形式化多主体系统中的交互及交互协议*

焦文品, 史忠植

(中国科学院 计算技术研究所,北京 100080)

E-mail: shizz@ics.ict.ac.cn

http://ics.ict.ac.cn/

摘要: 深入研究了多主体系统中的交互及其协议,并用一种进程演算:即π演算进行了形式化的描述.为了研究主体之间的交互,首先对参与交互的主体的行为进行了分类,并形式化地描述了其行为规范,然后用进程定义了主体间的交互协议,并在此基础上分析了主体交互的一致性及无死锁性.

关键词: 多主体系统;交互;π演算

中图法分类号: TP18 文献标识码: A

主体(agent)技术与多主体(multi-agent)系统已成为分布式人工智能领域继分布式问题求解之后的又一个研究热点.与分布式问题求解不同的是,多主体系统所关心的是如何利用现存的自主主体来共同完成某个目标[1].在多主体系统中,单个主体可能由于所掌握的信息不够完整,也可能由于能力或资源不够的缘故,无法独立完成某项任务,这时主体必须与其他主体进行交互,在能力和资源上相互支持,以最终实现系统的目标.用形式化方法研究多主体系统中的交互,不仅有利于更好地把握主体间的交互行为及其特征,还可以为多主体系统的设计和开发提供可靠的主体交互机制,保证主体为实现系统目标而使交互正常进行.鉴于主体交互行为以及主体交互结构都具有动态性,在下文中将采用进程代数的方法来刻画多主体系统中的交互行为及交互协议,并将主体行为规范及主体间的交互协议都定义成进程.我们选择π演算[2]作为形式化手段,这主要是基于以下几方面的原因:

• π演算是一种基于命名概念的并发计算模型,它可以很自然地表示出具有动态结构的进程内及进程间的交互,是一种刻画通信系统的进程演算.在π演算中,进程通信是最主要的计算手段.

• 主体交互是一个动态的过程.在用逻辑方法来描述主体或多主体系统时[3],主体及多主体系统都是静态的.而π演算不仅可以刻画进程间的组合,将主体和交互协议结合起来进行考察,以研究主体交互的特征,而且π演算本身就具有刻画动态行为的能力.

• 主体是一种类似于进程的、并发执行的实体[1].主体行为的并发性无论是用经典逻辑或用非经典逻辑都难以刻画.而对π演算来说,刻画主体行为的并发性显得相当容易.

• 主体间一般通过通信来进行交互.在采用传统的逻辑方法时,必须加入非逻辑性手段来表示主体通信[3].π演算作为一种刻画通信系统的进程演算,它在刻画主体间的交互时无疑具有得天独厚的优势.

因为主体交互是在主体行动过程中进行的,本文首先用π演算进程定义了主体的行为规范及

* 收稿日期: 1999-05-10; 修改日期: 2000-04-13

基金项目: 国家863高科技发展计划资助项目(863-306-ZT02-01-3)

作者简介: 焦文品(1969-),男,湖北天门人,博士,主要研究领域为计算机软件与理论,智能软件;史忠植(1941-),男,江苏宜兴人,研究员,博士生导师,主要研究领域为智能软件,人工智能.

主体间的交互协议,然后在此基础上对主体交互的一致性及无死锁性进行了分析.

1 主体行为规范

主体的行为规范规定了主体为达到某种目标而采取的行动,也约束了当主体处于某种状态(或感知到某种刺激)时的行为模式.主体行为主要包括收集服务、申明服务、请求服务、搜索服务主体、提供服务以及那些与交互无关的内部行为等.

1.1 目标规范

主体的目标规范用来刻画主体为达到自己的目标而采取行动的全过程.主体为了实现自身的目标,可能需要其他主体提供帮助,并通过通信向外界提出服务请求.一般来说,主体实现目标的行为过程为:产生需求→在所收集的服务列表中搜索能满足需求的服务主体→提出请求→等待响应→运用所申请到的服务完成自己的任务.

主体目标 g 可以递归地定义成:可立即实现的空目标(nil)、或者一次服务请求以及尚未实现的子目标 g'.主体为实现目标的行为规范可以用 π 演算定义成如下的进程:

$$GoalSpec(g)=(vg')([g=nil]\mathbf{0}|\overline{q}(r_1)\cdot r_1(s)|QueryServer(r_1)|Req(w,s)\cdot Apply(w)\cdot GoalSpec(g')),$$

其中 g 为主体目标.当主体需要服务时,先通过子进程 $QueryServer$ 搜索服务主体,然后通过子进程 Req 获取服务的指针,并交给子进程 $Apply$ 在恰当的地方使用所获得的服务;此后,主体可以再请求其他服务以完成尚未完成的子任务 g'.

