计算机科学 1996 Vol. 23 №. 2

35-40

Agent 研究*)

朱 冰 TP18
(北京大学计算机科学技术系 北京 100871)

摘 要 This paper introduces agent learning, multi-agent systems analysis, agent negotiation, multi-agent systems programming language etc.

关键词 Agent, Multi-agent systems, Object, Active object.

agent、多 agent 系统已成为分布式人工智能研究的一个十分活跃领域。agent 通常指在一定的环境下持续自主地运行的实体，agent 有两个特点，一是自主性，一是相互合作性。agent 作为独立的个体，能自学习，自增长，同时又可以和别的 agent 进行磋商，合作以完全任务。现在，agent 不仅仅是分析人工智能问题的高层概念，而且有可能成为一个可实现的概念。本文主要参考第13界人工智能国际会议论文集，简单介绍了 agent 学习、多 agent 系统分析、agent 磋商与合作以及多 agent 程序设计等方面的研究现况。最后，初步分析了被动对象、主动对象和 agent。

一、agent 学习

1.1 环境历史活动的学习[1]

agent 的行为与其环境关系很大，它与环境的紧耦合表现在三个方面：开发时比较注重与外部世界的交互作用；为特定的目标和环境设计特殊的 agent；agent 的模型重点在于 agent 的相互作用，而不是对其行为的规划。

agent 通过环境中的历史活动来学习时，涉及到"例程"和"产地"两个概念，前者是指 agent 重复发生的活动，开发例程可以免除规划的烦恼，一般可定义

*)本项目研究得到863国家高技术计划的资助

HPF_LOCAL 说明了单 PROCESSOR、单 NODE 的程序接口。

这样 HPF 就可以与前面多种 FORTRAN 版本、C、PASCAL、ADA 等中高级语言结合起来使用。

三、HPF 中 CHOLESKY 分解的并行程序

```
      PROGRAM CHOL
      PARAMETER N=1024
      REAL A(N,N)
!HPF $ PROCESSOR PROCS(4)
!HPF $ DISTRIBUTE A (CYLIC, * )ONTO
      PROCS
      READ * M
      CALL CHOLSKY(A,N,M)
      END
      SUBROTINE CHOLSKY(A,N,M)
      REAL A(N,N)
      DO K=1,M
        A(K,K)=SQRT(A(K,K))
        FORALL(I=(K+1):M)
          A(I,K)=A(I,K)/A(K,K)
        DO J=K+1,M
          FORALL(I=MAX(K+1,J):M)
          A(I,J)=A(I,J)- A(I,K) * A(J,K)
        END DO
      END DO
      RETURN
      END
```

参 考 文 献

[1] High Performance Fortran Forum (HPFF), High Performance Fortran Language Specification, May 3,1993

[2] Charles H. Koelbel 等, The High Performance Fortran Handbook, The MIT Press,1994

[3] Geoffrey Fox, Seema Hiranandan 等, Fortran D Language Specification, April 15,1991

[4] ISO/IEC JTC1, Fortran 90 Language, Information Technology,1991

[5] K. Dincer 等, High Performance Fortran and Possible Extensions to Support Conjugate Gradient Algorithms, NPAC Technical Report SCCS 703

一个环境中的公共活动为例程。产地是指"通常可以找到一个人或事物的地方"。也就是说，一个 agent 不是活动在抽象的环境中，而是活动在具体的环境中。一个单独的 agent 是 agent 团体的一部分。例程则根植于单独 agent 的产地中。一个例程会多次反映其活动产地的细节。通过对例程活动的揭示，agent 可以熟悉其具体环境，方便地获取所要的信息，将例程作为一个框架，来组织产地的有关特性，这些累积的信息反过来又丰富和扩大了例程。对例程活动的开发可通过分析 agent 在环境中的活动历史来完成。也即考查 agent 以往交互作用的活动，对活动的某些动作片断以插曲形式收集起来，并进行分段和排序。

