Agent Oriented Programming 进展*)

Advances in Agent Oriented Programming

王一川　石纯一

(清华大学计算机系　北京100084)

Abstract Agent-oriented programming (AOP) is a framework to develop agents, and it aims to link the gap between theory and practical in agent research. The core of an AOP framework is its language and semantics. In this paper, we propose the necessary properties which agents should have, and then give a summary and analysis about different AOP languages based on these properties.

Keywords Agent oriented programming, Multi-agent system, Agent language, Semantics

1 引言

近十年 Agent 和多 Agent 系统(MAS)的研究逐渐成为 AI 学科的热点之一。MAS 中 Agent 是具有思维状态和交互能力的自治实体,彼此通过合作求解复杂问题,适合于动态开放环境。Agent 的思维状态通常采用 BDI 模型,B(信念)、D(愿望)和 I(意图)分别用模态算子给出,并在可能世界框架下给出其语义解释。这一模型语义明确直观,但实质上是一个计算资源无限的理想模型,因而在实际系统中,都采用限制和变通的方法来实现 Agent,因此导致了所谓的理论脱离实践的问题。

正是在这种情形下,Shoham 提出了 Agent Oriented Programming (AOP)[1],试图以此作为从 Agent 理论模型到实际应用的关键一环。近年,研究者们在这方面已经展开了众多卓有成效的工作,包括 AGENT0[2], PLACA[3], AGENT-K[4]; Concurrent METATEM[5]; AgentSpeak(L)[6]; 3APL[7]和 GOAL[8];Golog[9]、ConGolog[10]和 IndiGolog[11]等。在这些语言中,比较有代表性的是3APL 和 ConGolog,其语义模型较为成熟,并提供了相应的工具以构造 Agent。

文[1]认为 AOP 框架由三部分组成:一个语法和语义明确的形式语言,其中各思维属性如信念、承诺等分别以模态算子表示;与语言对应的解释器;将待求解的问题"Agent 化"的方法。概括地说即 AOP 的语言定义、实现和工程化。目前 AOP 的研究工作主要集中在第一部分,解释器的实现属于技术问题,而将问题"Agent 化"则属于面向 Agent 的软件工程的研究内容。本文主要讨论 AOP 的第一部分。

2 Agent,MAS 与 AOP 语言

文[7]指出 BDI-Agent 应当具备的三个属性:A1)内部思维状态,包括信念、愿望、规划、意图和能力等;A2)预动性和反应性,前者指目标驱动,后者即 Agent 能够对环境的变化做出反应;A3)思考能力,即推理能力。这三个属性也是 Agent 与对象(Object)的本质不同之处。

MAS 中的 Agent 需要通过交互和合作来完成复杂任务,在此过程中,Agent 也要保持自治性,即不是无条件地相信和采纳从交互中得到的信念和任务。MAS 所求解的任务从分解后的子任务之间的依赖关系来看有三种情形,一是子任务互不相关;二是子任务有顺序相关性,比如子任务 b 必须在 a 完成之后才能开始;三是子任务有并发相关性,如子任务 a 和 b 必须由不同 Agent 同时进行。第一种情况最为简单,多 Agent 合作求解这类问题的目的只在于加快速度,后两种情况要求 Agent 所做的联合规划能够体现子任务的顺序相关性和并发相关性。因此,一个适用于 MAS 合作求解的 Agent,还应该具备属性:A4)交互能力;A5)联合规划能力,所做的规划能够体现子任务的顺序相关性和并发相关性;A6)自治性。

Agent 在运行时,根据当前目标集合和信念集合,确定和执行下一步的行动,并接受来自环境的反馈和其它 Agent 的消息,依此循环往复。需要注意的是 Agent 对其外部动作没有完全决定能力,即不能决定外部动作的效果。这样,Agent 还应具备属性:A7)行动和感知能力。行为与感知的不同在于,Agent 的行动通常改变外部状态,而感知只改变内部状态。

一个理想的 AOP 语言的语法和语义模型的设计,应当能够使所构造的 Agent 具备属性 A1—A7,同时还需保证语义模型的直观性。已有的 AOP 语言在不同程度上满足这些要求,但仍然没有一个完全符合这7点。

