编程技术 Agent 人工智能

计算机科学2000Vol.27№.7

92-94

面向 Agent 编程——编程技术的一次飞跃*)

Agent-Oriented Programming

程显毅

(江苏理工大学计算机系 镇江212013)

TP311.1

Abstract Agent technology will provide an innovative computing model for computer and other science in furniture In this paper, we will discuss the basic principle of Agent-Oriented programming

Keywords Agent-oriented programming

一、引言

冯·诺依曼的"存储程序,程序控制"思想,使人们看到了用计算机模拟人的智能行为成为可能,1958年H. A. Simon 和 A Newell 对 AI 的一些乐观的十年估计,除了计算机下棋的目标比较接近外,其它的一些预测,直到今天还是预言,为了 AI 的目标,编程技术已经走过了两个阶段,面向规则的编程和面向对象编程。面向规则编程的核心是一种分析法,即把待求解的问题分解为若干子问题,然后将分解到的子问题再分解,如此下去,直到子问题成为可以被把握的对象为止,最后还原为待求解的问题.不可否认,这种方法是一种重要的思维方法,对一个未知对象的研究、分析的重要性是有目共睹的,然而它不是万能的,正是分析法的过度使用,使人们在研究一类具有非结构化的问题时,造成了认识上的混乱和困难,事实证明,分析法在 AI 研究中的使用是使 AI 陷入困境的根本原因。

分析方法应用到 AI 研究中会产生如下后果:通过对自然智能的分解,把人类智能分解为若干方面,如语音识别、图像理解、推理系统和专家系统等,在每个狭窄的领域应用取得了一定的效果,但往往与机器智能相比却显得肤浅得很,人们认识到"智能"是一种整体效应,如何在利用整体性编程是 AI 比较关注的问题,面向对象编程由此产生。

面向对象编程技术以类、对象、消息和方法为基本特征,追求的是求解空间与问题空间的直接模拟,即认为数据模型是与概念模型相对应的(数据模型是概念模型在数据世界的抽象描述),且两种模型各自之间的关系也是对应的,面向对象具有整体性,从宏观上把握问题,它能在不同的层次表达知识,在高层对象能封装复杂的行为,而且具体细节对该层知识又是透明的,还可构造相关信息,并把它们保持在一起,灵活性好。所以用面向对象方法处理知识表达,其优越性是明显的(如它所提供的封装和对象的局部存储机制)、面向规则的程序设计语言缺乏这种高级的结构化概念,然而,面向对象的不足在于:类及层次结构适合反映有序世界,对无序世界并不适合;它能很自然地表达分层知识,但不便表达启发式知识,尽管如此,面向对象方法给 AI 富于吸引力。在计算机科学、心理学、社会学、思维科学、协同学等相关学科迅猛发展的今天,考虑适用于无序世界的编程技术是自然的,面向 Agent 编程可能成为 AI 编程技术的第三次革命。

二、Agent 技术的基本原理

尽管对 Agent 尚缺乏统一的定义,但其基本思想是使 Agent(计算实体)能够模拟人类社会行为,即人类社会的组织形式、协作关系、认知方式,因此 Agent 概念较之对象概念具有更多的知识,具备更强的问题求解自治能力,Deker 认为,分布式问题求解的目标是创建一组粗粒度的协作 Agent,使它们共同解决某一任务[1]。因此,MAS(MultiAgent System)是通过多个 Agent 的协作来解决问题,其中每个 Agent 都具有独立的求解目标,可以自主地决定自身的行为且有能力使自己更好地适应环境。

根据这样一个原理,我们可以认为 Agent 是一个目标驱动的软件包(计算实体——具有目标、动机的对象),它通过感知器与外界通讯进行感知,并根据感知结果及内部状态的变化独立地通过效应器控制自身的行为,其模型如图1:

通讯模块是 Agent 参与活动、感知外界的接口,

*)本课题得到国家自然科学基金资助,程显毅 副教授,主要研究方向 自动推理,多 Agent,

它包含一个消息检测器、一个消息缓冲队列、一个消息处理器和一个消息发送器。工作时，消息检测器持续地监视外部环境的变化，并将外部事件存入消息缓冲队列，消息处理器则对消息缓冲队列中的消息进行合法性检查，分离出通讯参数和通讯内容。

