

一种新的基于 Agent 的体系结构

袁杭萍,肖登海,连向磊,王玲玲

(解放军理工大学 指挥自动化学院 指挥自动化系,江苏 南京 210007)

摘要: Agent 体系结构是由理论研究向系统实现的关键一步。首先,在 Agent 特性分析和形式化描述的基础上,阐述了 Agent 的层次概念框架。然后,研究 SubSumption, BDI 和 GRATE 等典型结构特点,比较了反应式和慎思式两种体系结构,反应式 Agent 和慎思式 Agent 体系结构各有优劣。确立采用分层式混合结构的设计思想,并结合 Agent 的特性,从信息流和控制流的角度来刻画更为复杂的层次结构。在此基础上,设计了一种带序列池(Pool)的二维混合体系结构,并对其实现机制进行了深入研究。在不增加系统复杂性的前提下,提高系统的效率 and 能力。

关键词: Agent; 体系结构; 二维结构

中图分类号: TP18

文献标识码: A

文章编号: 1673-629X(2010)01-0050-04

A New Architecture Based on Agent

QIU Hang-ping, XIAO Deng-hai, LIAN Xiang-lei, WANG Ling-ling

(Dept. of Command Automation, Institute of Command Automation,

The PLA University of Science & Technology, Nanjing 210007, China)

Abstract: The architecture of Agent is the key step from theory to realization. Firstly, due to analyzing the characteristic of Agent and formalization, a conceptual hierarchy frame is explained. Reactive and deliberative architecture is compared by studying representative architectures such as SubSumption, BDI, GRATE and so on. Both reactive and deliberative architecture have advantages and disadvantages. Combined with the characteristic of Agent, hierarchy is described in the form of information and control flow. A 2-View hybrid and hierarchical architecture, called as SPA-p architecture, is designed by the introduction of pool. Mechanisms of the model are investigated. The proposed model can improve efficiency and capability of system without increasing computational complexity.

Key words: Agent; architecture; 2-view hierarchical architecture

0 引言

Agent 的概念出现于 20 世纪 70 年代的人工智能中,80 年代后期逐步成长起来。而今,Agent 成为 AI 和计算机领域最活跃的研究内容之一。分布式人工智能(Distributed Artificial Intelligence, DAI)是人工智能研究领域的一个重要分支。DAI 研究工作大致分为分布式问题求解(Distributed Problem Solving, DPS)和多 Agent 系统(Multi-Agent Systems, MAS)两个方面。DPS 主要研究如何分解某个特定问题,并将其分配到一组拥有分布知识并相互协作的结点上;MAS 主要研究这组 Agent 为了求解问题时,如何协调各自的知识、目标、策略和计划等。目前,MAS 主要研究多 Agent 理论、通信和交互技术、体系结构和组织形式、面向 A-

gent 的程序设计方法和语言以及多 Agent 间的协调、协作和协商等^[1,2]。

在 Agent 结构以及模型研究方面,杨建池等人^[3]针对多 Agent 协作提出了一种 A2FC 体系结构;张健等人^[4]给出了一个满足弱概念特性的混合型 Agent 概念结构;JOHN R. GRAHAM 等人^[5]提出了一种灵活的多 Agent 模型-DECAF-A。

1 Agent 的定义和特性

Wooldridge^[6]给出了 Agent 的强定义和弱定义,其中关于 Agent 的弱定义得到大多数研究者的认可。

Agent 的弱定义:具有以下特性的硬件系统或基于软件的计算机系统称为弱概念意义下的 Agent(见表 1)。

表 1 特性-能力映射表

基本特性	自治性	反应性	社会性	主动性
对应能力	自我表现	针对环境	基于交互	面向目标

收稿日期:2009-05-18;修回日期:2009-08-09

基金项目:总装预研基金项目(9140A06020206JF8102)