3 主要 AOP 语言

3.1 AGENT0、PLACA 和 AGENT-K

AGENT0[2]是第一个 AOP 语言,Agent 由信念、承诺、承诺规则和能力组成。Agent 的信念由其相信为真的事实组成,承诺由 Agent 决定在将来时刻所要做的动作组成,在 Agent 执行循环中持续变化;能力和承诺规则由设计者事先给定,能力规定了 Agent 动作的实施条件,承诺规则规定在不同思维状态下对不同消息的处理方法。Agent 的动作既有 INFORM,REQUEST 等交互动作,也包括用户自定义动作。在这一结构中,承诺规则是 Agent 运行核心,每个循环中 Agent 接收消息后,将其与承诺规则相匹配,匹配成功则承诺执行相应的动作(包括时间参数);同时,对于执行时间为当前时间的承诺动作,检查其实施条件,如果成立则执行。Agent 的交互分为 INFORM(事实)和 REQUEST(动作)两类,因此不能实现基于目标的交互。

*)本文受国家自然科学基金资助(69973023,60173011)。王一川　博士生,主要研究领域为多 Agent 系统。石纯一　教授,博士生导师,主要研究领域为人工智能应用基础。

AGENT0中 Agent 没有显式的规划处理能力，只能对动作进行承诺，因此委托 Agent 必须知道受托 Agent 的基本行为集合；对于 Agent 动作执行失败的情形也没有考虑。文[3]对 AGENT0进行扩展，给出了 PLACA(Planning Communicating Agents)语言，引入意图，同时使用规划生成器为意图产生相应的规划。当动作执行失败时，重新对相应意图进行规划。Agent 之间的交互也能以意图而不仅仅是动作为内容。

AGENT-K[4]目的在于标准化 AGENT0的消息传递功能，为此，在 AGENT0语法中结合了 KQML。AGENT-K 对 AGENT0所做的修改和扩充主要有：以单一的 kqml(time，message)动作取代了原有的 inform，request 和 unrequest 三个消息发送动作，其中 message 的格式为[performative，keyword(value)]，这里 performative 是依据 speech act 设定的交互原语；修改解释器以适应新的消息处理界面，每收到一条消息，都对所有的承诺规则进行匹配，即一条消息可以匹配多个规则以产生多个动作。

对于 AGENT0的语义模型，文[1]只是提出了所要遵循的性质：思维状态的内部一致性、忠实性(Agent 只承诺自己能够做的动作)、内省性(Agent 相信自己的承诺)和思维状态的持续性，而没有给出严格描述。因此可能导致相同的 Agent 程序在不同的解释器上有不同的表现。文[12]以与3APL 一致的语义内核给出了 Single Agent Core of AGENT0(SAC-AGENT0，其中去掉了 AGENT0中的交互部分)的语义。

3.2 Golog，ConGolog 和 IndiGolog

Golog[9]是一种定位于机器人和自治 Agent 的逻辑程序设计语言，它对情景演算的基本行为理论进行了扩充，引入了顺序、条件、循环和非确定性选择等语法要素，以便构造复杂程序。对于情景演算中的框架问题，采用 Reiter 的方法解决。Agent 的原子动作由用户通过情景演算的公理来定义，即对每一个原子动作，规定其前置条件公理和效果公理。对于特定目标，设计者只需给出规划的轮廓，由解释器根据当时世界状态和动作公理，搜索得到一个合法的动作系列。因为是根据轮廓生成具体规划，所以在效率上有所提高，同时也保持了灵活性。

Golog 只适用于刻画单 Agent。在 Golog 的基础上，ConGolog[10,13]引入含优先级的并发进程、高级中断以及外部动作的处理机制，并遵循交叠式并发语义。因为引入并发进程，所以其解释器的推理机制与 Golog 有很大不同，Golog 的动作序列采用单进程搜索，如果对于某个动作，不满足该动作的前置条件公理，则引起回溯；而 ConGolog 采用阻塞机制，在此情形下挂起该搜索进程，而运行其它进程。如果所有进程都被挂起，同时目标没有实现，才进行回溯。与 Golog 一样，ConGolog 解释器只有找到了一个完成目标的动作序列(规划)，各 Agent 才真正开始动作。因为初始状态的信息可能不完全，或者某些信息只能在运行中得到，所以这种 off-line 方式的搜索很可能找不到能解决目标的规划。同时，如果程序规模较大，则在执行前的规划搜索可能需要较多时间，而在这一过程中很可能环境就已经发生了变化，因此这种方法不适用于动态开放环境。

