计算机科学1997Vol.24№.1

DAI中面向对象的并发程序设计

黄小虎 徐晓燕 郑南宁

(西安交通大学人工智能与机器人研究所 西安 710049)(西安交通大学系统工程研究所)

摘 要 Object-Oriented Concurrent Programming (OOCP) is a programming and design methodology in which the system to be constructed is modeled as a collection of concurrently executable program modules, called objects, that interact with one another by sending message. One of the major application areas of OOCP is Distributed Artificial Intelligence (DAI), which is concerned with cooperative problem solving by a decentralized and loosely coupled collection of knowledge sources. This paper presents two OOCP languages, AGENTS and ABCDL, which are used in DAI.

关键词 Object-oriented, Concurrent programming, Distributed artificial intelligence, Cooperating knowledge-based systems, Distributed problem solving

1 引言

对象具有一定的独立性且具有统一的通信协议,所以将系统分解成为并行运行的对象的集合,既提供了对象间合作的可能性又增加了系统组合的灵活性。因此,面向对象的并发程序设计(OOCP)[1]语言成为开发并行性的有力编程工具,OOCP的一个重要的应用领域是分布式人工智能(DAI)。单一的基于知识的智能专家系统的应用领域有限,但将简单的专家系统扩展到合作系统领域中使用,构成分布式的人工智能系统,就能发挥专家系统的潜力,使通常处理某类特殊问题的子系统协作解决原来单一子系统不能解决的复杂问题。但是不同的系统可能使用不同的专用语言,这些语言不兼容或难于集成,为了进行有效的合作就必须开发一种通用的形式化的语言,使各子系统之间可以相互通信。

分布式人工智能研究的是松散耦合的一组知识源相互合作求解问题的理论和方法[2],而OOCP中的对象就可以看成是具有一定计算和通信能力的知识源。在这里对DAI中通用的通信语言的研究就可转化为对OOCP语言设计的研究了。目前,已有一些基于对象的并发语言应用于分布式人工智能领域,较为成熟的OOCP系统有Act 1、Actor模型、ABCL/1、ConcurrentSmalltalk、Orient84/K和POOL-T等。本文介绍的是两个新近出现的面向对象的并发程序设计语言AGENTS[3]和ABCDL[4],以及它们在分布式人工智能中的应用。

2 AGENTS

AGENTS是多个专家系统进行合作工业设计的支持工具,结合面向对象和演绎两种系统,采用了类Prolog语言的语法和语义,这种专用语言具有面向对象语言的特征,如具有消息传递、继承性、属性、方法和约束等概念。在分布式人工智能系统中引入面向对象的结构增加了系统的模块化和通用性,同时也能很好地与普通Prolog系统在语法和语义上相互兼容,可利用Prolog为专家系统快速建模的特点。

在AGENTS系统中,知识源是由两部分组成的,即结点(agent)和系统。系统是主要为计算设计的专用证明语言;结点可以表示知识源,知识源之间的相互作用是通过结点合作实现的,且结点使用通用语言以便结点之间相互理解。

2.1 Prolog语言和面向对象的系统

用Prolog描述的推理系统是由子句集组成的:

黄小虎 博士生,研究领域:计算机并行图象处理,并行与分布式系统。徐晓燕 博士生,研究领域:分布式人工智能,多专家系统,联结的符号系统。郑南宁 博士,教授,博士生导师,研究领域:计算机并行图象处理,机器视觉系统,分布式传感系统,人工智能。

$$P_1 \wedge P_2 \wedge \cdots \wedge P_k \Rightarrow R \quad (1)$$

P_i 表示条件，R 表示结论，k 是条件数($k \geqslant 0$)。当 $k \geqslant 1$ 时，(1)式表示规则；当 $k=1$ 时，(1)式表示事实。Prolog 语言具有陈述性、基于规则的反向推理、约束发现、自动回溯、演绎和模式匹配等功能，很适于处理对象和对象之间的关系。

Prolog 的缺点是要求在不同的模块中不能有重复的谓词出现，不可预料的谓词重复会导致错误解释。另外它的数据抽象能力较差，事实和规则被简单地汇集在一起，没有用模块化的知识来描述复杂的概念和对象，因此难以构造层次和网络结构。若在其中加入面向对象的结构，这些缺点可加以弥补。