作者简介:袁杭萍(1965-),女,浙江杭州人,博士,教授,硕士生导师,研究方向为指挥自动化、人工智能、建模与仿真。

(1)自治性(Autonomy):在无人或其它系统的直接干预下可自主操作运行,对自己的行为和内部状态有某种控制能力。

(2)反应性(Reactivity):能够感知所处的环境,并在一定时间内对环境的改变做出反应。

(3)社会性(Social ability):通过某种 Agent 通信语言与其它 Agent 进行交互。

(4)主动性(Pro-activeness):Agent 不仅能够简单地对环境作出反应,而且通过接受某些启示信息,可以展现出目标驱动的主动行为。

文中将其归纳为四个能力:自我表现的能力、针对环境的能力、基于交互的能力和面向目标的能力。

2 Agent 的体系结构

Agent 体系结构是指基于 Agent 规范理论来建立设计、实现 Agent 系统的构架。它描述了组成 Agent 的基本成分及其作用、各成分的联系及交互机制等。

2.1 Agent 的层次概念框架

Nwana^[7]定义了 Agent 的三层概念结构:定义层、组织层和协作层。这个概念结构提供了一种描述 Agent 应用特征的框架。结构如图 1 所示。

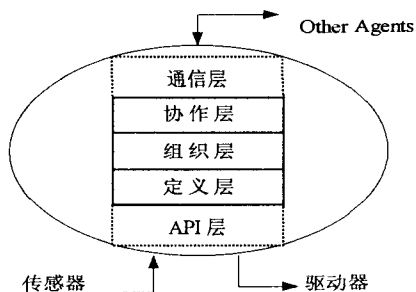


图 1 Agent 的三层概念结构图

定义层:在这一层中,Agent 被描述为一个自治的理性实体,包括 Agent 的推理学习机制、目标、资源、技能等。组织层:定义了 Agent 与其他 Agent 的关系,包括 Agent 在 Agent 团体中所扮演的角色以及 Agent 之间的相互感知等。协作层:指明了 Agent 的社会能力,例如它的合作和协商技术。通信层:定义 Agent 之间通信的更低一级的细节。API 层:将 Agent 与它的资源、技能的物理实现联系起来。

按照人类思维的层次模型,可以将 Agent 内部体系分为反应式和慎思式结构。

2.2 反应式和慎思式结构对比分析

根据上述讨论,对反应式和慎思式两种结构进行对比分析,如表 2 所示。

表 2 反应式和慎思式结构对比分析

类型	原理	模型	优点	缺点	代表
反应式	基于行为主义	See-Action	简单经济、易计算;响应快速	智能特性较低;灵活性低	包容结构
慎思式	基于符号推理	BDI	较高的智能性;灵活性较高	设计较困难;执行效率较低	GRATE

反应式和慎思式结构都各有优缺点,结合这两种结构的混合式体系结构成为研究的热点。PRS, TouringMachine 和 InterRRap 为三种典型的混合式结构。PRS 结构中解释器负责所有要素(信念、愿望、意图和动作)的维护和更新。TouringMachine 功能层和 InterRRap 任务层均被内嵌在一个控制子系统中,且行为模型是基于任务分解的。三种混合结构都兼具反应式和慎思式的特点,但解释器和控制系统均增加大量的额外开销,同时其设计也要求相当高。

根据以上分析,文中确立采用分层式混合结构,并引入序列池,提出一种简单高效的混合 Agent 模型。该模型采用基于事件-状态实现机制的。

3 SPA-P 结构模型

3.1 从信息和控制流两个角度进行分析

Agent 既能针对环境的变化作出及时响应,也能作出目标驱动的行动,要处理这些不同类型的行为,可以构造不同的子系统。各种子系统被排列成层次间相互交互的等级结构,从层次间信息流与控制流的角度来刻画这种结构,构建如图 2 所示的层次结构。

水平层次结构一个很大的优点在于概念上的简洁性: n 种不同的行为,那么就实现 n 个不同层。为了保证水平层次结构是一致的,这种结构一般要包括一个 modulation 函数,来决定在给定时间内由哪一层来控制 Agent。那么如果结构中有 n 层,每一层都能够建议 N 种可能的动作,这就意味着要考虑 N^n 种交互。所以,中心控制系统的引入也为 Agent 决策带来了瓶颈问题。