针对 ConGolog 的问题，IndiGolog[11]引入了感知动作(sensing action)，其解释器采用 on-line 规划方式和增量执行机制，在这一方式下，执行动作前不需要作出完整规划，动作一旦执行，便无法回溯。因此，IndiGolog 也引入重规划(replan)机制来处理动作序列执行过程中出现的变化。即便如此，在完整规划前尽早执行动作，一旦选择错误，仍然会导致一些问题。

除了 ConGolog 和 IndiGolog，Golog 系列的语言还包括 cc-Golog[14]，Sequential Temporal Golog[15]等。在 Golog 基础上的这一系列语言，不能够直接表示信念等思维状态。但从本质上来说，仍然属于 AOP 语言。由于 Agent 针对特定目标搜索动作序列的过程依赖于动作公理以及当前世界状态，因此隐含采用了知道逻辑，而非信念逻辑作为基本思维属性。

3.3 3APL 与 GOAL

3APL(An Abstract Agent Programming Language)[7,16]是一种基于规则的语言，结合了逻辑语言和命令语言的特点。3APL Agent 由四元组$\langle T,\Pi_0,\sigma_0,\Gamma\rangle$表示，T 是转换函数：$B\times B\rightarrow\delta(A)$，其中 B 表示信念集合，A 是原子动作的集合，$\Pi_0$和$\sigma_0$分别是初始信念和初始目标，$\Gamma$ 为推理规则集合，每一条规则给出了特定条件下 Agent 实现某目标所要采取的动作。Agent 实际上是一个基于规则的计算实体，即规则是 Agent 状态转换的核心。3APL 中存在四类规则：失败处理规则，反应规则，规划规则和优化规则。失败处理规则在目标不能实现时取消目标或修改规划；反应规则与 Agent 的当前目标无关，指定 Agent 在特定状态下的行为；规划规则定义实现命题目标的规划，优化规则指定规划在特定状态下 Agent 的优化规划。3APL 的操作语义用加标转换系统以结构化的方式给出。

为了给由3APL 编写的 Agent 所组成的系统提供规范化和验证手段，文[17]给出了3APL 的一个子集的指称语义和相应的程序逻辑，该逻辑是动态逻辑的一个变种，其中包含对 Agent 的动作和信念进行推理的算子。文[17]同时也证明了所给的指称语义与操作语义的等价性，并指出其程序逻辑适合于证明3APL Agent 的部分正确属性，即任何可终止的程序的终止状态。

3APL 中没有与 Agent 交互相对应的基本动作，因此不能用于 MAS，为此，文[18]对3APL 做了扩充，定义了两对同步通讯原语 tell/ask 和 req/offer 并给出相应的其操作语义。tell/ask 用于在 Agent 之间传递信息/询问，req/offer 用于传递请求/解释。其中解释给出的是所请求的问题的原因，因此对于已经成立的事实，所给的是事实成立的理由，而对于未成立的情况，给出的则是使之成立的规划。根据这两对原语的语义，存在两个问题，首先是 tell/ask(req/offer)的同步，这一点不符合直觉，通常应当是提问 Agent 先给出问题，然后再由回答方解答，而同步方式则相当于规定了过于严格的协议，双方对于所要谈论的问题，必须有相对应的顺序，否则交互无法进行；另一个更为根本的问题是 tell 和 offer 方并不需要保证提供的信息和解释是真实的(即自己不必相信)，而 ask 和 req 方对于得到的信息和解释，只要其确实解答了问题，则不论真假都将采纳，这一点对于自治 Agent 而言是不可接受的。

在3APL 中，目标是 goal-to-do 类型，即规划(动作序列)相关的，因此使用动作模态逻辑对其进行推理。但在 BDI Agent 中，目标属于 goal-to-be 类型，即所谓的 declarative goal，使用目标模态逻辑。因此，3APL 的语义模型与 BDI 模型存在较大差距，在定义和验证 Agent 程序时也不能使用目标模态逻辑。文[8]给出了一个面向目标的 Agent 语言 GOAL(Goal-Oriented Agent Language)，将目标定义为需要实现的状态，将信念和目标纳入到统一的框架中，并给出承诺策略以约束信念和目标：Agent 以 φ 为目标的前提是不相信当前状态下 φ 成立，同时$\rightarrow\varphi$ 不是必然的。GOAL 使用 Condi-

tional Action 来定义给定思维状态下 Agent 的行为。文[8]同时给出了 GOAL 的操作语义，并定义了相应的时态逻辑作为证明 GOAL Agent 属性的工具。