1.2 多 agent 系统的学习

在多 agent 系统中，agent 是平等协作的，agent 学习的一个重要研究内容就是若干 agent 如何学会协同操作以共同完成给定的任务。

通常，假定一个 agent 由传感元件、制动元件、知识库和学习元件组成。多 agent 系统中，用 agent 集来代替单个 agent 能更好地将许多现实世界的问题模型化，能处理诸如单个 agent 能力不足的问题，还能处理由于被执行的数据受制于物理分布而产生的自然约束问题；并且作为一个分布系统有其固有的优点，如具有坚固性、容错能力、并行性和可裁剪性。

研究多 agent 系统的学习问题，能自动提高人工的 agent 系统(如机器人系统)的行为能力，并且学习过程可以加深对自然的多 agent 系统(如人群或社会)的理解。多 agent 系统中，有两种类型的学习方式：一是集中的独立式学习，如单个 agent 创建新的知识结构或通过实践来学习；二是分布的汇集式学习，如一组 agent 通过交换知识或观察别的 agent 行为的学习。多 agent 系统的学习一般研究的是后者，前者归于单个 agent 的学习中。

多 agent 系统中，存在两种约束：一是不相容性约束，即一个动作的执行会引起环境的改变，这就会影响，甚至阻止其它动作的执行。在这种情况下，不同的动作是不相容的；二是局部信息约束，即一个 agent 一般仅仅了解实际环境状况的局部信息，并且所了解的信息不同于另外的 agent。

下面介绍多 agent 系统汇集式学习的 ACE 算法(动作评估算法)[4]。其基本思想是：重复执行一个基本的工作循环，以实现多 agent 系统学会彼此友好合作的目标。这个工作循环包括动作确定、多 agent 竞争和信用赋值三个活动。

动作确定是指多 agent 系统的每个 agent 独自根据它对整个环境的局部认识来确定它能执行的动作集。

所有 agent 一起竞争以希望能执行自己提供的动作。每次竞争时，每个 agent 都对它的所有可能的动作进行标值演算，并把其标值向其它 agent 声明。根据 agent 宣布的标值，选择要执行的动作。允许有最佳标值的 agent 执行其动作，并在动作集中撤消该动作，撤消与该动作不相容的动作。循环进行这一竞争过程，直到所有具有非0投标值的 agent 都执行完其动作为止。

信用赋值主要是总结上两步中多 agent 系统的经验，指导多 agent 系统以后的活动。具体是调整 agent 动作相关性的估值。修改 agent 动作标值的演算式。信用赋值机制有两方面的效果，一是成功动作序列中动作的估值随时间的增加而增长，以形成稳定的序列；二是不成功序列中动作估值随时间增加而减少。这样，增加了成功的序列的稳定性，逐步淘汰不成功的动作。

二、多 agent 系统分析

基于 agent 的系统由多个有其状态特征的 agent 组成，每个 agent 可看成是具有某种心智状态的个体。agent 的心智态度能影响它的决策，确定它所要采取的行为，并可由信念、期望和意向三个概念来刻画。信念表示 agent 拥有的信息，期望指 agent 的可评估的状态，意向表示 agent 在以前类似情况下作出的决定，对 agent 以后的动作有指导作用。多 agent 系统有动态和资源受限两个特点。系统完成给定任务有两个时间阶段：商榷阶段和行为阶段。如果花费在商榷上的时间太多，就会影响系统完成其任务的能力。商榷时间太短又可能导致系统缺乏远见，并且由于交互作用过多而影响系统效率。

每个 agent 都可看作进程的并发系统，其若干进程的执行表现为每个进程的原子动作的不确定的交互重叠。不同于传统的并发，agent 自身能直接控制它的动作，但它不能控制环境，只能受环境影响并通过自身动作来无意中影响环境。要将 agent 对动作的选择和 agent 对环境的视图区分开来。每个 agent 有它自己的环境视图和对其它 agent 智能状态的视图。视图有可能既不与环境重合，又不与其它 agent 所拥有的智能状态一致。

