

第8章 基于Agent的建模方法

□ 目前采用的行为建模方法主要有：

- 有限状态机
- 基于规则的方法
- 基于案例推理
- 神经网络

□ 各有优缺点：

- 有限状态机和基于规则的建模方法适用于具有小型知识库的系统。
- 神经网络建模方法无法跟踪模型的推理过程。
- 基于案例推理方法不能全面反应认知决策过程。

-
- Agent从模拟人的行为出发，对一个实体的信念、意志、期望等精神状态进行描述。
 - Agent具有自主性、社会性和主动性等特征。
 - 因此基于Agent的建模技术，更能准确表述和表达仿真模型中具有智能的实体，采用Agent技术可以方便构建基于智能体模型模拟人的行为，构建更实用的行为模型。

-
- 基于Agent的建模（Agent-Based Modeling, ABM）方法衍生出一系列的相关概念和技术：
 - 基于Agent的软件工程
 - 基于Agent的计算
 - 面向Agent的程序设计
 - Agent通信语言（ACL）等。
 - 20世纪90年代以来，Agent成为计算机领域和人工智能领域研究的重要前沿，许多领域借鉴或采用该概念（或思想）进行本领域的研究工作（建模与仿真），如社会科学、经济学、生物学等。

Agent的定义

- Agent 最初来源于分布式人工智能的研究。
- 目前，由于Agent 已经渗透到计算机科学技术的许多领域和许多非计算机领域中，所以从一般意义上很难给出Agent 严格而清晰的定义，到目前为止，还没有形成一个统一确定的Agent定义。

Agent的定义

□在英文中，“Agent”有三种含义：

- 一是指对其行为负责任的人；
- 二是指能够产生某种效果的，在物理、化学或生物意义上活跃的东西；
- 三是指代理——接受某人的委托并代表他执行某种功能或任务。

●基于对Agent英文原意的理解，常被人解释为代理。但随着Agent广泛应用的不同领域，不再局限于“代理”。

Agent的定义

□ 一个Agent应具有的特性：

- 1) Agent是一个具有明确边界和界面的问题求解实体。
- 2) Agent处于特定环境之中，通过感知器来观测环境，通过效应器来作用于环境。
- 3) 自治性。Agent能自行控制其状态和行为，能够在没有人或其他程序介入时操作和运行，这是Agent最本质的特征。
- 4) 社会性。无论是现实世界，还是虚拟世界，通常都是由多个Agent组成的系统。在该系统内，单个Agent或多个Agent的行为必须遵循符合Agent社会的规则。能通过Agent交互语言，与其他Agent进行灵活多样的交互，并有效进行合作。

Agent的定义

5) 反应性。Agent能够感知其所处的环境（物理世界，或操纵人机界面的用户，或与它进行交互和通信的其他Agent等等），能及时迅速地作出反应以适应环境变化。

● 在一些特定领域的研究，特别是人工智能领域的研究，还赋予Agent一些更高级的特性，使其更符合于所研究对象的特征：

- 1) 理性。Agent没有相互冲突的目标。
- 2) 诚实性。Agent不故意传播虚假的信息。
- 3) 友好性。Agent总是尽可能地完成其他Agent的请求。

Agent的定义

□ Agent的特性常常因应用的不同领域而有所不同，形成对 Agent 的不同理解或定义，自治性是Agent概念的核心。

□ 在实际应用中，Agent常被分为三种类型：

- 类型Agent：描述特定实体或某一类实体。
- 集中服务Agent(多Agent系统中)：为多个Agent提供特定的服务或一组服务。
- 移动Agent：可在不同的实体之间进行移动。

Agent的定义

□ 总结:

- Agent是实际系统的物理实体抽象或系统的功能抽象，能够在一定的环境中为了满足设计目标而采取一定的自主行为；
- Agent总是能感知其所处的环境，并且有可以影响环境的多个行为能力，能够适应环境变化。

一个Agent不就是一个对象吗？

NO



Agent与对象

- 对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位。
- 一个对象由一组属性和对这组属性进行操作的一组服务组成。
- 从认识论的角度：对象就是一种抽象技术，最基本特征是封装、继承和多态；
- 从软件的角度：对象是一个计算实体，封装了一些状态以及可根据这些状态采取特定措施的方法，对象之间可通过消息的传递来进行交互。

Agent与对象

□ 对象与Agent有许多共同点:

- 如数据和方法的封装;
- 如Agent拥有对象的继承与多态等性质。

Agent与对象

□ Agent与对象的一些明显区别：

● 一是Agent和对象的自治程度。

对象可以完全控制其内部状态，没有在控制其行为方面表现出自治性——一个对象申明了一个公有方法，不能控制该方法是否被执行了。

➤在面向对象情况下，决策由调用方法的对象决定；

➤在Agent情况下，决策由收到请求的Agent决定。

● 二是有关自治行为的灵活性，即自治性、反应性、社会性。

标准的对象模型没有关于行为特性的说明。

Agent与对象

- 三是Agent有自己独立的控制线程；而在标准的对象模型中，整个系统才有一个控制线程。
- 标准对象模型中的主动对象可能与Agent概念最相近。
主动对象——是一组属性和一组服务的封装体，其中至少有一个服务不需要接收消息就能主动执行（主动服务）。
- 主动对象拥有自己的控制线程，可以在没有其他对象控制情况下自主实施自己的某种行为或某些行为。
- 主动对象不一定具有Agent灵活的自治行为。