3.4 其它 AOP 语言

Rao 设计 AgentSpeak(L)[6]的出发点是以实际系统为背景(PRS 和 dMARS)，给出一个描述语言，并使其操作语义与该系统的具体实现一致。AgentSpeak(L)在一阶语言基础上定义了信念、目标、事件和动作。Agent 是事件驱动的计算实体，事件包括信念/目标的增加和删除，其运行核心是规则集合，每一条规则规定了事件发生时指定情形下应做的动作或产生的子目标。文[6]也给出了 AgentSpeak(L)的操作语义。直观地说，Agent 在每一运行循环中，首先响应事件，可能有多条规则对应于该事件，从中选出与当前状态一致的那些，如果仍多于一个，则随机选择，所得的结果即为意图，解释器从当前意图集合中选取其一执行。

Concurrent METATEM[5]是一种基于线性时态逻辑的 Agent 语言。在 Concurrent METATEM 系统中，Agent 用时态公式(时态规则)表示，解释器直接执行这些时态公式，Agent 之间通过异步广播方式进行信息交互。这样，Agent 的逻辑模型和执行模型是一致的，只要能够证明 Agent 公式符合某些属性，就能断定 Agent 在运行时将表现出这些性质。文[5]也给出了 ConCurrent METATEM 的操作模型，文[19]给出了其基于知道逻辑的语义。Concurrent METATEM 中动作是作为谓词求值的附加作用结果的，这一点并不直观，由于缺乏对动作的直接描述，难以简洁地表示规划。

此外，还有一些其它的 Agent 语言，例如旨在包装传统代码以使之 Agent 化的 Temporal Agent Programm[20]，它基于时态逻辑，与 Concurrent METATEM 较为接近；着重于 Agent 交互而给出的 April[21]及其后继者 MAIL[22]；在并发的面向对象语言 CUBL 基础上的 DAISY[23]等。

4 比较和展望

从语义模型的角度，文[24]以可观察行为来定义 Agent 的模拟和互模拟，通过将 AgentSpeak(L)Agent 转换为相应的3APL Agent，并证明后者模拟前者，从而证明在保证 Agent 结构一致性的前提下，3APL 的表达能力不弱于 AgentSpeak(L)。文[25]证明了3APL 的表达能力等价于 ConGolog。而文[12]依据于3APL 一致的语义模型给出 SAC-AGENT0 的操作语义，也说明了3APL 不弱于 SAC-AGENT0。

我们从 AOP 语言所构造的 Agent 能够具有的属性的角度，以 Agent 的 A1—A7七条属性为依据来比较 PLACA、3APL 和 IndiGolog，这三种语言分别在各自系列中最有代表性。表1列出了比较结果。

表1 PLACA，3APL 和 IndiGolog 的比较

	PLACA	3APL	IndiGolog
思维能力	支持	支持	隐含
预动性和反应性	支持	支持	支持
推理能力	支持	支持	支持
交互能力	支持	较差	不需交互
联合规划能力	间接支持	只支持顺序规划	隐含顺序规划
自治性	支持	较差	只适用于合作求解
行动和感知能力	只支持动作	只支持动作	支持

思维能力、预动性和反应性、推理能力和行动能力是每个 AOP 语言所构造的 Agent 都具有的，PLACA 和3APL Agent 是规则驱动，显式表示思维属性，而 IndiGolog 则是目标驱动，在规划轮廓的基础上由解释器给出具体规划，隐含实现思维状态。Agent 的交互能力通常决定了其自治性，如前所述，3APL 对 Agent 交互的支持较差，而 IndiGolog 因为由解释器在全局范围内搜索规划，Agent 之间并不需要交互，因此，这两种语言所支持的 Agent，只能适用于合作求解。对于机器人足球赛这样的对抗场景，IndiGolog 的最好处理方法是将两球队看作不同的两个 MAS 系统，队员的动作对另一支队而言都是外部动作；而3APL 则需要重新定义交互动作，才能支持。对于联合规划能力，PLACA 因为在请求中包含时间因素，所以间接支持了并发相关的规划。

总的看来，PLACA 似乎要优于另两种语言，原因是它没有明确的语义，因此实现者可以采用一些方法来避免如自治性冲突等问题。但缺乏明确语义会带来其它的问题。

对 AOP 语言而言，最重要的是语义模型，3APL 在交互性、自治性和联合规划能力上的欠缺，根本原因在于最初所采用的语义模型并没有这方面的考虑。文[26]给出了一个新的真并发语义模型，有望成为3APL 或该研究团体将来设计的语言的核心，但这一模型并未提供真并发带来的动作冲突等问题的解决方法，可能会给语言和解释器设计带来障碍。