分析和研究多 agent 系统的特性时，一般要将系统作为一个整体来考虑。整个系统的特性和其单个 agent 的局部特性的相关程度取决于特性本身和 agent 定义这些特性的方式、定义特性时用到的背景知识(如整个系统的行为上下文)越少，agent 个体特性与多 agent 系统的整体特性的相关性就越少。总之，分析系统特性时，要考虑整个系统的行为，要在

系统给定的背景下分析系统的各部分。也即对系统的某种特性的认识一般不能仅仅通过带这种性质的agent的相互作用获得，也不能仅仅通过分析单个agent来获取。

可用无死锁、活动性和公平性等特性来说明多agent系统特性和agent特性的关系，首先对有关特性进行粗略说明。无死锁指没有循环等待现象。活动性是指总有某一动作可以操作或总会发生某一事件，显然，具有活动性的必要条件是无死锁。公平性是对动作的合理调度能力，考虑到agent之间的关系时，要考查agent的全局无死锁、全局活动性和全局公平性。就agent自身，要考查agent的局部无死锁、局部活动性和局部公平性。

文[4]基于一个抽象语言来研究agent及多agent系统的无死锁、活动性和公平性，有如下结论：

如果一个多agent系统中至少有一个agent是全局无死锁的，则系统无死锁。但存在有某多agent系统，系统无死锁，而其所有的agent都不是全局无死锁的。

多agent系统具有活动性和公平性当且仅当它所有的agent具有全局活动性和全局公平性。

如果一个多agent系统具有活动性，则其所有的agent具有局部活动性。

存在有这样的多agent系统，系统无死锁，而所有的agent都不是局部无死锁的；系统具有公平性而所有的agent都不具有局部公平性。

存在有某多agent系统，系统不是无死锁的，而其所有agent都局部无死锁；系统不具有活动性而其所有agent都具有局部活动性；系统不具有公平性而其所有agent都具有局部公平性。

三、多agent的磋商

agent之间的磋商一般指合作前或合作中的通讯过程。多agent磋商的目的就是为了合作。对磋商机制和策略有如下的评估标准。其一，对称分布，即要求没有agent是特殊的，每个agent的地位都相同。其二，有效性。即通过磋商能实现有效的解，如满足Pareto-最优性，也就是所得解不会与其它解相冲突。其三，稳定性。即结果是平衡的，没有一个特别的agent因磋商获利，尽管agent群体会获利。其四、简单性。即指低通讯代价，低计算复杂度。下面介绍两种磋商机制。

1 将某领域和某磋商策略相连[5]

通过考查许多不同领域的若干磋商协议，并研究它们的性质，建立某个领域和某个或某些磋商协议的联系。agent通过对其所处领域的了解，根据领域与磋商协议的关系，找到并选择合适的磋商协议。

可借助TOD(面向任务的领域)的定义来研究某些领域的分类和描述。TOD是根据多agent相遇时的情景来定义的，从直观上讲，这些领域中，agent是可合作的，agent间的目标一致，没有矛盾。每个agent都欢迎其它agent的存在，因为它们相互的存在只会彼此有利。如负责传递信件的邮递领域就是一个TOD。

2 投票机制[6]

投票机制的出发点在于一群agent要进行合作时，一定要达到某种共识。假定一群agent处于状态S1，要达到某不确定的目标状态S2。每个agent都有其私自的目的，由于状态的变迁可能涉及群体的行为，如多agent间共享相同资源等，因而要求该群中所有的agent遵循一个联合规划，以达到共同同意的最终状态。一群agent的协作动作与一组规划相对应。可以使用动态迭代的搜索过程来形成一个群体共认的规划。在过程的每一步中，所有agent都为规划的下一个联合动作的确定进行投票。这样，递增地构造了一个联合规划。

四、多agent合作

多agent的合作可以是有共同目标的合作和没有共同目标的合作。

4.1 有共同目标的合作情况

4.1.1 开放的多agent环境中的合作。大型多agent系统可看作开放的分布式环境，其agent可能有不一致的、带偏见的世界视图。系统中，一个agent有时需要和其它的agent合作以构造复杂的规划，或完成它不能单独完成的大型组织任务。由于一组agent中，每个agent都可能有不正确的关于世界的信念和不完全的知识，也由于各agent能力各异，因而在多agent中构造一个合作规划很不容易。

