Silvano 在此基础上又引入了位置、带有位置的进程以及代码传递的原语，得到了一个简单的 Blue 演算之分布式变种——Dπ* 演

算。其研究目的是探讨并发对象的移动特性能否与对象的其它特性（如同步约束等）相正交的定义。

Nicola、Ferrari 和 Pugliese 的 KLAIM 也是为描述智能体的交互与移动而开发的。与上述基于 π 演算的形式系统不同，它是一个基于 Linda[25] 的变种，可以看作一个异步、高阶的进程演算。其基本操作来源于多元组空间概念下的 Linda，并在此基础上扩展了显式的结点位置信息与一组借鉴于 CCS 的原语操作。同时，KLAIM 的类型系统也被开发出来以控制移动智能体对资源的访问权限。

面向网络计算编程的一个核心问题就是安全性，如数据的保密与完整性问题。在一个开放的分布式系统中，我们不能保证网络上的 Agent 都是善意的，因而防止恶意的 Agent 访问或修改私人数据便十分重要，接受移动 Agent 的站点必须能够检测 Agent 的行为，以确保其不会对本地系统造成破坏。当然，动态地对 Agent 作身份鉴别和权限控制可以达到目的，可是在分布式系统中试图每个资源的每次使用都进行动态的认证是很不可取的，这将大大降低系统的性能。因此，许多研究将安全问题集成到形式化系统中，通过类型系统的静态分析来直接推导安全方面的性质，保证对系统资源的安全访问。例如，有许多基于通道的分布式 π 演算类型系统[26,21,27] 被开发了出来。在文献［26］中，Pierce 和 Sangiorgi 定义了一个 π 演算的类型系统以限制通道上的读写能力；Amadio 在文献［21］中给出的类型系统保证了某个通道名只能在全局的一个位置上的定义；Kobayashi 在文献［27］中给出的类型系统能保证在某一通道上传输的线性特征。KLAIM 的类型系统[28] 则是基于元组空间定义的，与上述基于 π 演算的类型系统相比，其资源控制的粒度更大。

3 进一步的工作展望

上述的形式系统都为移动智能体系统提供了不同程度的理论基础，总体的构造思想也大致相同，如均提供了显式的位置概念和 Agent 的移动原语。但是各自在实现过程中却有许多不同，从上述的分析已可看出一二；各自的研究深度也参差不齐，许多进一步的开发与完善工作仍需进行。以 Hennessy 的 Dπ 演算为例，由于开发的初衷是为了提供一个分布式 π 演算的类型系统，以保证 Agent 在获得权限之前不能对资源进行非法存取，因而 Dπ 演算中混杂了许多与类型相关的概念与标识，增加了读者理解的难度。同时为了简化工作，它没有考虑通道上的一组线性传递引发的原子性问题，在其类型系统中也不允许递归类型的出现，而没有递归类型，许多有趣的系统结构均无法描述，如涉及表、树等数据类型的系统。

本文认为进程代数是对移动智能体作形式化研究的有力武器，具有良好的研究前景。利用它可以提供一个有效的模型，对移动智能体系统作形式化描述，并且可以推演计算，验证实现的正确性。因此，在进一步的工作中，将采用进程代数的手段，通过对 Polyadicπ 演算的变种与扩充，得到一个异步（Asynchronism）、高阶（High-Order）的 π 演算来作为移动智能体系统的形式化基础，并给出它的类型系统，以推导移动计算中的安全问题。具体措施包括：

· 异步化变种。移动智能体系统中计算实体的迁移是一个异步的过程，对于它的形式化描述也应基于异步的并发演算。而 Polyadicπ 演算本身是一种基于同步交互的形式系统，需要对之作异步化变种。

· 位置概念与移动原语的引入。π 演算有一个平行的进程空间，所有进程都处于同一层次，利用共享通道实现交互。但在移动智能体系统中，Agent 是与位置绑定的，并能在位置之间作自主移动，通道也具有不同的作用范围。因此，需要引入位置的概念及显式的移动操作原语。

· 高阶变种。虽然 π 演算有能力表述基本的高阶演算，如 λ 演算。但是 π 演算只能传递对计算实体的引用，不能传递计算实体本身。而高阶的进程演算能提供对在通道上传递活跃实体这一概念的理解，此时传递实体的通道可能跨越不同的位置和安全界限。因此，有必要对 π 演算作高阶的变种。这样也就可以直接讨论移动计算系统中的一些其他问题。例如，在整个计算实体移动时，如何处理标识的闭包与范围约束。

· 类型系统。为了在编译时作有关类型的自动检测，保证对系统资源的安全访问，编程语言应该有一个好的类型系统。因此，我们将在上述异步、高阶 π 演算的基础上，引入一组类型（包括可递归

类型），及类型之间的相互关系，然后给出其类型系统，并讨论它的完备性问题。

4 结论

为了研究移动智能体系统模型，更好地开发移动应用系统，必须做相关的形式化研究。本文着重介绍了当前有代表性的研究成果，包括 λdist 演算、Mobile UNITY、分布式 π 演算等等。并分析了其中的不足，进一步给出了对后续工作的展望。希望能以此作为移动智能体形式化研究的基础。

参考文献：

[1] Harrison C G，Chess D M，Kershenbaum A. Mobile Agents：Are They a Good Idea？[R]. IBM Research Report，RC19887，1994.