结语 近年来关于 BDI Agent 思维状态模型方面的研究的一个重要方面是 Agent 的社会性，如联合意图、联合信念和联合规划等，以及在此基础上的合作、联盟、组织、规范等。从这一点来看，AOP 语言的设计也需要从 Agent 社会性的角度出发，推理能力和社会能力并重，才能对 MAS 及其应用提供切实的支持。

参考文献

1 Shoham Y. Agent-oriented programming. Articial Intelligence, 1993, 60(1):51～92

2 Shoham Y. AGENT0: A simple agent language and its interpreter. In: Proc. of the Ninth National Conf. on Articial Intelligence (AAAI-91), Anaheim, CA, 1991. 704～709

3 Thomas S R. The PLACA agent programming language. In M. J. Wooldridge and N. R. Jennings, eds. Intelligent Agents, LNAI 890, 355～370

4 Davies, Winton H E, Edwards P. Agent-K : An Integration of AOP and KQML: [King's College Technical Report AUCS/TR9406]. 1994

5 Fisher M. A survey of Concurrent METATEM --- the language and its applications. In: D. M. Gabbay, H. J. Ohlbach, eds. Temporal Logic - Proc. of the First Intl. Conf. LNAI 827, 480～505

6 Rao A S. AgentSpeak(L): BDI agents speak out in a logical computable language. In: W. Van de Velde and J. W. Perram, eds. Agents Breaking Away: Proc. of the 7 th 24 European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038, 42～55

7 Hindriks K V, et al. Formal Semantics for an Abstract Agent Programming Language. In: M. P. Singh, A. S. Rao, and M. J. Wooldridge, eds. Intelligent Agents IV, LNAI 1365, 215～229

8 Hindriks K V, et al. Agent Programming with Declarative Goals. ATAL 2000, LNAI 1986. 228～243

9 Levesque H, et al. GOLOG: A Logic Programming Language for Dynamic Domains. The Journal of Logic Programming, 1997, 31: 59～83

(下转第97页)

生命周期 TTL_{a_i}，则把 a_i 的地址信息 l_{a_i} 加入 t 时刻的更新列表 L_t；否则不加入。这里，TTL_{a_i} 就是访问频率 F_{a_i} 的倒数（即：$TTL_{a_i}=1/F_{a_i}$）。

如图3，Q_p 是要发送给原始服务器的预取列表队列。预取队列的优先级低于请求队列（Q_r），缓存首先满足来自客户端浏览器的请求，然后预取队列可以在无请求队列的时候发送或网络不拥挤的时候穿插在请求队列中发送。

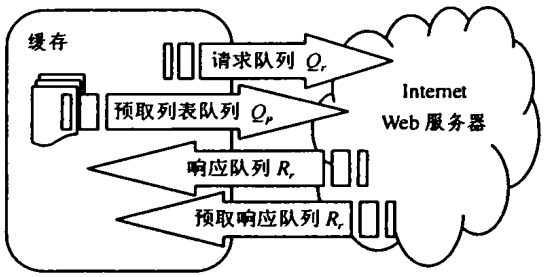

图3 请求队列、预取队列

把响应队列返回的页面与缓存中的同一页面进行差异比较，如果差异部分的比例大大超过了以往改动的平均水平，我们就认为生成该页面的程序或后台数据库发生了改变，需要刷新基对象 O_T，而不缓存差分对象 $O_{T+\Delta t}$。同时，记下改动的时间，并根据基对象改动的时间，计算该对象的辅助生命周期 TTL'_{a_i}（或者叫后台生命周期）。很明显，TTL_{a_i} 是客户端用户对页面 a_i 的期望周期，而 TTL'_{a_i} 则是一个测量值，同样也不能反映真实的改动周期，所以我们取两者的最小值近似地作为页面的改动周期以同步原页面信息的变化。

3 实验与分析

我们在实验室环境下实现了该动态 Web 页面缓存技术的原型系统。实验中我们发现，由于缓存中不仅存放了原始 Web 服务器的页面，同时还存放了后台数据库的数据，因此存储量比一般的静态缓存系统要大很多。为此，我们考虑对于那些动态部分占全部内容的比例70%～90%以上的动态内容（比如搜索引擎返回的页面），我们可以对其压缩以进一步减少存储量。