开放的分布式环境中，服务、处理能力和计算元素的拓扑结构都在不断地改变。由于合同网的动态特性，开放的分布式模型中，常使用合同网类型组织模式。但是，开放的多agent系统中，agent的粒度和规划也在动态改变，并且，agent还可能有不同的类型，这些附加的复杂性使得多agent系统中应用合同网非常困难。

主要是合同网协议中存在有两个问题：任务分解和任务分配。当请求者分解任务时，任务固有的分解方式可能不适合于开放的分布式环境。它不知道当时可提供怎样的agent，也不知道潜在的被请求者有哪些技能，因为它们的技能是不断变化的。应答者为每个子任务选择一个agent，并将子任务分配给该

agent。没有一个 agent 通过自己的规划来获得一个子任务，就算采用子合同的形式，这种将一个子任务分配给一个 agent 的固定任务分配策略也会导致低效率，将合同网协议用于层次的多 agent 规划中时，问题更为严重。请求者想要某个 agent 去完成目标，但如果在开放的分布式环境中，请求者没有足够的知识，不能正确地分解一个复杂的目标，那么它就不能要求任何单个的 agent 去完成该目标。它的任务分配策略是失败的。因而，需要更可行的策略来选择被请求者。下面介绍一个合作规划系统[7]。

该系统中，需要帮助的 agent 动态地组织一个群体。首先，agent 发送一个信息给公告牌 agent，以宣布一个协议请求(REP)。读到 REP 的 agent 为该请求安排一个个体规划，该规划也可能不完整，再将规划发送给作为请求者的最初始的 agent。请求者再在潜在的被请求者之间调查合作的可能性。如果有可能合作，则请求者给被请求者发出合作建议和合作酬金。给被请求者的合作建议包括：对其它 agent 是很明显的障碍，而被建议的 agent 可能有能力解决；以及合作的 agent 可以操作的动作。该建议建立一个偏序模型以指示其它 agent 的动作。根据这些建议，还有 agent 个体的规划和信念，每个合作的 agent 通过推论构造一个可合作的规划。构造合作规划时，每个 agent 都要决定它应做的动作，别的 agent 应做的动作，以及与其它 agent 一起完成的动作。同时，每个 agent 都要考虑三个因素：为其它 agent 消除障碍；实现 agent 间的工作负载平衡；用较低的代价去获得较高的效率。如果得出的合作规划包含有未经证实的假设，则该规划是不完全的。可将该假设作为目标，由另一些 agent 集合来实现或达到该假设。

产生 agent 的个体规划的模型是基于多世界模型的。一个 agent 在时间 t 的信念被模型化为一阶公理系统，并称为一个世界。操作则表示从一个世界到另一个世界的转换。因而，一个规划可被看作连接几个世界的操作链。

系统运作的整体算法为：请求者给公告牌 agent 发协议请求(REP)；自由 agent(指当前无任务的 agent)请求公告牌 agent 提供一个 REP；公告牌 agent 将 REP 发送给请求的自由 agent；自由 agent 生成个体规划；如果自由 agent 有个体规划，则将规划发送给请求 agent；请求者调查合作可能性，形成合作建议；请求者将合作酬金发送给被请求者；被请求者形成合作计划。

4.1.2 成组 agent 的合作。规划和动作的合作是多 agent 合作问题求解的主要内容。许多多 agent 系统要求多个 agent 之间不仅要避免相互间的动作或状态的冲突，而且还要有一起规划和操作的能力。合作的规划和活动要考虑单个 agent 的规划和活动，但不能简单地根据单个 agent 的规划来分析，要综合考虑在一起合作的多个 agent 的信念和意图。

Pollack 的模型[8]中考虑了制定规划时 agent 的心智状态。一个 agent 在进行个体规划时有四种心智态度：信念1——操作动作 Bi 时要用到动作 A；信念2——该 agent 能操作每个 Bi；意向1——该 agent 做每个 Bi；意向2——根据 Bi 的操作来做 A。