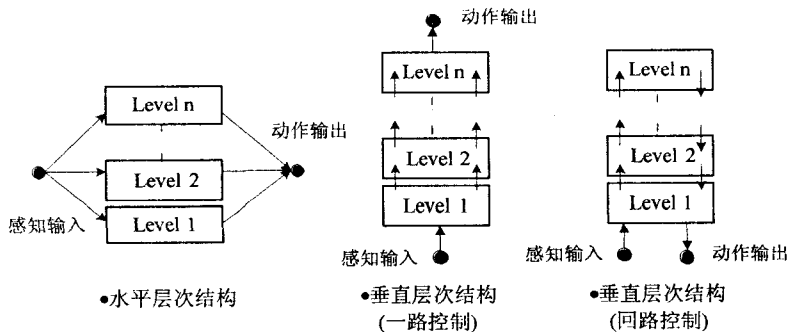


图 2 层次结构

垂直层次结构中,单向结构控制流顺序经过每一层,直到最后一层才生成动作输出。在回路结构中,信

息向上流动,然后控制再向下流动回来。信息都是向上流动到组织的最高层,然后命令再流动下来。层次间交互作用的复杂性都降低了: n 层之间有 $n-1$ 个接口,如果每一层都能够建议 N 个动作,那么最多有 $N^2(n-1)$ 个层次间的相互作用需要考虑。这显然比水平层次结构的情况简单多了。然而,这种简单性却是以灵活性为代价的:为了使垂直层次结构作出决策,控制命令必须经过每个不同层。同时,任何一层的故障都可能会对 Agent 的性能产生严重的影响。

在上述分析基础上,加入一个序列池单元,以提高 Agent 响应的速度,增强应对紧急情况的能力。该结构需要考虑 $N^2(n-1) + N$ 个层次间的相互作用。在不增加系统复杂性的情况下,提高系统的效率和能力。结构如图 3 所示。

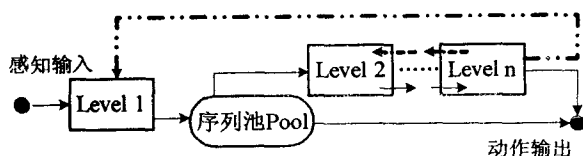


图 3 带序列池的混合结构

3.2 从结构和功能两个视图进行设计

设计带序列池的混合体系结构,该结构包含二维结构视图,其中功能视图下有感知、控制和执行三个单元模块,结构视图下有物理层、反应层和慎思层三个层次。

感知单元获取 Agent 所处任务环境以及内部状态的相关信息。控制单元包括反应层和慎思层两个层次。反应层由预设的简单规则支持,而慎思层由知识库和状态库支持。反应层为 Agent 提供低层次的快速控制,使其具有一定的快速反应能力。慎思层为 Agent 提供较高层次的自主控制能力,其特征是基于知识进行复杂推理。慎思层需要对感知单元提交的环境信息进行较为细致的理解,结合自身状态进行综合分析,确定下一步将采取的行动,并将产生的行动系列提交给执行单元。执行单元为 Agent 提供基本的行动能力。控制单元中的反应层和思考层,基于环境状态以及指令等信息,产生相应的行动序列,执行单元根据行动序列选择并加载执行相应的执行模块。