[2] ObjectSpace Voyager [EB]. ObjectSpace Inc.，http：//www.objectspace.com/voyager/，1997.

[3] The IBM Aglets workbench [EB]. IBM Corp.，http：//www.trl.ibm.co.jp/aglets/，1996.

[4] OMG Mobile Agent Systems Interoperability Facility Specification [S]. OMG TC Doc. ORBOS/97-10-05.

[5] Shoham Y. Agent-Oriented Programming [J]. Artificial Intelligence，60 (1993)：51-92

[6] 毛新军. Multi-Agent 系统中 Agent 计算的理论框架 [D]，博士论文，国防科技大学，1998.

[7] Robin Milner. The polyadic π-calculus：a tutorial [R]. Technical Report ECS-LFCS-91-180，1991.

[8] Giacolone A，Mishra P and Prasad S. Facile：A Symmetric Integration of Concurrent and Fnctional Programming [J]. J. Parallel Programming，vol.18，no.2，1989.

[9] Cardelli L.. A Language with Distributed Scope [J]. Computing Systems，8 (1)：27-59，Jan.1995.

[10] Reppy J. Higher Order Concurrency [D]. PhD thesis，Cornell University，Tr-92-1285，1992.

[11] White J E. Mobile Agents [A]. Software Agents，J.M. Bradshaw，ed.pp 437-471，1996.

[12] Tatsurou S and Akinori Y. A Calculus with Code Mobility [A]. Proc. FMOODS ' 97，Canterbury，July，1997.

[13] Chandy K，Misra J. Parallel Program Design：A Foundation [M]. Addison-Wesley，1988.

[14] Roman G C，McCann P J，Plun J. Mobile UNITY：Reasoning and Specification in Mobile Computing [J]. ACM Trans. on Software Engineering and Methodology 6 (3)：250-282，1997.

[15] McCann P J，Roman G C. Compositional Programming Abstractions for Mobile Computing [J]. IEEE Trans. on Software Engineering，24 (2)：97-110，1998.

[16] Picco G P. Understanding，Evaluating，Formalizing，and Exploiting Code Mobility [D]. PhD thesis，Politecnico Di Torino，1998.

[17] Silvano D Z. Quiet and Bouncing Objects：Two Migration Abstractions in a Simple Distributed Blue Calulus [A]. Proceedings of the Workshop on Semantics of Objects as Process '98，35-42，Denmark，1998.

[18] Fournet C. and Gonthier G. The refliexive CHAM and the Join-calculus [A]. Conference Record of the ACM Symposium on Principles of Programming Languages，Paris，Jan 1996.

[19] Hennessy M，Riely J. Resource Access Control in Systems of Mobile Agents [R]. Technical Reports，University of Sussex，1998.

[20] Nicola R D，Ferrari G L，Pugliese R. KLAIM：A Kernel Language for Agents Interaction and Mobility [J]. IEEE Trans. on Software Engineering，24 (5)：315-330，1998.

[21] Amadio R. An asynchronous model of locality，failure，and process mobility [A]. COOR-DINATION ' 97，Vol 1282 of Lecture Notes in Computer Science，Springer-Verlag，1997.

[22] Amadio R and Prasad S. Locations and Failures [A]. Proc. 14th Foundations of Software Technology and Theoretical Computer Science，Vol 880 of Lecture Notes in Computer Science，Springer-Verlag，1994.

[23] Milner R，Parrow J，Walker D. A Calculus of Mobile Processes，Part I and II [J]. Journal of Information and Computation，100 (9)：1-77，1992.

[24] Boudol G. The π-calculus in direct style [A]. In Conference Record of POPL ' 97：The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Progamming Languages，228-241，Paris，France，1997.

[25] Carriero N，Gelernter D. Linda in context [J]. Communications of the ACM，32 (4)：444-458，1989.

[26] Pierce B，Sangiorgi D. Typing and subtyping for mobile processes [J]. Mathematical Structures in Computer Science，6 (5)：409-454，1996.

[27] Kobayashi N，Pierce B，Turner D. Linearity and the Pi-calculus [A]. In Conference Record of the ACM Symposium on Principles of Programming Languages，Paris，1996.1，ACM Press.

[28] Nicola R D，Ferrari G L，Pugliese R. Coordinating mobile agents via blackboards and access rights [A]. In COORDINATION ' 97，Vol. 1282 of Lecture Notes in Computer Science，Springer-Verlag，1997.