Shared Plan 模型[8]允许多个 agent 一起形成一个规划去完成要求多个 agent 行为的某些动作。规划 agent 的心智态度时，参考了 Pollack 模型的定义，以集合的形式来描述 agent 进行规划时的心智态度，为可能的合作准备了信念和意向集的描述。

4.1.3 实时系统中 agent 的合作。实时系统的基本特点是尽可能简单、高效。为解决某些复杂的问题，除传统的处理方式外，还可以采用基于知识的 agent 合作处理方式。MARVEC 分布式系统是一个复杂的实时诊断系统[9]。该系统中，多 agent 可以合作地解决涉及多个领域的问题。其基本思路是：将问题划分到子区域，由单个 agent 尽可能地完全负责某子区域，这样，可以减少通讯量。系统中的 agent 是内嵌的诊断子，是由数据驱动的。每个 agent 限制在其领域中，即负责一个小的但明显分划的专业领域中。agent 的领域间不重叠；以避免冗余推理。agent 报告所有超出其领域的问题，失败领域不直接映射到 agent 领域。元知识被包含在每个知识库规则集中，提供 agent 之间合作的途径。agent 通过元知识寻找超出其领域的合作。agent 以层次组织形式来协调其动作。这样，多 agent 就在领域间分享了诊断的责任。

4.2 没有共同目标的合作情况

4.2.1 将某 agent 的子任务转交给别的 agent 完成。分布式问题求解要考虑的情况之一是 agent 之间的相互帮助，这体现在 agent 分享信息、共同承担任务。另外，即使没有公共的目标和全局的一致的知识，agent 也可能希望将自己的子任务分配给别的 agent，也就是说，一个 agent 为了完成其个体规划，将它自己做不了或别的 agent 能比它更好更有效地完成的子任务分配给其它的 agent。

问题在于 agent 间没有一个共同目标，而每个 agent 都不是那么慷慨大方的情况下，agent 如何使别的 agent 为它服务，并且要求被分配任务的 agent 能有效地工作，同时，分配任务的 agent 不用过多操心，而去做别的事件。

有两种方法[10]，一是通过特定的接口操作，二是允诺以酬金。在 agent 彼此不承担义务的非合作环境

下，一般采用后者。较典型地是通过签定合约来实现agent的任务转移。在这种情况下，agent的主要功能之一就是能签定一份合约，然后，包工agent付给合约agent报酬，合约agent为包工agent完成任务。所签定的合约必须简单，其算法可计算；该合约没有影响其它的合约安排；该合约具有稳定性，即合约能按期完成。签约的对象可以是单个的agent，也可以是一个agent小组。在确定的状态下签约的算法与非确定状态下签约的算法非常不同。

4.2.2 重复子规划的分派和接收。在产生agent和执行满足agent目标的多agent环境中，个体规划是独立产生的。这些规划间存在负向和正向两种关系。在负向关系中，规划要在同一时刻使用相同资源，这就可能发生冲突。对负向关系的研究主要是避免或解决冲突，使得动作能正确执行。这也就是有关临界区的管理，这类研究已有很多，并且较为成熟。在正向关系中，规划包括重叠的子规划，子规划由公共动作和单独动作组成。对正向关系的研究主要是考虑一个agent如何将其重叠子规划的执行部分分派给另一个agent，能降低系统的执行费用。正向关系又分为包含关系和互惠关系。在前者中，一个agent规划的动作与另一agent动作相同或包含有另一agent的动作。这样，若另外的agent执行了它自身的动作，则该agent无需再执行。在后者中，一个agent的规划与另一个agent的规划部分重叠，但不相同或包含。如果另外的agent愿意接收规划的重叠部分，并连接执行被接收规划的非重叠部分，则此agent无需再执行其规划的重叠部分。下面主要介绍对互惠的正向关系的研究[11]。