不同角色的 Agent,将根据其任务目标,携带不同的任务模块。模型结构如图 4 所示。

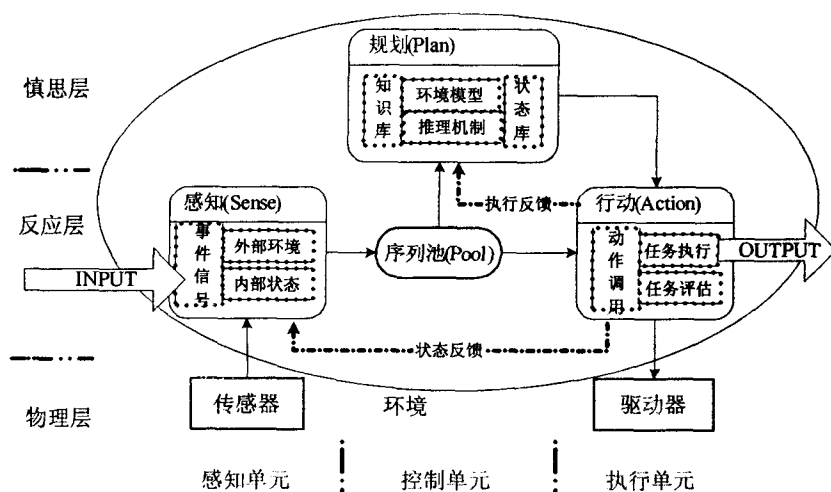


图 4 SPA-p 二维结构模型

SPA-p(Sense Plan Action-Pool)结构模型,综合了反应式和慎思式两种结构的优点,具有较强的灵活性和快速响应性。包括如下两部分的层次结构:高层是一个包含符号世界模型认知层,它用传统符号 AI 的方式处理规划和进行决策;低层是一个能快速响应和处理环境中突发事件的反应层,它不使用任何符号表示和推理系统。反应层被给予更高的优先级。

该模型由一个介于慎思层和反应层之间的序列池来决定系统运行的通路,一条通路实现传感-反应,另一条通路实现传感-规划-反应。第一条通路显然避免了复杂的规划网络。将紧急事件的处理从一般事件中抽离出来,在传感器 Agent 和效应器 Agent 之间建立新的映射关系:当紧急事件发生时,传感器 Agent 的报告时间直接影响效应器 Agent,这样系统不必等待融合规划 Agent 的消息。紧急事件处理完毕后,仍然可以继续通过融合规划 Agent 进行控制。这种改进的关键技术包括:支持混合结构的事件-状态模型和一个高效独立的序列池。

3.3 实现技术

(1) 事件状态安全机制。

(离散)事件状态模型(Discrete Event State Model, DESM)^[8]用于描述自主式机器人的行为及 Agent 的行为。该模型用事件信号描述状态变化,用状态变化过程描述行为,以建立 Agent 行为模型。但该模型对于紧急事件、紧急状态的变化没有明显体现,并且是在安全环境的前提下提出的。

针对外部环境的复杂性,以及 Agent 应对紧急情况快速反应能力,提出了事件状态安全机制(Event State Safety Mechanism)。采用八元组结构来进行刻画: $ESSM = \langle E, Eu, Q, R, F_0, F_e, F_s, F \rangle$ 。 E 是事件集, Q 为状态集, R 是一个映射: $R(q_n, e_x) = q_m (q_m \in$

$Q, e_x \in E$), F_e 为执行态, F_s 为暂停态, F 为终止态。该结构模型对状态和事件分别进行抽取, 不仅仅能快速响应紧急事件, 而且能感知环境安全态势进行动作响应。

在初始状态 F_0 , 启动环境状态感知, 判断环境安全态势。当满足安全条件时, Agent 进入执行状态 F_e , 执行计划任务。同时, 不断进行态势更新。当环境安全处于临界状态时, 停止任务执行模块和通信模块的工作, 进入暂停状态 F_s , 等待恢复正常工作, 并采取计划变更等措施。当环境安全态势评估为安全或接收到激活命令时, 重新恢复执行任务, 转为执行状态 F_e 。当感受到的安全风险超过预定危险阈值或接收到终止指令时, Agent 将进入终止阶段 F 。

(2) 序列池的实现机制。

一个好的控制器在保证将信息快速准确传递的同时, 还必须保证不会给系统造成额外的负担。普通状态下, 序列池在传感器 Agent 与推理 Agent 之间传递事件和状态; 紧急状态下, 序列池直接将紧急事件转换为适当的控制命令送往效用器 Agent。在遇到紧急情况时, 改进后的模型直接将事件映射为控制消息, 发往效用器 Agent。这样做的目的是绕开复杂的推理从而提高系统的反应能力。

将 Agent 实体和它所处的环境都看做具有有限状态的自动机, 这样, 它们内部特征可以简化为相关的内部状态变量, Agent 在交互过程中的思维变化则可以通过内部状态的转移来刻画^[9]。

首先, 定义外部环境为如下的两组:

Environment \equiv < STATE, CHANGE >

其中 STATE 表示外部环境状态集, state(state? STATE) 表示某时刻 Agent 所处外部环境的状态。映射 CHANGE 用以描述环境状态的变化。

序列池 Agent 可定义为:

PoolAgent:: = {ID, PER, E, R, D_{pe} , D_{er} , Rule - Pool, See, Next, Estimate, Action}

其中, ID 为 Agent 的唯一标识, PER 为视觉状态集, E 表示为所有可能的事件集合, R 表示所有可能的事件对应的所有可能的结果状态所构成的集合, D_{pe} 表示视觉状态与方案集的对应规则集, D_{er} 表示事件集与结果状态集之间的对应关系, Rule - Pool 表示具体的决策法则构成的集合, 它们分别定义为:

$D_{pe}:: = \{(p, e) \mid p \subseteq P, e \subseteq E\}$

$D_{er}:: = \{(e, r) \mid e \subseteq E, r \subseteq R\}$

Agent 的内部行为表示为下列映射:

See: $S \rightarrow P$

Next: $P \rightarrow E$

Estimate: $E \rightarrow R$

Action: $E \times R \rightarrow E$

实现过程描述为: 对于给定的 $p_0 (p_0 \in p)$, 若 $\exists (p, e) \in D_{pe}$, 则 e 为对应的事件集; 对给定的 $e (e \in E)$, 若 $\forall e \in e, \exists (e_0, r_0) \in D_{er}$, 则 $\bigcup r_0$ 为对应结果状态集。此时, 已确定某视觉状态的对应事件集 e_0 及其对应结果状态集 r_0 , 则基于满意法则的 Pool 决策实现可描述如下:

1. 找 $\gamma \subseteq r_0$, 使得 $\forall \gamma_0 \in \gamma$, 有效用值 $\varphi(\gamma_0) \geq K$, 其中 K 是反映 Agent 期望水平的常量;

2. 找 $\epsilon_0 \in e_0$, 使得 $\sum_{\gamma_0} \partial(\epsilon_0, \gamma_0) \gg \sum_{\gamma'} \partial(\epsilon_0, \gamma')$, 其中 $\gamma_0 \in \gamma, \gamma' \in (r_0 - \gamma)$, $\partial(\epsilon_0, \gamma_0)$ 表示 γ_0 对应 ϵ_0 发生的主观期望概率;

3. 如果 e_0 中存在多个 ϵ_0 , 则提高期望水平 K 并重新返回步骤 1; 如果 e_0 中不存在满足条件的 ϵ_0 , 则降低期望水平 K 并重新返回步骤 1; 直到产生一个满意的 ϵ_0 作为确定的输出行为为止。

(3) 性能分析。

系统的第 m 次处理延迟时间是指从传感 Agent 第 m 次感知环境, 经过融合规划网络 Agent 处理, 传递消息给效用器 Agent, 直到控制完成的时间, 记为 $T(m)$ 。

设 $t_s(m)$ 为传感器 Agent 最大处理时间, $t_p(m)$ 为融合规划网络 Agent 总处理时间, $t_a(m)$ 为效应器的最大处理时延, $t_{sp}(m)$ 为传感器 Agent 到融合规划网络 Agent 之间的网络时延, $t_{pa}(m)$ 为融合规划网络 Agent 到效应器 Agent 之间的网络时延。

那么, 不带序列池的处理时间可表示为:

$T(m) = t_s(m) + t_p(m) + t_a(m) + t_{sp}(m) + t_{pa}(m)$

SPA - p 模型在紧急情况下的处理时间可表示为:

$T(m)' = t_s(m) + t_b(m) + t_a(m) + t_{sp(b)}(m) + t_{(b)pa}(m)$

可以看出, 时间 $T(m)$ 和 $T(m)'$ 主要的差别在于 $t_p(m)$ 和 $t_b(m)$, 即融合规划网络的处理时间和序列池的处理时间。序列池的处理时间设计上比融合规划网络的处理时间要小。所以, 采用 SPA - p 模型后, 系统的时延得到了有效的缩短。

4 结束语

在对 BDI, GRATE, PRS, TouringMachine 和 InterRRap 等典型的体系结构研究基础上, 引入序列池, 提出二维视图的 SPA - p 模型。

分别从分析、设计和实现技术角度进行了阐述, 在

(下转第 57 页)