Agent与对象

● 值得注意的是：尽管Agent与对象有着重大的区别，但这并不妨碍用面向对象技术来实现Agent。许多Agent开发工具和应用实例都是用面向对象技术来实现。

◆ Agent技术在继承面向对象技术所有优点基础上，赋予Agent更多人性化的特征，与实际自然系统或人工系统更贴近。
面向对象能做的事，Agent技术都可以做，Agent技术还可以处理许多更为复杂的问题。

● 目前，行为建模研究热点是基于多Agent系统的建模。

多Agent系统

- 单一Agent很难对存在于动态开放环境之中的大规模复杂问题进行求解。
- 人类智能本质上是社会性的，人们往往为解决复杂问题组织起来，这些组织能够解决任何个人都无法解决的问题。
- 面对复杂系统，必须用多个Agent来刻画、抽象这样的系统。由多个Agent组成的系统称为多Agent系统（Multi-Agent System, MAS）。

多Agent系统

多Agent系统 (Multi-Agent System, MAS) ——将多个Agent组成的系统，是一种新的方法论，是生产分布式控制、自适应及处理复杂过程的关键技术。

□ 多Agent系统具有以下特点：

1) 高层次的交互。

可以描述复杂的社会交互模式：合作、协调、协商。通过更高层次的Agent通信语言（如基于言语行为理论Speech-Act theory），在知识层次交互，是一种柔性交互。

多Agent系统

2) Agent之间丰富的组织关系。

➤ Agent间的关系——来自于组织中各种关系，如同等关系、上下级关系等。

➤ Agent 系统结构——来自组织的结构，如团队、群组、联盟等。

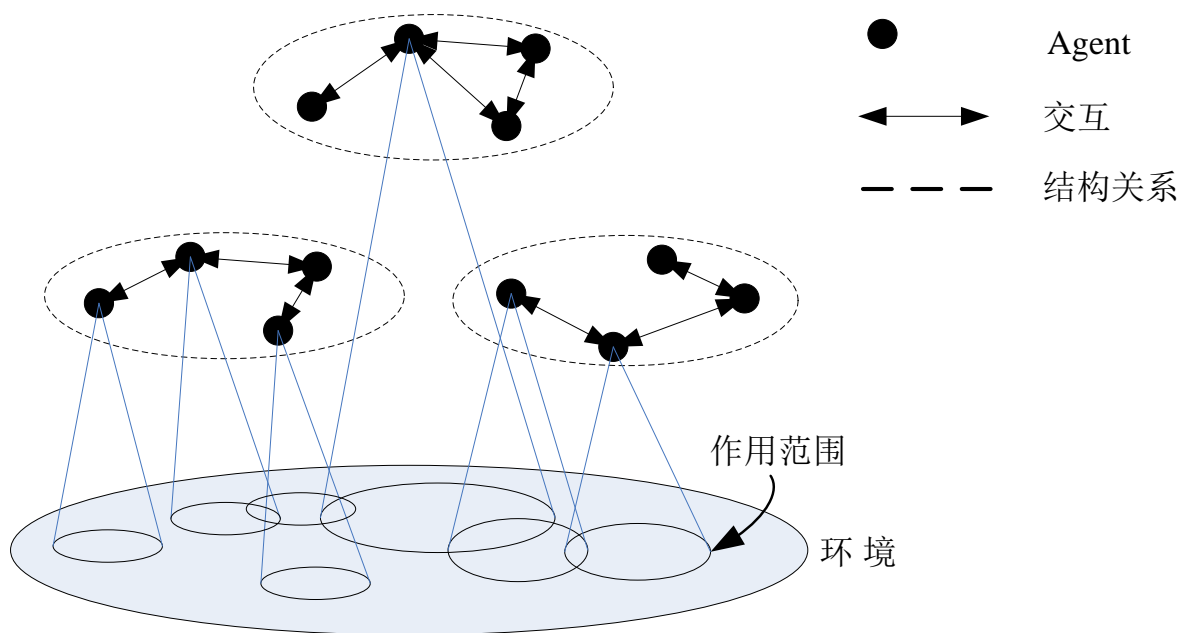
➤ 这种关系和结构随Agent之间的交互不断演化。

3) 数据、控制和资源的分布。

MAS特别适合需要多个不同的问题求解实体 (Agent) 相互作用共同求解某个共同的问题或各自问题的领域，多数情况下，实体、数据、资源在物理或逻辑上是分布式的。

多Agent系统的结构

- 各个Agent相对独立，Agent之间可能存在复杂的关系



多Agent系统的关系类型

□ 结构相关

结构相关性是指不同Agent之间具有结构关系，如小组关系、上下级关系等。

这种结构关系将对系统中Agent的运行以及Agent之间的相互作用产生影响。

□ 行为相关

不同的Agent对环境的一部分产生影响，某些Agent的影响范围发生重叠，则它们之间就产生了行为上的相互影响。

Agent的作用

□MAS架构特别适合用来解决具有模块化、分散性、可变性、不良结构、复杂性等特征的应用问题。

1) 模块化:

- 最适合于被分成自然模块的应用问题。
- 一个Agent有自己的状态变量集，不同于环境的状态变量。
- Agent状态变量的一些子集与环境状态变量的一些子集相耦合，以提供输入输出。
- 重用性增加。

Agent的作用

2) 分散化:

一个Agent不需要外部的激励，可自主地监视自己的环境，并在认为合适的时候采取行动。

●Agent的分散性特性适合能被分解为独立进程的分布式应用。
独立进程——在不需要其他进程连续指导情况下能够做有用的事。

Agent的作用

3) 可变性:

因为Agent 非常适合于模块化和分散化的问题，当一个问题可能经常变化时，Agent的两个特征结合在一起会使它们具有特殊的价值。

➤ 模块化允许部分修改，

➤ 分散化使得模块变化对其他模块行为的影响达到最小。

一个系统快速、经常变化，且无不良边际影响的能力，对系统的竞争力越来越重要。

Agent的作用

4) 不良结构:

不良结构——在系统设计时，并非所有必须的结构化信息能够得到。Agent自然地支持这样的应用。

●仅需要识别系统中实体的类和它们对环境的影响，能够与改变环境的其它任何Agent进行适当的交互。

5) 复杂性:

➤衡量系统复杂性的一种方法——系统必须演示的不同行为的数目；

➤Agent体系结构可以将组合行为空间的发生从设计时移到运行时，急剧地减少必须要设计的软件代码数，进而降低构造系统的成本。

基于Agent的建模思想

□ Agent技术发展和应用的两个基本推动力：

1) 无论是现在还是将来，计算机科学及其应用领域内，由Agent组成的MAS有能力扮演重要的角色。

因为现在的计算机平台和信息环境都是分布的、开放和异构的，计算机不再是一个独立的系统，而是越来越多的与其他的计算机及它们的用户紧密的联系在一起。

2) 在建立和分析人类社会中的交互模型和理论方面，MAS也可以扮演重要的角色。

基于Agent的建模思想

□ 基于Agent建模的思想：

- 将Agent作为系统的基本抽象单位，必要的时候可赋予Agent一定的智能（Intelligent）；
- 然后在多个Agent之间设置具体的交互(Interact)方式，从而得到相应系统的模型。
- Agent、智能和交互——是基于Agent建模思想中最基本也是最重要的内容。

基于Agent的建模思想

□ Agent 在建模中的角色:

- 1) Agent是一个自治的计算实体。
- 2) 智能性——Agent在变化的环境中灵活而有理性地运作，具有感知和效应的能力。
- 3) 互交能力——Agent可以被其他为追求自己的子目标而执行相应任务的Agent所影响。

● 由于将Agent看成是主动对象，基于Agent的建模技术完全可以从面向对象技术中继承并发展。

识别个体Agent

□在一般情况下，建立系统模型是由一群Agent组成的MAS，可以用三个层次结构来描述：

- 1) Agent层：系统中所有反映问题域和系统责任的Agent。
- 2) 个体Agent（特征模型层）：Agent的结构与特征，包括内部状态（数据、变量）和行为规则（函数、方法等）。
- 3) MAS层：组成系统的Agent群体所采用的体系结构，主要要解决的问题是Agent之间的通信与协调等问题。