面向对象的系统具有一些优良特性，如高度模块化使得结点成为黑箱，彼此之间不需知道对方的内部结构和方法；为系统快速建模的特性使得在系统设计中，除了定义继承关系以外，无需详细规划和定义对象之间的连接关系。

2.2 面向对象的 Prolog

面向对象的 Prolog 系统与现有 Prolog 系统在语法和语义上相互兼容。系统由结点组成，每个结点是建立在子句基础上的普通 Prolog 系统。结点如何与外界相联系是由方法(Methods)决定的。消息传递是结点间通信的基本机制。子句是结点的基本元素，且对于结点是局部和专用的，但不同结点的子句表达方法可以一样。

·Prolog 中的查询。在 Prolog 程序中用户可查询：?-grandfather(smith,Who)。此问题可由式(2)的规则回答。

grandfather(X,Z):-father(X,Y),father(Y,Z). (2)

这条规则将继续查询谁是 smith 的父亲和谁是 smith 的父亲的父亲(见图1)。

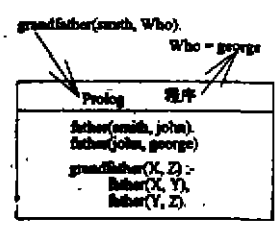

图 1 普通Prolog 程序的查询实现

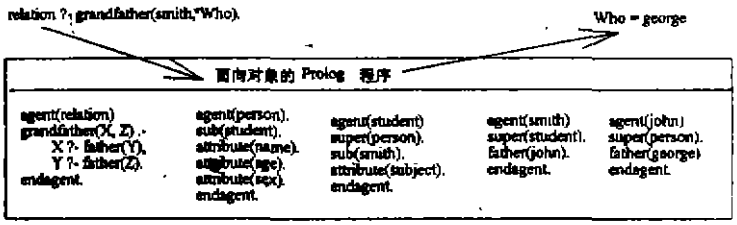

图 2 AGENTS程序对查询的实现

·面向对象的 Prolog 中的查询。在面向对象的 Prolog 程序中，对某一假设真假的询问是直接向结点提出的(见图2)。例如对于与上例相同的查询，问题是向"关系"结点提出的。在家庭关系结点中有一条关于"祖父"关系的规则，向关系结点提问以及各结点之间的提问都是以消息传递的方式进行。询问的形式是：Receiver?-Goal.。Receiver 是一个结点，Goal 是来自用户或其它消息的问题。图2中 'sub()' 是子结点，'super()' 是父结点。

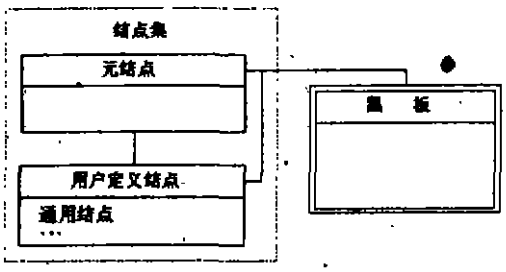

图3 AGENTS结构

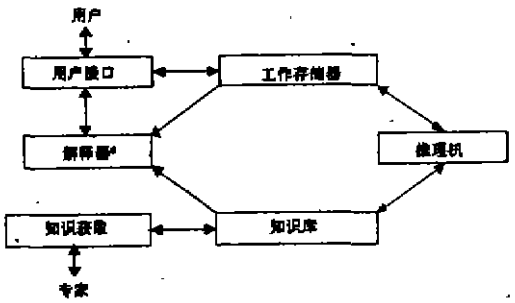

图4 结点的结构

2.3 AGENTS 的体系结构

AGENTS 系统结合了面向对象和逻辑推理两者对事物的表示方法。AGENTS 的结构主要包括两部分：结点集和黑板(见图3)。结点通过在黑板上读写来交流信息和讨论问题。有一类内部结点称为元结点(meta agent)，它提供一些标准的 Prolog 谓词和扩展的面向对象的结构。其它所有结点都可以继承元结点的特性。

问题的解是合作结点通过黑板不断讨论得出的。黑板的功能有:1)描述问题求解的全局状态;2)记录中间结果的依赖关系;3)真值保持;4)通信和控制。

结点的体系结构是标准的知识库专家系统(图4),以 Prolog 作为知识库编程外壳。图中各模块的功能是:1)知识库:知识库包括事实和规则。2)工作存储器:也称为工作环境(context),存储的内容为当前问题求解的状态描述。3)推理机:通过知识表示的形式化和控制推理的策略实现的,与领域知识无关的推理机构。4)用户接口和解释器:人机交互可以通过解释器实现。5)知识获取:知识获取机制保证知识库中存有最新的数据信息,从而使建立在其上的决策是正确的。