假设主体保存有一张(请求,服务)对照表,搜索服务主体的进程可以定义为

$$QueryServer(r)=(vx)q(x)\cdot[x=r]\overline{x}(s).$$

此进程根据请求名 r 查询主体所收集到的服务列表,并返回相应服务的名称 s.

向外界提出服务需求的主体请求进程可以定义为

$$Req(w,r)=(vnid)\overline{r}(w)\cdot w(nid)\cdot\overline{nid},$$

其中 r 为主体所请求的服务名,w 为获取服务的位置,nid 为服务的指针.服务请求子进程可以非形式化地描述为:哪里请求,请求什么.

1.2 申明服务

有能力向外界提供服务的主体不必关心到底向谁提供服务,只有当出现请求时才给予响应,但为了保证其他主体能够正确地查找到谁能提供对应的服务,它应该首先向外界申明自己的能力.主体申明服务相当于向那些正在收集服务的主体广播服务名的过程:

$$DeclareService(r,s)=(vid)[s=nil]\overline{id}|(id\cdot\mathbf{0}+!(vp)\overline{service}(p)\cdot\overline{p}(r)\cdot\overline{p}(s)),$$

即如果存在新的服务,则通过端口 $service$ 向外界输出与请求 r 相对应的服务名称 s.

1.3 收集服务

与申明服务过程相对应,主体在申请服务前必须知道其他主体能提供什么样的服务,因此它必须随时收集相关的信息.与申明服务进程对应的收集服务进程为

$$CollectService(r,s)=service(p)\cdot p(r)\cdot p(s)\cdot UpdateReqSrv(r,s),$$

即通过端口 $service$ 收集并建立(请求,服务)对照表(即 (r,s) 表).

1.4 提供服务

主体能够感知外界的刺激并理解刺激所蕴含的意图,然后依此作出相应的反应,如主体在感知

到外界请求提供服务的刺激后,响应请求并提供对应的服务.向外界提供服务的子进程可以非形式化地描述为:向谁提供,提供什么.即,

$$Prov(s)=(vx,id)!(s(x)\cdot\overline{x}(id)\cdot id\cdot Service(s)),$$

其中 x 表示服务将送往何处,s 为外界所请求的服务名,$Service(s)$ 为服务 s 的具体实现,id 为主体的标识,用来充当服务指针.主体通过端口 s 获得外界请求后,通过端口 x 向外界发送服务 s 的指针.

2 交互协议

主体间的交互协议规定了主体在交互过程中应遵循的规则或准则.交互协议的作用在于协助交互主体寻找合适的交互对象,并规定交互过程的模式.我们将交互协议分为两部分:服务查询以及请求与响应.

(1) 服务查询

主体在与其他主体发生交互前必须先找到交互对象,即找到能提供其所需服务的对等主体.服务查询的过程一般如下:

i) 前期准备:在查询服务前,主体必须已经知道其他主体能够提供什么样的服务,这样,在查询时就不至于漫无目的.前期准备包括申明服务及收集服务,它们是两个互补的过程.

$$Prepare=(vt,x,y)[s=nil]\overline{t}|(t\cdot\mathbf{0}+!(vp)\overline{service}(p)\cdot\overline{p}(r)\cdot\overline{p}(s))$$
$$service(p)\cdot p(r)\cdot p(s)\cdot UpdateReqSrv(x,y).$$

ii) 提出查询,即产生查询请求,并等待查询结果返回.

$$NewQuery(r)=(vx)\overline{q}(r)\cdot r(x).$$

iii) 查询并返回结果,即根据前期准备阶段所收集到的服务列表查询请求所对应的服务,并返回相应的服务名称.

$$ReplyQuery(r,s)=(vx)q(x)\cdot[x=r]\overline{x}(s).$$

于是,服务查询协议可以定义为

$$QueryServiceIP=Prepare|(NewQuery(r)|ReplyQuery(r,s)).$$

(2) 请求与响应

主体在找到交互对象之后,紧接着的交互实质上是一个请求与响应请求的过程.请求与响应过程一般为:

i) 提出服务请求,即产生服务需求,并等待其他主体提供相应的服务,以便在获得所需的服务后在合适的地方运用该服务来实现自己的目标.

$$Request(w,r)=(vx)\overline{r}(w)\cdot w(x)\cdot\overline{x}\cdot Apply(w).$$

ii) 响应请求并返回服务,即等待其他主体发出的服务请求,并返回对应的服务.

$$Reply(s)=(vx,z)!(s(x)\cdot\overline{x}(z)\cdot z\cdot Service(s)).$$

于是,请求与响应协议可以定义为

$$Request\&ReplyIP=Request(w,r)|Reply(s).$$

(3) 交互协议

综合上述两种子协议,交互协议可以定义成它们的组合:

$$InteractionProtocol=QueryServiceIP\ Request\&ReplyIP.$$

3 主体行为与交互协议的一致性分析

独立存在的主体因为交互的需要,而通过交互协议联系在一起,但由于主体具有自主性,它在与其他主体进行交互时,会根据自身的需要自主地采取行动,因此保证主体行为与交互协议之间的一致性对主体能否顺利完成自身的目标以及多主体之间的交互能否顺利进行都是至关重要的.主体行为与交互协议是否一致实质上是指主体的行为规范是否遵循交互协议,即主体是否按照交互协议所规定的行为模式在进行交互.