基于Agent
的系统模
型的层次
如图8.1所
示。



图8.1 基于Agent的系统模型的层次

识别个体Agent

- 给定系统有确定的系统问题和系统边界，识别个体Agent的任务就是解决这样的问题：
- 将系统中的什么映射作为Agent？也就是对系统进行Agent抽象。

Agent抽象的基本原则：从系统的物理结构出发，围绕着系统的目标来对系统进行抽象。

- 以系统的物理结构作为抽象的基本点，可根据物理世界的实际构成来划分Agent。

识别个体Agent

● **一般的处理原则：**将组成系统的每个实体都抽象为一个Agent（实体Agent）。这对自然的分布式系统尤为实用。

要注意两个问题：

1) 异质Agent与同质Agent的处理

- 实体之间可能是异质的，存在本质上的区别，如经济系统中的人、企业与政府等；
- 有些实体之间是同质的，在本质是相同的，如一个生物种群中多个生物个体。

● **处理方法：**将异质Agent分别形成相应的Agent类，
而将同质的多个Agent抽象归结为一个Agent类。

识别个体Agent

2) 抽象的粒度——抽象的层次

在确定了实体Agent后，有时为了实现系统的目标，还要设计一些其他的辅助Agent——集中服务Agent。

➤ **实体Agent**：一般来说，组成Agent群体的多个Agent有共同的状态和行为，由此抽象出这类Agent。

➤ **辅助Agent**：为一个Agent群体提供某些共同的服务，或提供有关这个Agent群体的信息。

➤ **移动Agent**：为物理上分布的系统提供传输信息或执行特定的功能。

识别个体Agent

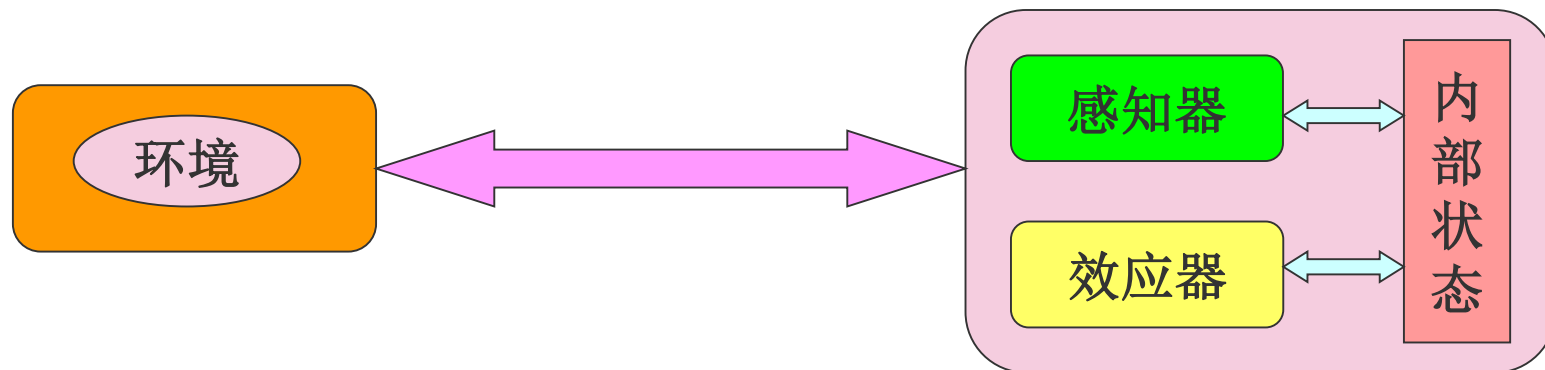
□ 经过上面的处理后，**确定组成系统的所有Agent，建立系统的Agent类图**，在实际的分析与建模的过程中，可根据需要反复进行这一过程。

□ 个体Agent的建模需解决的问题：

- 1) Agent由那些模块组成？
- 2) 模块之间如何交互信息？
- 3) Agent感知到的信息如何影响行为和内部状态？
- 4) 如何将这此模块用软件或硬件的方式组合起来形成有机整体，真正实现主体。

Agent的通用模型

就目前来说，可以将Agent视为由环境、感知器和效应器三部分组成，Agent 的通用模型如图8.2所示。



- 1) 每个Agent都有自己的状态。
- 2) 每个Agent都拥有一个感知器来感知环境，根据环境的状态来改变自己状态的方法。
- 3) 每个Agent都拥有一个效应器作用于环境，用来改变环境状态的方法。

Agent的通用模型

从计算的角度看，主体是一个计算实体，

- 具有属于自身的资源，
- 能够感知环境信息，
- 根据 内部的行为控制机制 确定主体应采取的行动，
- 主体的行动实施后，将对自身状态和环境状态产生影响。



Agent形式化

- 假设环境变化可以抽象为一个环境状态序列，环境在任何离散的瞬时状态的有穷集合为：

$$E = \{e_0, e_1, e_2, \dots\}$$

- 主体有一个可执行动作集合 $A = \{a_0, a_1, a_2, \dots\}$

- 主体在环境中的一次执行r是环境状态与主体动作的一个交替序列：

$$r : e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} \dots \xrightarrow{a_{u-1}} e_u$$