在某些情况下由于没有足够的学习信息，对于第一次新发现的后缀，系统可能会把不认识的动态页面理解成静态页面，但是在一致性策略范围内是没关系的。

由于某些页面（如 BBS 等）刷新频率很高，因此我们考虑当 TTL 小于某个阈值时就不作缓存，以提高处理速度。

另外，出于安全考虑，在实际系统中对于安全套接字协议层（Secure Sockets Layer，SSL）的对象将不进行缓存。

参 考 文 献

1 Cao P，Zhang J，Beach P B. Active Cache：Caching Dynamic Contents on the Web. In：Proc. Middleware '98 Conf. 1988

2 王克宏，汤志忠，胡蓬，等. 知识工程与知识处理系统. 北京：清华大学出版社，1994

3 Heckel P. A Technique for Isolating Differences Between Files. Commun. ACM，1978，21：264～268

4 Arlitte M，et al. Evaluating Content Management Techniques for Web Proxy Caches. In：Proc. of the 2nd Workshop on Internet Server Performance (WISP '99)，Atlanta GA，May，1999

（上接第122页）

10 Lespérance Y，et al. Foundations of a Logical Approach to Agent Programming. In：M. Wooldridge，J. P. Müller，and M. Tambe，eds. Intelligent Agents II-ATAL95，LNAI 1037. 331～346

11 Ng H-K. Topics in High-Level Robot Control：Integrating Planning and Reactivity，and Multiple-Robot Control：[M. Sc. Thesis]. Dept. of Computer Science，York University，2001

12 Hindriks K V，Lesperance Y，Levesque H. An Operational Semantics for the Single Agent Core of AGENT0：[Technical Report UU-CS-1999-30]. Department of Computer Science，Universitry Utrecht，1999

13 Giacomo G D，Lespérance Y，Levesque H J. ConGolog，a concurrent programming language based on the situation calculus. Artificial Intelligence，2000，121：109～169

14 Grosskreutz H，cc-Golog G L. Towards More Realistic Logic-Based Robot Controllers. Proc. AAAI/IAAI 2000. 476～482

15 Reiter R. Sequential，Temporal GOLOG. In：Proc KR 1998. 547～556

16 Hindriks K V，et al. Agent Programming in 3APL. In：Jennings N，Sycara K，Georgeff M，eds. Autonomous Agents and Multi-Agent Systems，2(4)：357～401

17 Hindriks K V，et al. A Programming Logic for Part of the Agent Language 3APL. In：Rash J L，et al，eds. Procs Formal Approaches to Agent-Based Systems，FAABS 2000，LNAI 1871. 78～89

18 Hindriks K V，et al. Semantics of Communicating Agents Based on Deduction and Abduction. In：Dignum F，Chaib-draa B，eds. Proc. of the Workshop on Agent Communication Languages. 105～118

19 Wooldridge M J. A Knowledge-Theoretic Semantics for Concurrent METATEM. In：J. P. Muller，M. J. Wooldridge，N. R. Jennings，eds. Intelligent Agents III，LNAI 1193. 357～374

20 Dix J，Kraus S，Subrahmanian V S. Temporal Agent Programs. Artificial Intelligence，2001，127(1)：87～135

21 McCABE F G，CLARK K L. April-Agent Process Interaction Language. In：Wooldridge，M J，Jennings N R，eds. Intelligent Agents，LNAI 890，324～340

22 Haugeneder H，Steiner D，McCabe F G. IMAGINE：A Framework for building Multi-agent Systems. In：Deen S M，ed. Proc. of the 1994 Intl. Working Conf. on Cooperating Knowledge Based Systems (CKBS-94)，1994. 31～64

23 Poggi A. DAISY：An object-oriented system for distributed articial intelligence. In：Wooldridge M J，Jennings N R，eds. Intelligent Agents，LNAI 890，341～354

24 Hindriks K V，et al. A Formal Embedding of AgentSpeak(L) in 3APL. In：Antoniou G，Slaney J，eds. Advanced Topics in Artificial Intelligence，LNAI 1502，155～166

25 Hindriks K V，Lesperance Y，Levesque H. An Embedding of ConGolog in 3APL：[Technical Report UU-CS-2000-13]. Department of Computer Science，Universitry Utrecht，2000

26 de Vries W，et al. A Truly Concurrent Model for Interacting Agents. In：Intelligent Agents：Specification，Modeling，and Applications，4th Pacific Rim Intl. Workshop on Multi-Agents (PRIMA 2001)，LNAI 2132，16～30