

# Artificial Intelligence

## 人工智能

### 第6章 Agent



# 第6章 Agent

6. 1 概述

6. 2 Agent及其结构

6. 3 协调与协作

6. 4 Agent研究方向



## Agent(智能体)的概念与示例

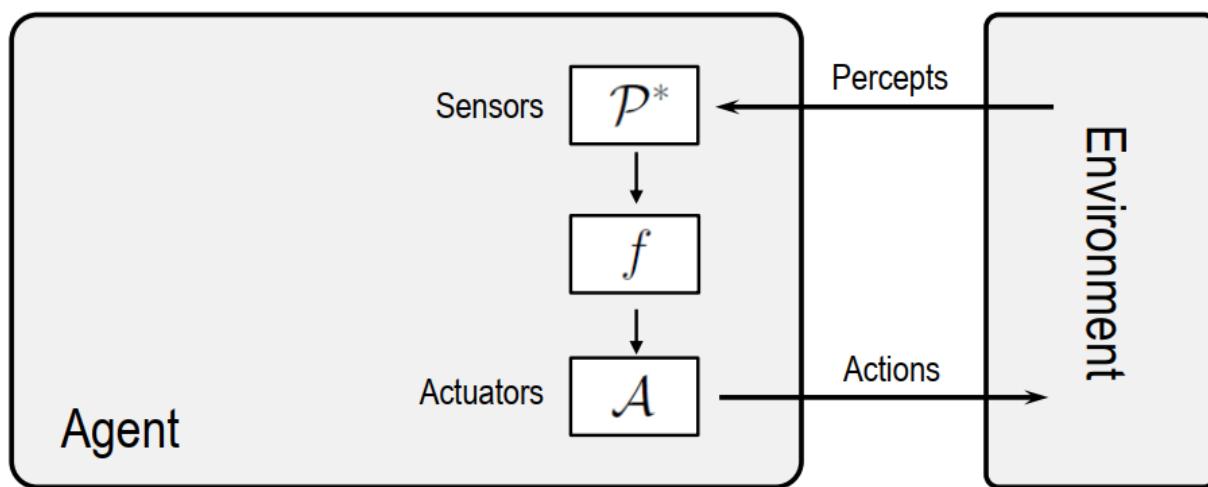
- 智能体概念应用广泛，在不同的研究领域对它有不同的理解。
- 在人工智能领域，智能体是指驻留在环境下能够自主、灵活地实施行为以满足设计目标的行为实体。
- 具体的，智能体具有几个方面的特征：
  - 智能体是行为实体
  - 智能体驻留在环境中（驻留性）
  - 智能体行为具有自主性
  - 智能体行为具有不同程度的灵活性

# 概述

## 数学定义

- 一个智能体的行为可以数学上被描述为一个智能体函数，将每个感知映射为动作：

$$f : \mathcal{P}^* \rightarrow \mathcal{A} \quad \text{where} \quad \mathcal{P}^* = \sum_{t=1}^T |\mathcal{P}|^t$$





# 概述

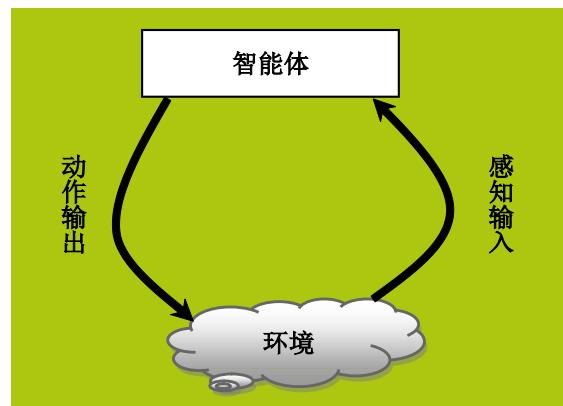
## 智能体是行为实体

- 智能体是具有行为能力的实体，它具有一组动作并能执行这些动作，是动作执行的拥有者和实施者。智能体的动作反应了智能体所具有的能力，这种能力主要体现为二个不同的方面。
  - 从内部角度上看，智能体动作的实施将有助于实现智能体的任务和目标；
  - 从外在行为上看，智能体的动作实施将对智能体所驻留的环境产生影响。
- 一般地，不具备行为实施能力的实体通常不被视为是智能体。如：数据、资源、信息等。
- 负责对数据、资源、信息等进行管理、维护的实体可被视为是智能体。例如，服务机器人是一个智能体，它具有行走、转弯、抓取物体等行为。