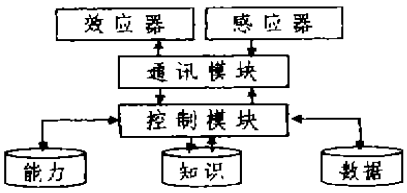

图1 Agent 模型

控制模块根据传来的消息的内容查询 Agent 能力以决定是否接受委托的任务，消息发送器回答接受/拒绝和发布请求信息。能力是对外界环境呈现的服务描述，是目标驱动的自身功能说明，刻画一个 Agent 能做什么。对那些超出能力的任务，Agent 拒绝接受，对接受的任务，控制模块首先从知识库中复制对应的问题求解规则，然后在问题知识的约束下，将问题求解规则转化为一个符合任务求解要求的可执行的事务处理流程，并产生相应的流程执行控制命令。控制模块解释这些命令，激活相应的事务处理进程。事务处理结果经通讯模块传到外界，过程中的 Agent 自身状态的变化记录在数据集中，事务处理进程所需的所有内部数据由数据集提供。

Bradshaw[5] 用意识立场，把 Agent 看作理性主体，通过信念(Belief)、愿望(Desire)、意图(Intention)属性来预测 Agent 行为，即 BDI 理论。

对于把主体看作意识系统的好处是：(1)对于设计者和分析者来说，这样是自然的；(2)对于描述复杂系统的行为提供了简洁的表示，有利于理解和解释；(3)不依赖于具体物理实现就可以得到许多主体的规则和模式；(4)可被 Agent 自身用来互相推理[5]。

实践证明，对于那些很复杂的系统，即使已经拥有了完全详尽的结构和工作机理的描述，也很难从设计立场来预测和解释其行为。在这种场合，就更适合采用意识立场描述。总之，意识观念是一种抽象工具，为人们描述、解释和预测复杂系统的行为提供了一种方便且熟悉的方式[6]。

Agent 的 BDI 理论最初是作为一种分布式智能的计算机模型被提出的，当前的研究主要有两个方面的动力：(1)控制分布式计算的复杂性；(2)克服人机界面的局限[5]。

伴随着分布式计算模型的广泛使用，急需解决 Agent 之间的智能性、互操作性的复杂性问题，人们就试图使用与人类意图对应的概念，从而在规划层提出一种封装，类似于对底层通信协议的封装[5]。

随着任务复杂性的增加，当前流行的直接操纵界面的优势已逐渐消失，人们试图用 Agent 技术实现一种间接管理的风格，无须用户显式地告诉计算机具体的行为，而只需给出大体的指导即可[5]。

三、面向 Agent 编程

现在我们知道，如果事物由分析并导致还原论来决定，那么这个世界就不会是我们今天所面临的如此纷繁的世界，人类也就不可能发生进化。

任何一门技术、一种方法，如果没有坚实的理论，就没有生命力，Agent 理论要建立在整体论、系统论、超协调、耗散结构、社会学、行为学、心理学、协同学、思维科学等众多交叉学科基础上。

生物学证明，构成人大脑智慧物质基础的核心是神经细胞，而神经细胞不过是由我们所熟悉的原子组成(如：碳、氢、氧等)，然而我们却始终不明白智慧的奥秘，原因在于"智能"是一种整体特性的表现形式，是大量组成成分神经细胞的综合能力的表现。心理学实验表明，人的"智能"具有强烈的时效性、灰度性、层次性，因此"智能"模型应是一种多维多层、动态互联的综合网络。思维科学也证实，人的思维是形象思维、抽象思维、灵感思维的综合。耗散结构理论认为，非线性系统具有开放性(通讯、协商、合作等)，这正是"智能"集成能力所必备的特性。……

我们有理由相信面向 Agent 编程主要致力于解决以下问题：

1 多信道接受信息

人类自身的肢体能力有限，却能自如地解决那些看起来难以处理的棘手问题，并能做出合理的决策和反应，重要的一个原因是人类可以通过多种信道(如：眼、耳、鼻、舌、身等)获取各种信息，综合利用各种信息，采取逐步尝试(试错)的方法达到合理的目标，得到有意义的解，给计算机装上录音机、摄像机、传感器等，计算机就可以实现多信道接受信息的功能。

2 自动获取方法

人的知识有限，但人类对问题求解方法却是多种多样的，在进行问题求解时，最主要的是将这些方法综合在一起，组成一个复杂的推理过程，虽然人的能力表现形式很多，如听力、视力、记忆力、想象力等等，但这些能力不是孤立的，而是相互依存、相互制约。仅有知识的 AI 系统不是有能力的系统，重要的是有运用知识求解问题的方法。目前已开发的一元方法(如试错法)系统导致 AI 系统对新问题探索能力提高，如果能开发出多元、非线性方法获取系统，方法获取能力将会