- 主体的动作决策部件可以定义为以下函数： $Choose : E^* \rightarrow A$

E^* 为环境演化的状态序列。

Agent形式化

- 主体的动作将对环境状态产生影响，定义影响函数为：

$$Change: E \times A \rightarrow \wp(E)$$

- 标准主体定义为以下三元组：

$$Agent = \langle A, Choose, Change \rangle$$

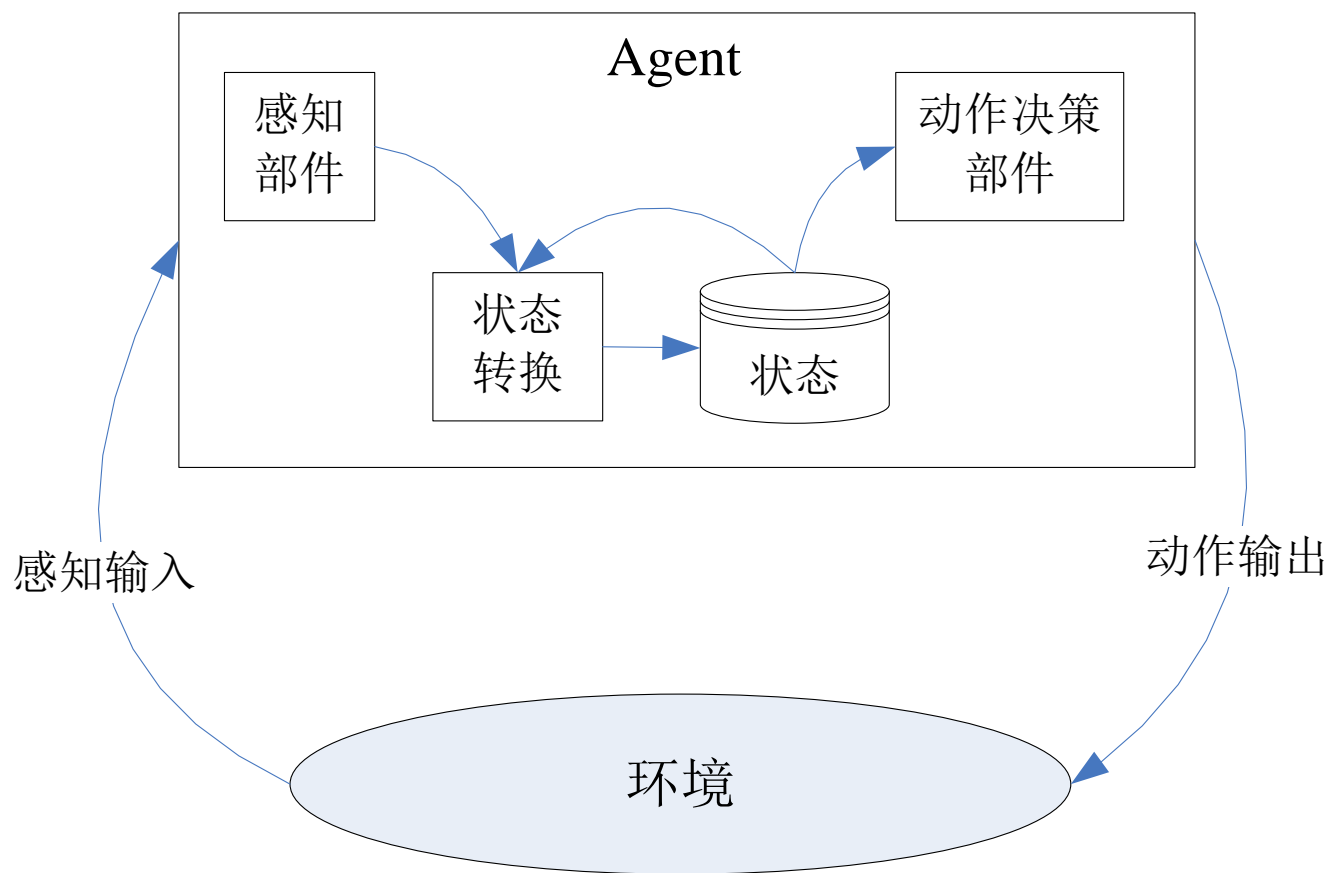
慎思Agent（认知Agent、思考Agent）

——是一个显式的符号模型，包括环境和智能行为的逻辑推理能力，具有信念-期望-意图（Beliefs-Desire-Intentions, BDI）结构，它保留经典人工智能传统（数理逻辑的符合学派），是一种基于知识的系统。

环境模型一般是预先实现的，形成主要部件——知识库。

□ 慎思Agent是具有内部状态的主动软件，具有知识表示、问题求解表示、环境表示、具体通信协议等。

慎思Agent（认知Agent、思考Agent）



Agent的慎思结构反映了传统人工智能的特点。

慎思Agent（认知Agent、思考Agent）

慎思Agent要面对两个基本问题：

1) **转换问题**：如何在一定的时间内将现实世界翻译成一个准确的、合适的符号描述——计算机视觉、自然语言理解等领域的研究；

2) **表示/推理问题**：如何用符号表示复杂的现实中的实体和过程，以及如何让Agent在一定时间内根据这些信息进行推理做出决策——知识表示、自动推理、自动规划等领域的研究。