# 概述

## 智能体驻留在环境中（驻留性）

- 任何智能体都不是孤立和封闭的，它驻留在一定的环境之中，需要与环境进行交互，获取环境及其变化信息，以此来指导其行为实施。
- 如下图所示，智能体与其驻留环境之间的交互主要表现为两个方面：
  - 通过感知输入获得环境变化
  - 通过动作输出来影响环境。





# 概述

## 智能体行为具有自主性

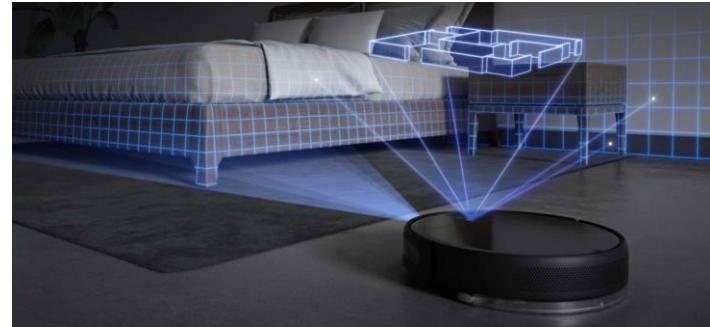
- 智能体能在没有人类或其他智能体的干涉和指导下运行，并能根据其内部状态和感知到的环境信息来决定该实施什么样的动作，进而控制自身的行为。
- 简单的讲，智能体行为的自主性是指智能体在适当的场景下知道该选择什么样的动作来执行。

# 概述

## 智能体行为具有不同程度的灵活性

- 智能体行为的灵活性主要体现为以下三方面：

- 反应性 (Reactive)
- 自发性 (Pro-active)
- 社会性 (Social)



- 反应性是指智能体能够感知所处的环境，并能对环境中发生的相关事件（比如智能体间的交互、用户的指令等）作出适时反应，以满足系统设计目标。

例子：家庭服务机器人一旦遇到障碍物，它需要立即实施躲避行为，以免发生碰撞。

# 概述

- **自发性**是指智能体不仅具有目标并根据目标行事，而且能够主动地产生目标，进而实施自发的行为。

例子：一旦发现有陌生人访问家庭，服务机器人会自发产生一个目标：将来访者信息告知用户，进而主动实施一组行为，如呼叫用户的手机、给用户发送短信等。



- **社会性**是指智能体所驻留的环境可能存在其他智能体，它拥有这些智能体的有关信息和知识（比如这些智能体的物理位置、所拥有的资源和能力等），并能与它们进行交互和通信，实现灵活多样和复杂的**合作**、**协商**和**竞争**，以满足系统的设计目标。

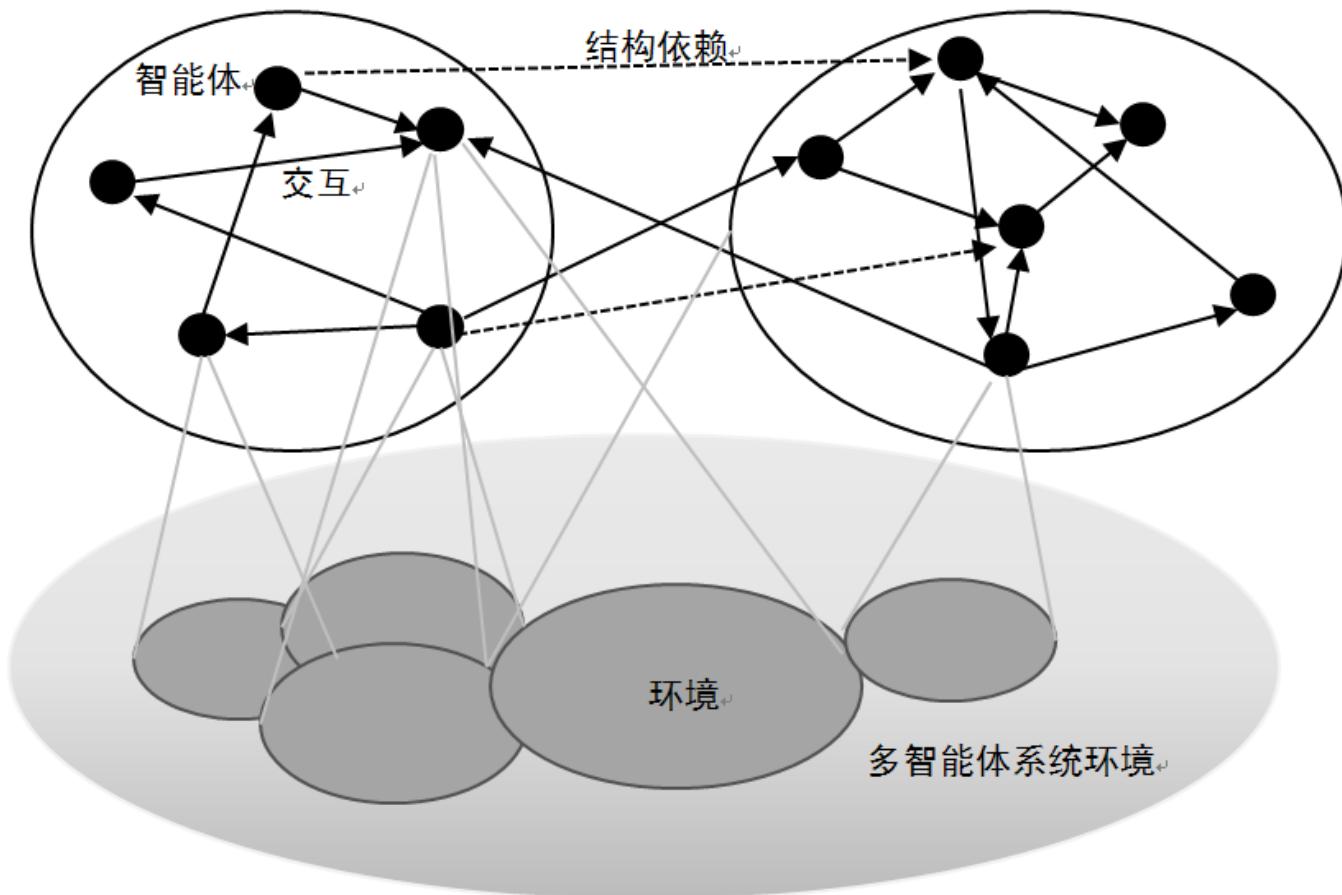
例子：家庭服务机器人能够与互联网上的气象信息提供智能体进行交互，从而获取天气信息。