得到改善。

3 利用常识

我们知道，人类拥有大量的常识知识，人们会在求解问题时不知不觉地应用常识知识，这些广泛的常识知识的应用，往往能极大地界定人类的思考范围，使推理加速收敛，领域知识与常识知识的综合，在人的大脑中体现了"智能"的整体性。Lenat 的 CYC 工程是常识知识利用的典范，由于常识巨大（已运行了15年），CYC 取得了一些惊人的成功，当然 CYC 还处在一种非常初级的阶段，Agent 具有常识利用机制体现在两个方面，一是它的记忆特性（常规程序没有记忆特性），二是知识挖掘特性。

4 系统集成

将已有的 AI 系统集成在一个系统中，使它们相互联系，相互制约，根据人的行为的社会性，MAS 的集成能力，使得软件 Agent 获得了整体综合效应。

5 模式识别（非搜索推理）

对一个特定的字母（如"A"）无论是大是小，是正是斜，是倒是草，是标准还是不标准，有背景噪音等，人们能轻而易举辨识它们，甚至缺损的"A"也能辨识，这决不是搜索，而是整体的模式辨识，相当于人的形象思维，这是综合法较难使用的步骤。现在不行不等于将来不行。

6 类比归纳

如果 A 和 B 看起来有些无法解释的相似之处，那么值得费时间去寻找其它的共性。这是人类具有创造能力的源泉，类比和归纳涉及当前情况与另一情况的部分匹配，这里有两个相互独立的方案[3]：(1)纵向变换：遇到复杂情况，同一个更简单的情况做类比说明；(2)横向变换：跨领域映射较少见，但能带来好处。如："治疗疾病象是打仗"可帮助医生设计出新的治疗方法，也有助于士兵发明新的作战方案。

类比归纳渗透在人的交流中，似乎不合推理，但却常常有效，拓展了知识库中的知识相关性，它可用来建造关于情况和数据的有趣和新颖的解释，猜测属性值，提示可能行得通的方法等。

由于人们对智能的认识在不断的深入，用面向 Agent 编程可解决的问题还会增加。根据 Agent 的 BDI 理论模型，面向 Agent 编程的具体过程为：

问题 —解合→ 信念 —需求满足→ 愿望 —规划→ 意图 —决策→ 行为（目标，软件 Agent）

为了对面向 Agent 编程有更清晰的认识，我们对比一下分析法编程的过程：

问题 —分析→ 算法 —描述→ 流程图 —编码→ 程序 —执行→ 结果

结束语 面向规则编程的基本单元是规则，使用分析法，面向对象编程的基本单元是对象，使用模块法；面向 Agent 编程的基本单元是 Agent，使用综合法。面向 Agent 编程语言主要有：Z 语言；KQML（Knowledge Query and Manipulation Language）语言；AOPL（Agent-Oriented Programming Language）语言；Agent0语言等。

尽管面向 Agent 编程在开发复杂新型应用系统中起着重要作用，涌现出大量的基于 Agent 的系统软件，但基本上都可以采用传统的技术解决。主要是三个方面的原因：

(1)对面向 Agent 编程原理理解有偏差。

(2)另外 Agent 技术不成熟限制了它的应用领域，主要表现在：第一，整个系统不可预测，不确定的，哪个 Agent 将与其它哪个 Agent 以什么方式交互来实现什么目标，这是不能事先确定的，由于 Agent 可以自由作出决策，我们不能保证 Agent 之间的依赖关系能得到有效管理，由此可能导致死锁。第二，整个系统的性质和行为在设计阶段不能确定，虽然可以给出每个 Agent 的行为规范，但由于整体行为只有在运行时才能体现。

(3)由于面向 Agent 编程技术的应用受理论和硬件的限制，在目前情况下，我们要探讨如何充分发挥面向 Agent 编程技术特点，编写模拟高层次"智能"的程序。

由于我们对 Agent 的认识还很肤浅，文中用的一些概念可能不恰当，但我们的目的是让 Agent 技术象面向对象技术一样，使计算机软件编程产生一个飞跃。

参考文献

1 Lenat D B On the Thresholds of Knowledge. Artificial Intelligence. 1991，47 Special Volume on Foundation of Artificial Intelligence

2 聂勇军，等. 论人工智能整体观 计算机科学，1997，24(5) 70～72

3 毛新军，陈火旺. 智能体的理论研究. 计算机科学，1997，24(5) 63～66

4 Deker K. Distributed Problem-Solving Technologies a Servey. IEEE trans. on System, Man and Cybernetics, 1987，SMC-17(5)，729～740

5 Bradshaw M An Introduction to Software Agents. In Software Agents, Springer, 1996

6 Wooldridge M, Jennings N R. Intelligent Agents: theory and practice The Knowledge Engineering Review, 1995, 10(2)，115～152