如前所述，多agent环境可看作开放的分布式环境，因此，agent的合作不是由系统规则强迫的，而是以自我为中心执行的。涉及互惠关系的合作原则一般是如果能获利则合作，不能获利就不合作。一个agent将它的动作派遣给其它agent，则该agent的执行费用能大大降低；而接收动作负责完成动作的agent的执行费用则增加了一些。如果agent能派遣动作而又不愿接收动作，那么很难进行合作；如果能保证这次接收动作的agent在将来某个时候也能派遣出它的某些动作，那么就可以进行合作。合作机制一般是基于某种社会法则，如担保，保证交易中某些预定的条件将会实现。这样，agent会平衡接收动作的费用。

在进行合作时，首先要了解agent进行规划和动作的情况。现假定：每个agent都有实现自己目标的规划，但不知道其它agent的规划；获得子目标的子规划由动作串组成；动作和相应费用是预定的，对每个agent都平等；agent能了解其规划中哪一子规划能发送；agent能发现可与之协商以派遣子规划的其它agent。其次要明确合作的法则。由于派遣会导致费用改变，派遣法则为：如果一个agent接收的动作超过另一agent，则在下一次协商时，它将不接收动作；如果一个agent派遣的动作超过另一个agent，则下一次协商时，它不能派遣动作。最后，给出合作算法：agent搜寻一个能派遣给别的agent的子规划；agent提出一个子规划来派遣，并搜寻能参与派遣协商活动的agent；agent进行策略决定，使得自己的偿还期望值最小；两方的agent将自己的决策都交给对方；两agent根据另一agent的决策来决定自己的行为，即如果决策不冲突，就根据决策来修改其规划，而如果决策冲突，就随机地决定哪个agent派遣和接收。

五、agent程序设计

agent程序设计是一种以计算的社会观为基础的新型程序设计范型，其核心是agent的设计和构造，强调agent的动态特征和agent之间的交互作用特点，agent程序设计可能涉及并发程序设计技术、逻辑程序设计技术和面向对象程序设计技术。

agent程序设计概念最初由Standford大学的一个研究小组于1990年提出，该小组以机器人设计为背景，采用agent程序设计的语义途径，强调agent几种心智状态的逻辑界定，并以此作为agent程序设计语言的约束。

Oz[12]也是一种agent程序设计语言，该语言综合了逻辑和并发程序设计的思想。从逻辑程序设计中，Oz继承了逻辑变量和逻辑数据结构，在这种设计风格中，关于变量的值的信息是并发和递增获得的。它容纳了高阶程序设计又没有舍弃一阶逻辑的记号和变量形式，Oz的另一特点就是约束通讯，即利用逻辑变量的异步通讯形式。并发逻辑程序设计采用的传统通讯机制是流通讯。约束通讯比流通讯灵活，且具有不确定性。一个通讯事件用连接通讯合作者的一个等式取代两个互补的通讯token，约束通讯引入了与逻辑数据结构完全等价的最小状态形式，并能有效实现公平通讯。Oz的语义由Oz演算这一数学模型来描述，Oz演算参考了Miller的π-演算的思想。基于约束通讯和高阶程序设计，Oz能支持包括多继承在内的多种面向对象程序设计风格。总之，该语言从语义出发，着重描述了agent之间的并发行为。

还有的agent程序设计采用语法途径[13]，以知识信息处理系统为背景，强调将知识的表示和处理

有机统一于 agent 这一框架中，以表示智能主体的知识状态和处理方法。这一 agent 程序设计语言的技术支点是面向对象程序设计技术、基于条件重写的逻辑程序设计技术以及元级知识处理技术。其 agent 的设计规范是：agent 由知识状态和知识处理方法两部分组成；agent 的知识状态被分割成若干一致的知识单元，是 agent 自身可以改变的部分；知识处理方法是 agent 自身不可改变的部分，agent 以传递消息的方式与外界建立联系。其知识单元的定义基于条件重写系统，具有描述计算过程和表达说明性知识的双重特点。