# 多智能体系统的概念与示例

- **多智能体系统**是指由多个相对独立同时又相互作用的智能体所构成的系统。
  - 独立性：每个智能体都是自主的行为实体，封装了行为以及行为控制机制，可以在无需外部指导下实施行为；
  - 协同性：这些智能体并不是孤立的，它们之间存在各种关系，需要相互交互和协同进而达成问题的求解。



# 概述



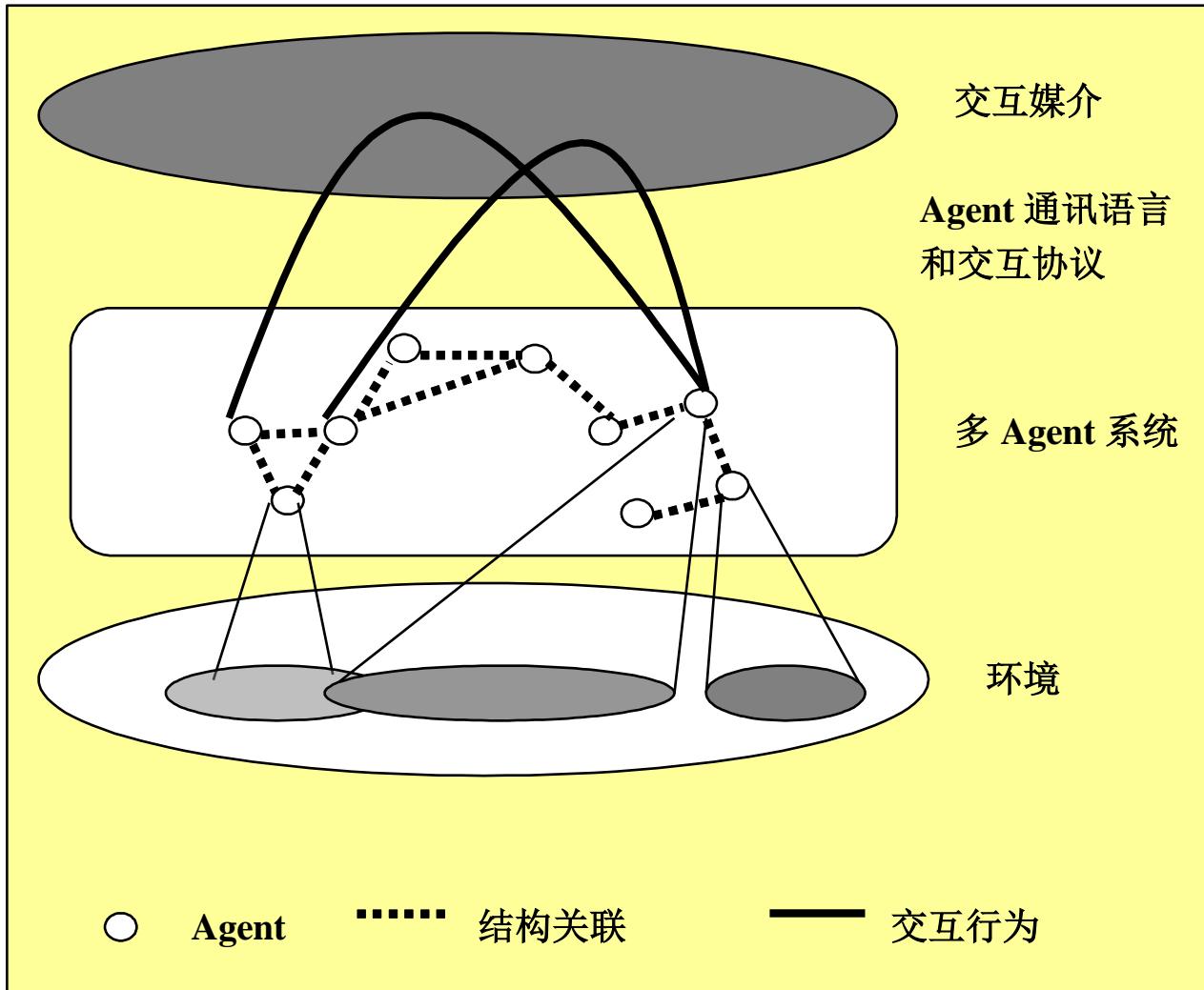
图：多智能体系统的高层视角



## 多智能体系统的特性

- **无全局视角：**对于绝大多数多智能体系统而言，系统中的智能体要获得关于系统以及环境的完整、准确和及时的信息是非常困难的。这是由于智能体所驻留的环境可能会比较复杂，具有以下特点：