主体行为除了交互以外,还有很多私下活动,甚至包括主体内部的通信活动.但如果我们以旁观者的角度来观察主体的行为,可以说只有主体间的通信过程是可见的,而主体内部的行为对外界来说都是不可见的.只要那些不可见的行为不影响主体间可见的交互行为,而可见的行为与交互协议能保持一致,我们就可以说主体行为与交互协议是一致的.

定义 1(可观察的,observable). 设 B 是出现在 A 中的一个动程,若存在 B 的某次出现没有前缀,即并不是 B 在 A 中的所有出现都是以 $\alpha \cdot B$ 的形式存在的(其中 α 为一动作),则说 B 在 A 中的出现是无监督的(unguarded).若在进程 P 中存在无监督出现的 $\alpha \cdot A$,且 α 是不受限制的,则说进程 P 相对于动作 α 是可观察的,记作 $P \downarrow_{\alpha}$.

定义 2(弱约简模拟,weak reduction simulation). 弱约简模拟为具有如下特征的关系:若 P 弱约简模拟 Q,记作 PSQ,则若 $P \rightarrow P'$,则存在 Q' 使得 $Q \rightarrow^{*} Q'$,且 $P'SQ'$.即如果 P 弱约简模拟 Q,并且 P 经过某个动作 α 后演算成 P',那么存在一系列动作 $\alpha_1, \alpha_2, \ldots, \alpha_n$,使得 Q 演算成 Q',且 P' 也弱约简模拟 Q'.

定义 3(可观察的弱约简模拟). 可观察的弱约简模拟是具有如下特征的二元关系:若 PSQ,则(1) 若 $P \rightarrow P'$,$Q \rightarrow^{*} Q'$ 且 $P'SQ'$;(2) 对于每个 α,若 $P \downarrow_{\alpha}$,则 $Q \rightarrow^{\times} \downarrow_{\alpha}$,其中 $\rightarrow^{\times}$ 表示一次约简的自反传递闭包.

另一方面,交互协议所关心的是主体交互过程应遵循的原则,对主体在交互过程中所交互的具体内容并不关心.而在定义具体的主体行为时,行为规范中所使用的各种名字(如端口、请求、服务名等)会因主体的不同而不同,主体在行动过程中,每一步动作的名字显然会与交互协议中所规定的有所不同,因此,主体行为进程若仅可观察地模拟交互协议进程,还无法说明主体行为与交互协议的一致性.

定义 4(进程 P 遵循 Q). 设 $\sigma: N \rightarrow N$ 为一个名字替换函数,其中 N 为进程中的名字集,在应用时,用 $P\sigma$ 来表示对进程 P 的一次替换,用 $\{y_i/x_i\}_{1 \leqslant i \leqslant n}$ 来表示具体的替换 σ,也就是说,当 $1 \leqslant i \leqslant n$ 时,$x_i\sigma = y_i$,否则 $x\sigma = x$.若对于所有替换函数 σ,$P\sigma$ 可观察地弱约简模拟 $Q\sigma$,则说进程 P 遵循 Q.

定理 1. 前文所定义的主体行为规范遵循交互协议.

证明:略.

4 主体交互的无死锁性分析

在多主体系统中,主体行为与交互协议的一致性保证主体行为规范是合式的(well-formed),即没有违背公共行为准则,但这并不一定能保证主体交互的顺利进行.例如,主体 A_1 和 A_2 的行为分别为

$$A_1 = (\nu x)Req(x, s_2) \cdot Prov(s_1), \quad A_2 = (\nu x)Req(x, s_1) \cdot Prov(s_2).$$

不难看出,主体 A_1 与 A_2 之间的交互 $A_1 | A_2$ 无法顺利进行,它们会一直等待对方提供所需的服务

而有可能陷入僵局,这表明 A_1 与 A_2 之间的交互出现了死锁.

出现上述现象的原因在于在主体的目标规范中出现了串行的请求与提供服务动作,因此,我们有必要对主体的行为规范进行适当的约束以避免上述现象的发生,即避免死锁的出现.

定义 5. 在主体为实现其目标而与其他主体发生交互的过程中,若主体所发出的服务请求最终总能得到满足,则说交互是无死锁的.

我们假定:(1) 主体不主动提供服务,即只有在响应外界的请求时才提供服务.这就是说,(2) 主体在实现其目标的行为规范(即目标规范)中不存在提供服务的动作,即提供服务的进程总是与目标规范进程并发进行的.