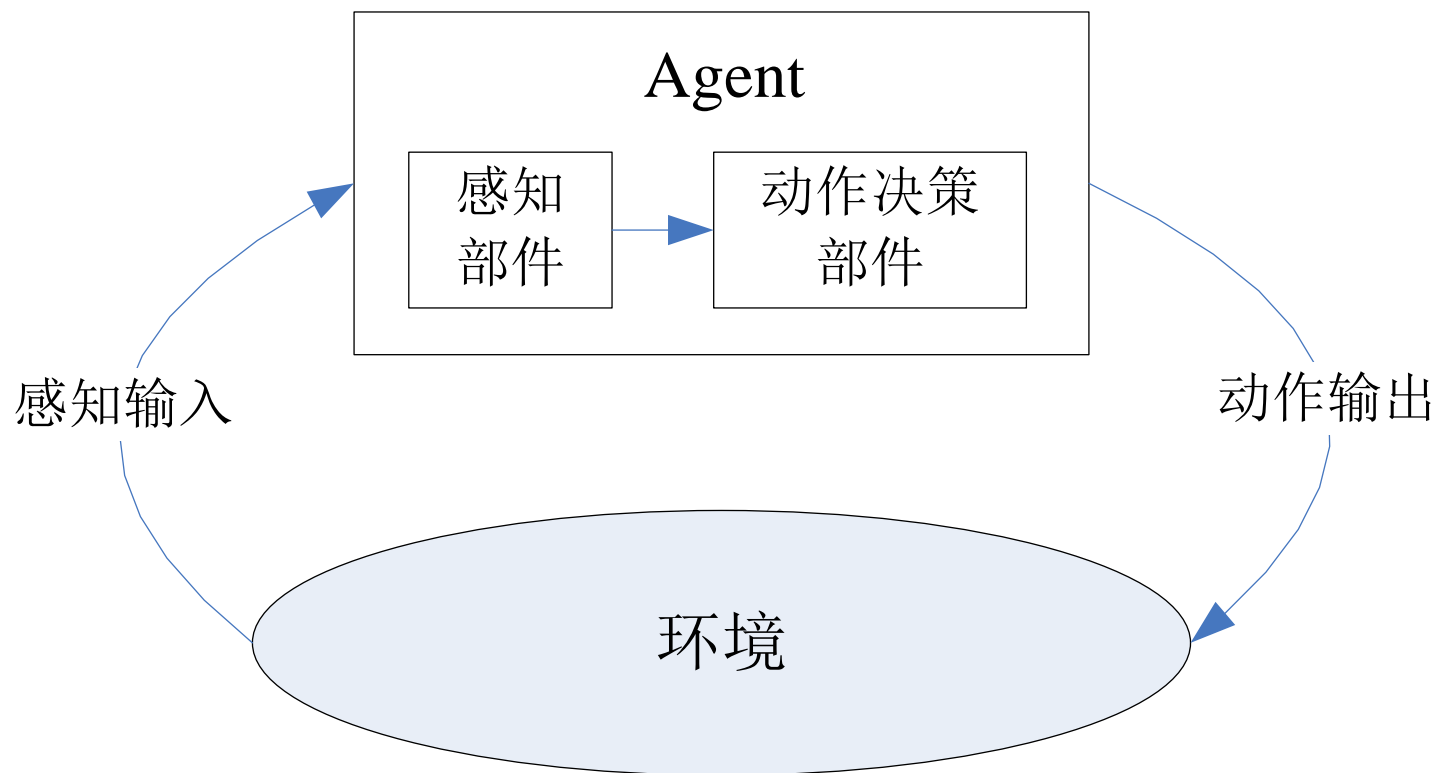
□ **慎思Agent缺陷：**

- 1) 由于表示的复杂性，慎思Agent适应动态环境有一定局限性；
- 2) 没有必要的知识和资源，在Agent执行时，加入有关环境的新信息和知识到已有的模型中比较困难。

反应Agent

- 采用应激响应的行动方式对其所处环境的当前状态进行响应。（没有符号表示的世界模型，不使用复杂的符号推理，不对环境进行描述）
- 通过与其他Agent简单交互，使Agent系统表现出复杂的整体行为。（不考虑历史情况，也不为未来制定计划）
- 具有坚定性和容错能力两大重要属性。（响应速度较快，不需要进行复杂推理和费时思考）