AGENTS 系统将面向对象系统和演绎系统结合。AGENTS 的结点具有知识库系统的标准组成,每个结点也可看成是一个独立的知识库专家系统。结点有一个通用环境以便合作。

3 ABCDL

DAI系统要解决的主要技术问题有三点:子任务分布、控制的分布和结点间通信。分布的面向对象的语言适于描述和实现 DAI 系统,因为它具有数据和过程抽象的能力,并且可以在对象(结点)之间通过消息传递进行通信,实现合作,从而满足了 DAI 的上述技术要求。ABCL/1模型对 Actor 模型作了改进,增加了丰富的同步机制,并引入大粒度过程以减少作用体的产生量。

这里要讨论的 ABCDL (Actor-based Concurrent Distributed Language)系统基于 Actor 模型,它由三个实体组成:顺序作用体(sequential actor)、分布式作用体(distributed actor)和通道管理器(channel manager)。ABCDL 语言是描述分布式系统的基于消息的离散事件仿真工具。

3.1 语言简介

ABCDL 的三种实体都是对象并且具有面向对象方法的所有特点,采用点对点的消息传递通信。

顺序作用体是一个计算实体,对输入数据处理,并将得到的结果通过一定的通道发送出去。它的描述式为:$sa=(P,D,f_{code})$。其中 P=输入参数集;D=输出数据集;f_{code}:P→D=输入参数到输出参数的映射关系。

通道管理器管理结点之间的通信。通道是一个同步实体,它从输入作用体接收消息,向输出作用体发送消息。它并不是一接收到消息就发送,而是依据消息的内容决定何时发送。它的描述式为:$ch=(M_{ic},f_{cond},M_{co},f_{cync})$,其中 M_{ic}=通道的输入作用体发出的消息集;$f_{cond}=M_{ic}$数据必须证实的激活条件;M_{co}=通道发给输出作用体的消息集;f_{cync}:$M_{ic}\times f_{cond}\rightarrow M_{co}$=输入参数(由输入消息给出)到输出数据(由输出消息给出)的映射关系。

分布作用体是一个计算实体,它的任务分布到子部分(其它作用体)集合中去完成,并由通道进行同步。分布作用体直接管理其唯一的外部接口。它的描述式为:$da=(P,A,C,D,f_{inter})$,其中 P=输入参数集;A=内部作用体集;C=内部通道集;D=输出数据集;f_{inter}:$P\times C\times A\rightarrow D$=输入参数到输出参数的映射关系。

分布作用体是一个复合对象,它可以树形结构组织一组对象。树的叶子结点是通道或顺序作用体;而其它结点(根结点和中间结点)描述其子结点及其相互连接关系,是分布作用体,它们从不同的细节层次描述整体问题。顺序作用体的性质可以改变,即可变为分布作用体,使系统具有开放性、可重构性和增长进化的特性。

3.2 用 ABCDL 仿真 DAI 系统

3.2.1 仿真步骤

ABCDL 是基于消息传递的离散事件仿真语言,它使用一个新实体描述时间,即世界时钟(world clock),它的定义如下:$wc=(A,t_{clock},f_{clock})$,其中 A=作用体集合;$t_{clock}$=当前时间;$f_{clock}$:$t_{clock}\times A\rightarrow t_{clock}$=作用在所有作用体上的时钟函数。

ABCDL 的仿真步骤为:

·用户向外部作用体发送消息;如果外部作用体是顺序作用体 SA,运行 SA 的程序代码,然后返回结果;如果外部作用体是分布作用体 DA,发送消息到 DA 的输入通道。

·通道 CM 收到消息时,当且仅当条件 f_{cond} 满足时,通道将消息发送给 CM 的输出作用体。

·当一个输出作用体 OA 从一个通道 CM 接收到一条消息时:如果 OA 是顺序作用体,则 OA 执行代码 f'_{code},并且当运行结束时($t_{clock}=t_{code}$)向 OA 的所有输出通道发送消息;如果 OA 是外部分布作用体,CM 在 OA 的内部,则 OA 将它的输入消息作为结果直接发送出去;如果 OA 是分布作用体并且持有通道,CM 在 OA 的内部,那么 OA 将消息发送给它的输出通道;如果 OA 是分布作用体,CM 在 OA 的外部,则 OA 将输入消息发送给它的内部通道。