定理 2. 若主体的目标规范中不包含提供服务的子进程,则主体间的交互是无死锁的.

证明:略.

5 相关工作

主体交互不仅是多主体系统中单个主体实现自身目标的基本手段,也是设计和实现面向主体的程序设计语言的关键问题之一.在文献[4]中,将主体间的基本交互机制分为主体标识、异步消息传递、隐式接收原语等,主体通过基于言语动作(speech act)的原语来进行通信,并在此基础上定义了一种基于 Actor 模型的 Actor 代数,以研究言语动作与该代数之间的关系.由于该代数的目的在于为面向主体的程序设计语言以及主体通信语言提供语义基础,它所刻画的交互动作的粒度太小,这使得它不太适合于用来刻画大系统中较高层次的组合.本文在分析主体行为时,将主体间的交互重点放在服务请求与提供这一抽象级别上,并且选择了一种能够刻画主体的组合行为的 π 演算来作为形式化手段,这为在不同层次上以不同粒度来刻画多主体系统提供了较好的形式化基础.

在文献[5]中,将主体间的交互协议定义为由 4 层子协议构成的层次型协议,即注册层、服务层、任务层及通知层.其中每一层子协议都由请求方协议和提供方协议两部分构成,每一方的协议都是一个不确定的有限自动机.主体在交互过程中,其交互粒度与文献[4]相似.与本文将交互协议定义为多主体的公共行为规范所不同的是,它将交互协议作为构成主体的一部分,需要为每个主体中都定义一系列交互协议,这必然会导致主体定义及其行为的复杂化.本文将交互协议定义成一种公共行为规范,并能保证主体行为与该公共规范的一致性.这样,在定义主体时,只要主体行为遵循该公共规范,就不必考虑其他主体的存在,因而可以简化主体的定义.

6 结束语

在多主体系统中,主体为了实现自己的目标就不可避免地要与其他主体之间进行交互.鉴于主体交互的动态性,本文采用进程代数的方法对多主体系统中的交互行为及交互协议进行了研究.为了研究主体间的交互及其交互协议,本文从服务的请求及提供这一抽象层次上对参与交互的主体的行为进行了分类,并对其行为规范进行了形式化的描述.另外,我们把主体间的交互协议定义成多主体间应遵循的一种公共行为规范,这样,主体行为只要能遵循该公共行为规范,就能保证其目标顺利实现,而不必考虑其他主体是否存在,这充分体现了主体的自主性及独立性.

在本文中,还讨论了主体交互与交互协议之间的关系,并证明了主体的行为规范与交互协议的一致性以及主体交互的无死锁性,这保证了只要主体根据自己的行为规范与其他主体进行交互,最终就必然能实现自己的目标,从而保证了交互的有效性.

主体交互是研究多主体系统中主体行为的基础.在此基础上,我们将进一步研究主体间的协作

过程、协作协议以及与协作有关的一系列问题.

References:

[1] Jennings, N. R., Sycara, K., Wooldridge, M. A roadmap of agent research and development. International Journal of Autonomous Agents and Multi-Agent System, 1998,1(1):275~306.

[2] Milner, R. The polyadic π-calculus: a tutorial. In: Bauer, F. L., Brauer, W., Schwichttenberg, H., eds. Logic and Algebra of Specification. London: Springer-Verlag, 1993. 203~246.

[3] Fisher, M., Wooldridge, M. On the formal specification and verification of multi-agent systems. International Journal of Cooperative Information System, 1997,6(1):37~65.

[4] Dalmonte, A., Gaspari, M. Modeling interaction in agent system. UBLCS-95-7, Italy: University of Bologna: Laboratory for Computer Science, 1995.

[5] d'Inverno, M., Kinny, D., Luck, M. Interaction protocols in agents. In: Demazeau, Y., ed. Proceedings of the 3rd International Conference on Multi-Agent Systems. IEEE Computer Society, 1998. 112~119.

Formalizing Interactions and Interaction Protocols in Multi-Agent Systems*

JIAO Wen-pin, SHI Zhong-zhi

(*Institute of Computing Technology, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: shizz@ics.ict.ac.cn

http://ics.ict.ac.cn/

Abstract: In this paper, interactions and interaction protocols in multi-agent systems are investigated deeply and described formally in a process calculus, the π-calculus. To discuss interactions between agents, the behaviors of agents are cataloged first, and then their specifications are formalized. After that, interaction protocols between agents are defined using processes, and the consistency and deadlock-freedom of interactions are also analyzed in this paper.

Key words: multi-agent system; interaction; π-calculus

* Received May 10, 1999; accepted April 13, 2000

Supported by the National High Technology Development Program of China under Grant No. 863-306-ZT02-01-3