用 agent 程序设计语言设计的系统与传统的知识处理系统有很大的不同，主要表现在两个方面。其一，agent 中知识信息的表示结构和处理方法是一体的，设计 agent 时，要把它们作为一个整体来考虑，而传统的知识处理系统一般基于知识库，其知识和知识处理是分开的，分属两个不同的子系统。其二，agent 具有某种心智状态，强调其主观的特点，agent 具有自主的行为能力，而传统的知识是对现实世界的客观认识，力求与客观一致，它总是被动地被运用。

六、结束语

被动对象、主动对象和 agent 都属于对象的概念范畴。被动对象是由系统或别的对象来驱动的，主动对象则可以根据它所处的环境来决定它自身的行为，但主动对象所能具有的行为集是确定的，它没有磋商的能力，更不具有自学习和自增长的能力。

被动对象是一个广泛被认同与实现了的概念，是面向对象方法中对现实世界的静态的结构模拟，没有时间因素，只有记录其状态的存储空间，被动对象的操作虽然被封装，但对其操纵属全局控制。主动对象一般用状态转换器对它进行模型化，具有并发的特点，是对现实世界考虑了时间属性的结构模拟，从空间上讲，除有自己的存储空间外，还考虑了周围环境空间对自身行为的影响，另外，可以根据环境的变化来操纵自己，做到了控制的部分局部化。agent 具有并发和自主的特点，是对现实世界考虑了时间和空间属性的结构模拟，它有自己的时间概念，它有自己对环境的了解，有自己的心智态度，从某种意义上讲，agent 具有较强的自我控制的特点，比被动对象和主动对象具有更好的封装性和模块性。现在，对主动对象和 agent 的研究主要处于模型和原型阶段。特别是 agent，由于其丰富的内涵，尚未找到一个共认的中文译词。

从现实世界中的对象、认识世界中的对象和计算机世界中的对象的对应关系来看，计算机世界的被动对象、主动对象到 agent，对应于认识世界，是对对象概念理解的逐步深入，再对应于现实世界，是对其对象的更好模拟。

对 agent 的研究主要是在分布式人工智能领域展开的，在 AI 研究之初，由于硬件的限制，AI 研究的思路不可避免地为硬件条件所限，如知识处理系统或知识工程中知识库加推理机的模式就是冯·诺依曼体系结构的同构，其并发能力显然很低。如 R. Milner 所说，"在某种意义上讲，并发计算是我们的全部主题。"从工程学的角度看，AI 的成功与否应体现于其是否能构造出对社会和科学具有巨大影响的系统。如果 AI 不能很好解决并发问题，则其前景不容乐观。大规模并行计算机的出现必会导致基于高度并行人工智能的研究。而从这一研究方向上讲，agent 及多 agent 系统的研究是一个有益的尝试。

参 考 文 献

[1] R. Alterman et al., Agents, Habitats, and Routine Behavior

[2] Gerhard Weish, Learning to Coordinate Actions in Multi-Agent Systems

[3] A. S. Rao et al., A Model-Theoretic Approach to the Verification of Situated Reasoning Systems

[4] H. Burkhard, Liveness and Fairness Properties in Multi-Agent Systems

[5] Gilad Zlotkin et al., A Domain Theory for Task Oriented Negotiation

[6] Eithan Ephrati et al., Multi-Agent Planning as a Dynamic Search for Social consensus

[7] Ei-Ichi Osawa, A Scheme for Agent Collaboration in Open Multiagent Environments

[8] Barbara Grosz et al., Collaborative Plans for Group Activities

[9] S. J. Russell et al., Provably bounded optimal agents

[10] Sarit Kraus, Agents Contracting Tasks in Non-Collaborative Environments

[11] Kei Matsubayashi et al., A Collaboration Mechanism on Positive Interactions in Multi-agent Environments

[12] Martin Henz et al., Oz—— A Programming Language for Multi-Agent Systems

(以上参考文献出自：Proc. of the Thirteenth Intl. Join Conf. on Artificial Intelligence, Vol. 1 IJCAI-93)

[13] 王怀民、陈火旺、高洪奎，"面向智能主体的程序设计"，计算机学报，Vol. 17, No. 5, 1994