·当所有顺序作用体的状态都是不活动的时候，仿真结束。

3.2.2 作用体驱动机制

顺序作用体的驱动基于函数调用，分布作用体的驱动基于消息传递和活动值。活动值是面向过程编程的基本计算机制。活动值将方法与槽相联系，当值从某个槽中取出或存入时，相应的读/写方法将根据槽值决定运行什么过程。分布作用体的驱动由两类值（见图5中的小黑球）决定。第一，与保存当前时间 t_{clock} 的作用体槽有关。当更改时钟变量时，方法将测试作用体的运行是否结束（$t_{clock}=t_{code}$），如果结束将结果消息传给它的所有输出作用体。第二，与保存作用体消息的通道槽有关。当通道接收一条消息，方法将测试条件是否满足，如果满足消息便发送给通道的外部作用体。

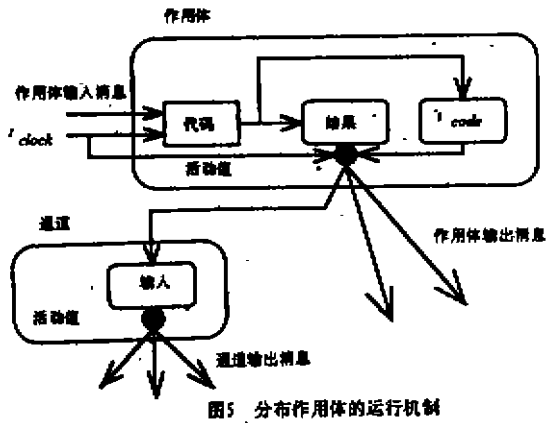

图5 分布作用体的运行机制

用 ABCDL 语言定义 DAI 系统，能够描述系统的控制和通信机制。ABCDL 模型由三类实体组成：顺序和分布作用体（分别表示单结点和多结点网络），以及通道（管理结点间的通信）。分布作用体的多层结构使问题分解多层次化，因而使子问题可以自然地分布在多个作用体上求解。对分布作用体的内外作用体间的合作管理是集中式的，由分布作用体作为接口；而内部作用体之间的合作管理是分布在各个通道上实现的。ABCDL 用于车辆导行系统时，对最优路径的求解分为长期（全局）路线求解和短期（当前）路线求解两个步骤。长期路线由离线规划器求出，短期路线由在线规划器求出。

结论 AGENTS 和 ABCDL 是面向对象的并发程序设计（OOCP）语言，可用于编制 DAI 的应用程序和为 DAI 系统建模。它们较之先前 OOCP 语言，如 ConcurrentSmalltalk、CooperA 和 Orient84/K，更易于理解，且更适合于 DAI 的应用领域。

为了更好地利用 OOCP 的方法，应该建立简便易行的编程和调试环境。程序设计过程中使用了大量的对象和消息传递关系，为了更有效地观察求解结构和建模结果，这些对象和消息传递关系应易于跟踪和操作，可将它们逐个以图形的方式记录下来。我们可以在工作站上设计和实现一个多窗口的编程辅助系统。例如，在图中添加一个结点表示增加了一个对象；添加一条连线表示增加了一种消息传递关系；用鼠标点某个结点的图标表示要对相应对象进行代码编辑或编译，此时会弹出操作选择菜单；调试并发程序是更困难的任务。一种调试辅助方法是建立对象的消息传递的长期历史记录，即每次收到或发送消息时，将消息连同当时局部存储器的状态作记录；也可间隔一段时间作一次记录。

参考文献

[1]Yonezawa A., Tokoro M., Object-Oriented Concurrent Programming: An Introduction, Object-Oriented Concurrent Programming, Yonezawa A., Tokoro M, Eds., The MIT Press, 1987

[2]Durfee E. H., Lesser V. R., Cooperative Distributed Problem Solving, The Handbook of Artificial Intelligence, Vol. IV, Barr A, Cohen P. R., Feigenbaum E. A., Eds., MA: Addison-Wesley, 1989

[3]Huang G. Q., Brandon J. A., AGENTS, Object-Oriented Prolog System for Cooperating Knowledge-Based Systems, Knowledge-Based Systems, 5(2), 1992

[4]Adorni G., Poggi A., An Object-Oriented Language for Distributed Artificial Intelligence, Int. J. Man-Machine Studies, 38, 1993