反应Agent



Agent的反应结构反映了基于行为的人工智能的特点。

混合型Agent

□ 混合结构的Agent系统融合经典和非经典的人工智能，在一个Agent中包含两个（或多个）子系统：

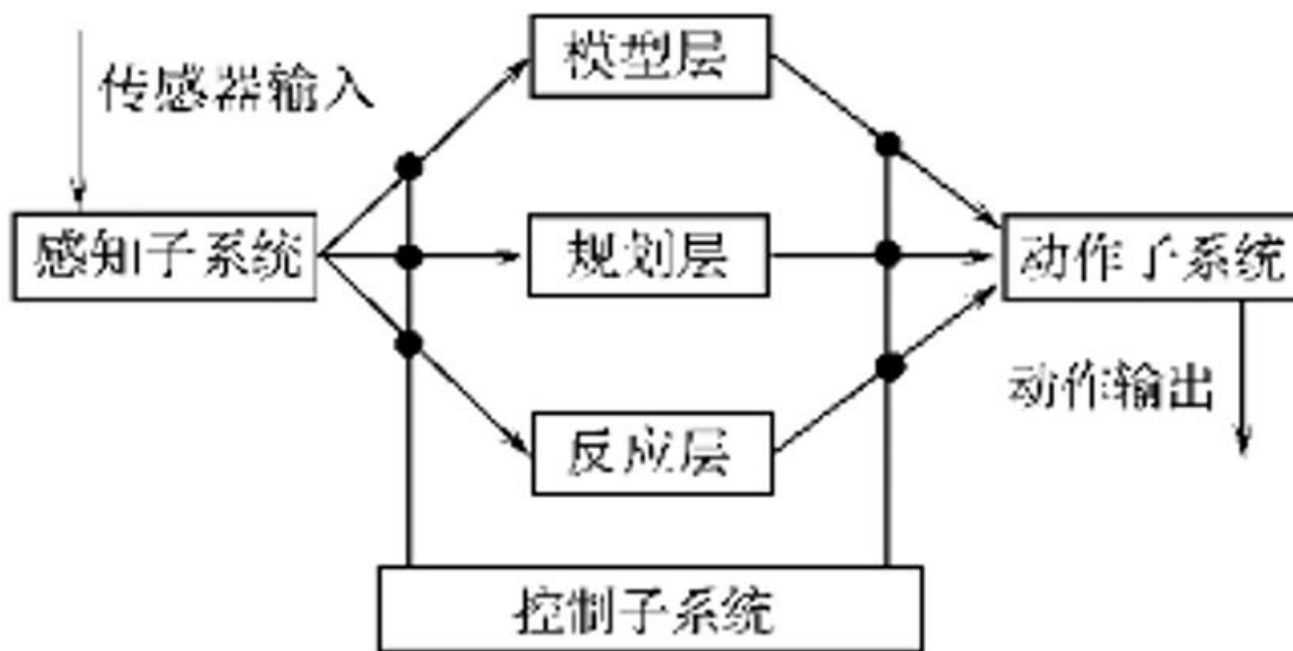
➤1) 慎思子系统：用符号表示的世界模型，用主流人工智能中提出的方法生成规则和决策；

➤2) 反应子系统：不经过复杂的推理就对环境中出现的事件提供反应。

➤3) 反应子系统优先级高于慎思子系统，以便对环境中出现的重要事件提供快速反应。

混合型Agent

- 混合型Agent采用一个层次结构，底层以反应Agent为主，高层以慎思Agent为主。
- 一个典型的结构模型如下图所示。



多Agent系统的体系结构

□ 为了建立由多个Agent组成的完整的系统模型，确定多Agent系统的体系结构，就要处理好以下5个问题：

1) 系统应有多少个Agent?

根据系统的目标要求，确定各种Agent的总数以及系统运行时Agent的数目是否可改变。

2) Agent之间采用什么样的通信渠道?

通常在传输介质（共享物理环境与数字网络）、访问（广播、面向目标、Agent到Agent）等方面可能有所不同。

多Agent系统的体系结构

3) Agent之间采用什么样的通信协议? 通信协议决定了被建立的Agent之间如何交流。

□ 通常采用的通信方式:

- 共享全局存储器 (如黑板机制) ;
- 消息传递;
- 两者的结合。

多Agent系统的体系结构

4) 怎样建立Agent与其相关的其他Agent之间的结构?

一个Agent群体的结构描述每个Agent的直接的熟人 (Agent) 和它们之间由于信息和物料流动等原因而产生的拓扑结构, 如分层嵌套结构、网络结构等。

5) Agent之间如何协调它们的行动?

多Agent系统的体系结构

- 要解决好上述5个问题的前提条件：对物理系统的透彻了解和对系统目标的准确把握。
- 在这个阶段：
 - 对相关技术的了解（如面向对象的消息传递、常用的合同网协议、耗散机制等）也会促进问题的解决，
 - 可能会根据需要返回到前两个阶段进行再分析（Agent层、个体Agent特征模型）。
- 问题解决后，我们就建立了一个完整的多 Agent系统模型。

多Agent系统的体系结构

- ❑ 多Agent系统的体系结构的核心：解决Agent之间的通信与协作问题。
- ❑ 在多Agent系统中，各个Agent通过相互间的消息发送和接收来协同工作。
 - 通信机制——使得各个Agent能够相互传递消息；
 - 协作机制——使得各个Agent根据运行过程中所传递的消息，协调彼此的行动，实现合作。
- ❑ 通信是协作的基础，通信方法分为黑板系统和消息/对话系统。