类各自添加不同的方法,而多态性实现同一功能,但用的是不同的方法和不同的子类。由于XML不是行为语言,而是一种标记语言,所以多态性出现在属性级别^[8]。采用前文的继承示例,Hstudent从student中继承了name,number,class属性,但是希望Hstudent的Number属性与Student的Number属性不同,而是某些特定的数字号码,因此可以在模式中实现这一规则,当Hstudent中Number不是这些特定的数字号码时,那么它将报告错误。

代码如下:

```
//定义 Hstudent 复杂数据类型
<xs:complexType name="Hstudent">
  <xs:complexContent>
    <xs:restriction base="student">
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="class" type="xs:string"/>
        <xs:element name="number" type="extentNumber"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
//定义 number 元素
<xs:simpleType name="number">
  <xs:restriction base="xs:string">
    </xs:simpleType>
//定义 extentNumber 元素
<xs:simpleType name="extentNumber">
  <xs:restriction base="number">
    <xs:pattern value="[0-9]{10}"/>
  </xs:restriction>
```

(上接第53页)

不增加系统复杂性的基础上,提高了系统的快速反应能力,同时保证Agent的理性思维能力。具有一定的理论意义和应用价值。

参考文献:

- [1] Tokoro M. Computational Field Model: Toward a New Computing Model/Methodology for Open Distributed Environment [C]//Proceeding of 2nd IEEE Workshop on Future Trends in Distributed Computing System. Cairo, Egypt: Amazon. co. uk, 1990.
- [2] Osawa E. A Scheme for Agent Collaboration in Open Multi-Agent Environment[C]//Bajcsy R. Proceeding of IJCAI'93. Chambéry, France: [s. n.], 1993: 352-358.
- [3] 杨建池,王运吉,黄柯棣.一种Agent体系结构A2FC研究

</xs:simpleType>

4 结束语

通过上文的分析,XML具有良好的面向对象特性。随着XML应用不断深入,规模不断扩大,设计XML的复杂度也就随之增加。因此,文中将更加深入的研究与分析,通过借鉴软件工程的思想来快速方便地构建XML。

参考文献:

- [1] Bohrer K, Liu Xuan, McLaughlin S, et al. Object Oriented XML Query by Example[M]. Berlin, Heidelberg: Springer-Verlag, 2003.
- [2] Goldfarb C F, Prescod P. XML手册[M].第4版.王艳斌,译.北京:电子工业出版社,2003:2-15.
- [3] Dewhurst S C. C++必知必会[M].荣耀,译.北京:人民邮电出版社,2006:1-5.
- [4] Lee D. Comparative analysis of six XML schema languages [J]. ACM SIGMOD Record, 2002(29): 117-151.
- [5] 周凯波,金斌,周剑岚,等.基于XML的面向对象案例表示方法[J].武汉理工大学学报:信息与管理工程版,2005, 27(3): 78-80.
- [6] 李浩,孙统风,孟现飞,等.基于面向对象思想构建XMLSchema[J].微机发展(现更名:计算机技术与发展), 2003, 13(6): 59-61.
- [7] Wang Guo ren. Extending XML schema with object-oriented features[J]. Information Technology Journal, 2005, 4(1): 44-54.
- [8] 姜岩一,宫义山,王国仁,等.基于面向对象XML文档的面向方面模型[J].沈阳建筑大学学报,2006, 22(4): 673-676.

[J].系统仿真学报,2008,20(10):2560-2567.

- [4] 张健,曾广周.一种满足弱概念的混合型Agent结构[J].计算机工程与应用,2008,44(17):6-9.
- [5] John R, Graha M, Keith S, et al. DECAF-A: Flexible Multi-Agent System Architecture [J]. Autonomous Agents and Multi-Agent Systems, 2003(7): 7-27.
- [6] Wooldridge M. Intelligent Agents: Theory and Practice [J]. Knowledge Engineering Review, 1995, 10(2): 115-152.
- [7] Nwana H. Software Agent: an Overview [J]. Knowledge Engineering Review, 1997, 12(2): 205-245.
- [8] 张友军,朱森良,吴春明,等.自主式移动机器人系统的体系结构[J].机器人,1997,19(5):378-383.
- [9] Nilsson N. Logic and artificial intelligence [J]. Artificial Intelligence, 1991, 47(1): 31-56.