多Agent系统的体系结构



Agent通信语言

Agent通信语言：

是一种用于表达Agent之间交互消息的描述性语言，
它定义了交互消息的格式（**语法**）和内涵（**语义**）。

影响较大的Agent通信语言：

- KQML
- ACL

通信方式

Agent之间常用的通信机制有三种：

- 黑板机制
- 邮箱机制
- 消息传递机制

交互协议

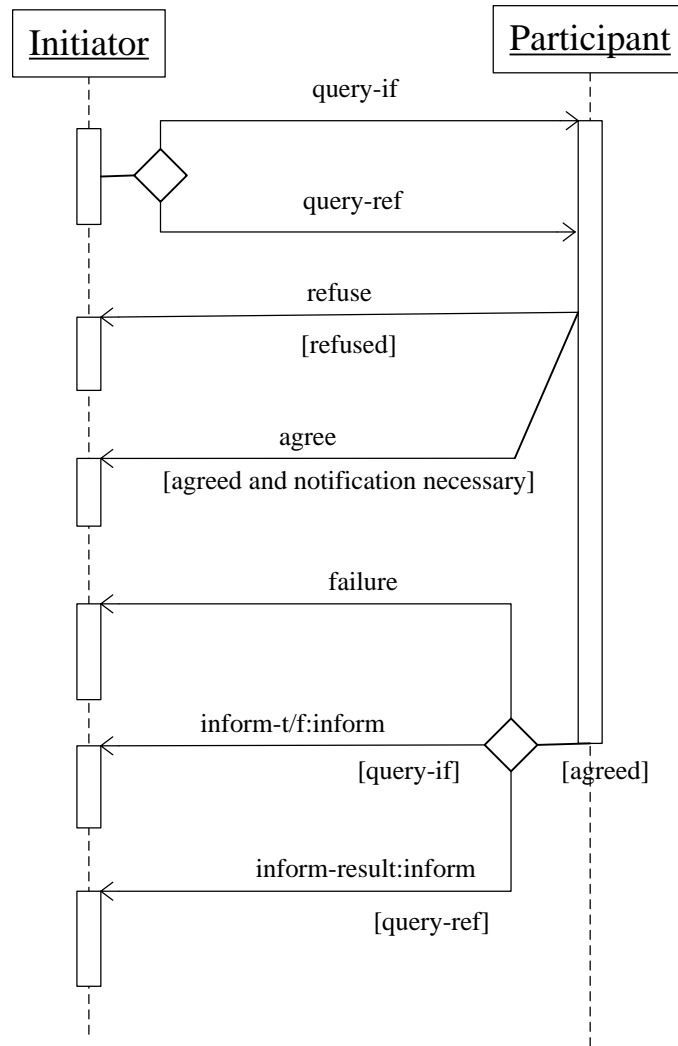
交互协议定义了Agent之间为了进行协作，实现某个特定目标而进行交互的结构化消息。

FIPA对一些典型的对话定义了交互协议：

- 请求 (request) ；
- 查询 (query) ；
- 合同网 (contract-net) ；
- 代理 (broking) ；
- 订阅 (subscribe) ；
- 建议 (propose) 等。

注：FIPA(The Foundation for Intelligent Physical Agents)是一个由活跃在Agent领域的公司和学术机构组成的国际组织，其目标是为异构的Agent和移动Agent系统之间能够互操作而制订相关的软件标准。

例：查询交互协议



多Agent系统的体系结构

□ 在开放、动态的多Agent环境下，具有不同目标、资源能力限制的多个Agent必须对其目标、资源的使用进行协调才能保证合作的有序进行。协调是手段，合作是目的。

□ 多Agent协调机制可以分为：

- 组织结构协调
- 计算市场模型
- 基于对策论的协商模型
- 基于合同网的协调机制
- 基于规划的协调
- 社会规范协调。

Agent的实现

□ 在实现系统模型之前，一个重要任务是开发平台的选择。

由于目前对Agent还没有一个统一的定义，所出现的Agent开发工具也没有可遵循的统一标准，大都是按自己的理解和需要，采用Smalltalk、C++和Objective C等语言打包而成的Agent模型。

● 目前在复杂系统的研究中，被广泛推荐和采用的基于Agent的建模和开发工具是由SFI研制的Swarm。Anylogic

□ 选择开发工具和建立系统模型相结合，反复调整直至比较完善。

Agent的实现

- 在具体实现的Agent可以用类似于下图8.3实体Agent类和图8.4集中服务Agent的伪码来简单地表示。

- **Agent_Entity:**
- **Begin**
-
- **States:**
- **Private_Preferences;**
- **Private_Variables;**
- **Public_Variables;**
- **And so on**
- **Perceptions:**
-
- **Accept_Information_Methods;**
- **Get_Information_Methods;**
- **And so on**
- **Behaviors:**
- **Make_Decisions;**
- **Compute_Internal_Variables;**
- **Draw_self;**
- **And so on**
- **End**

2022/12/21

图8.3典型实体Agent模型

```

Agent_Population :
Begin :
States :
    Internal_Represent_of_collection;
    Current_Active_Agent;
    Number_of_Agents;
    And so on;

Perceptions :
    Accept_Information_Methods;
    Get_Information_Methods;
    And so on;

Behaviors :
    Private Behaviors :
        Get_Nth_Agent(N);
        Randomize_Agents;
        And so on;
    Public Behaviors :
        Initialize;
        Special_Service_Methods;
        And so on;

End
  
```

59

图8.4典型集中服务Agent程序模型

复习题

- 基于Agent的系统模型的层次
- 个体Agent的建模需解决的问题，识别个体Agent
- Agent的通用模型及其Agent形式化
- 慎思Agent、反应Agent和混合型Agent
- 多Agent系统的体系结构，需要处理的5个问题