  - 开放性
  - 不确定性
  - 动态性等特点。
- **不可预测和不确定性：**在多智能体系统中，由于智能体行为的自主性、智能体行为对环境影响的有限性、系统中无全局控制智能体、智能体间交互行为的动态性，多智能体系统的运行具有不可预测性和不确定性的特点。

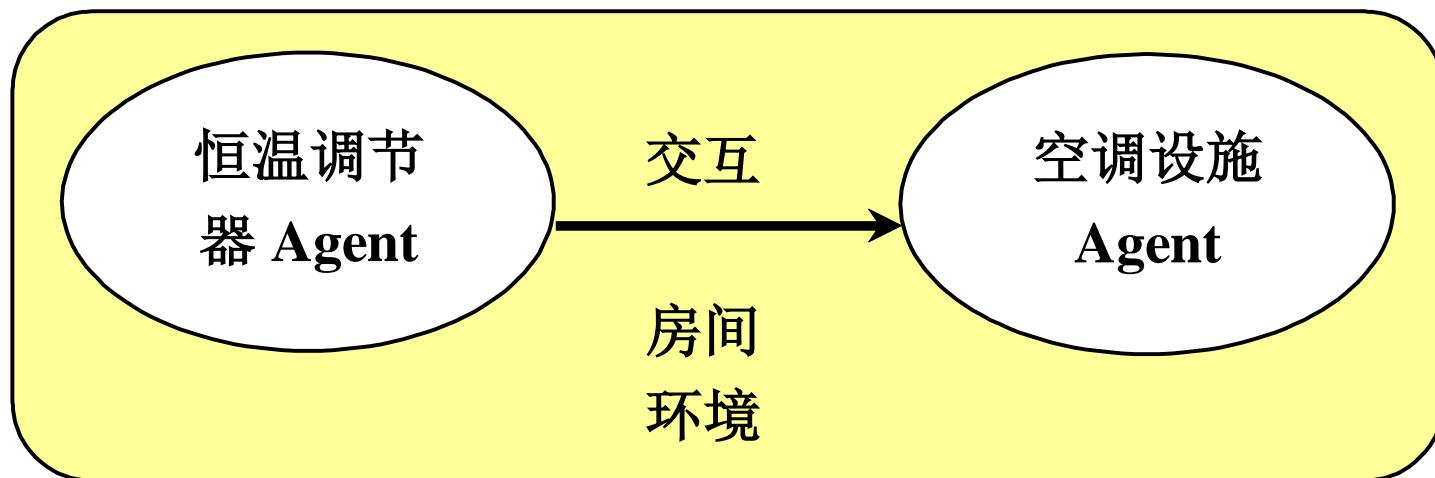
# 概述



图：多智能体系统的结构

# 例子1：物理多智能体系统

- 恒温调控系统
  - 恒温调节器Agent
  - 空调设施Agent



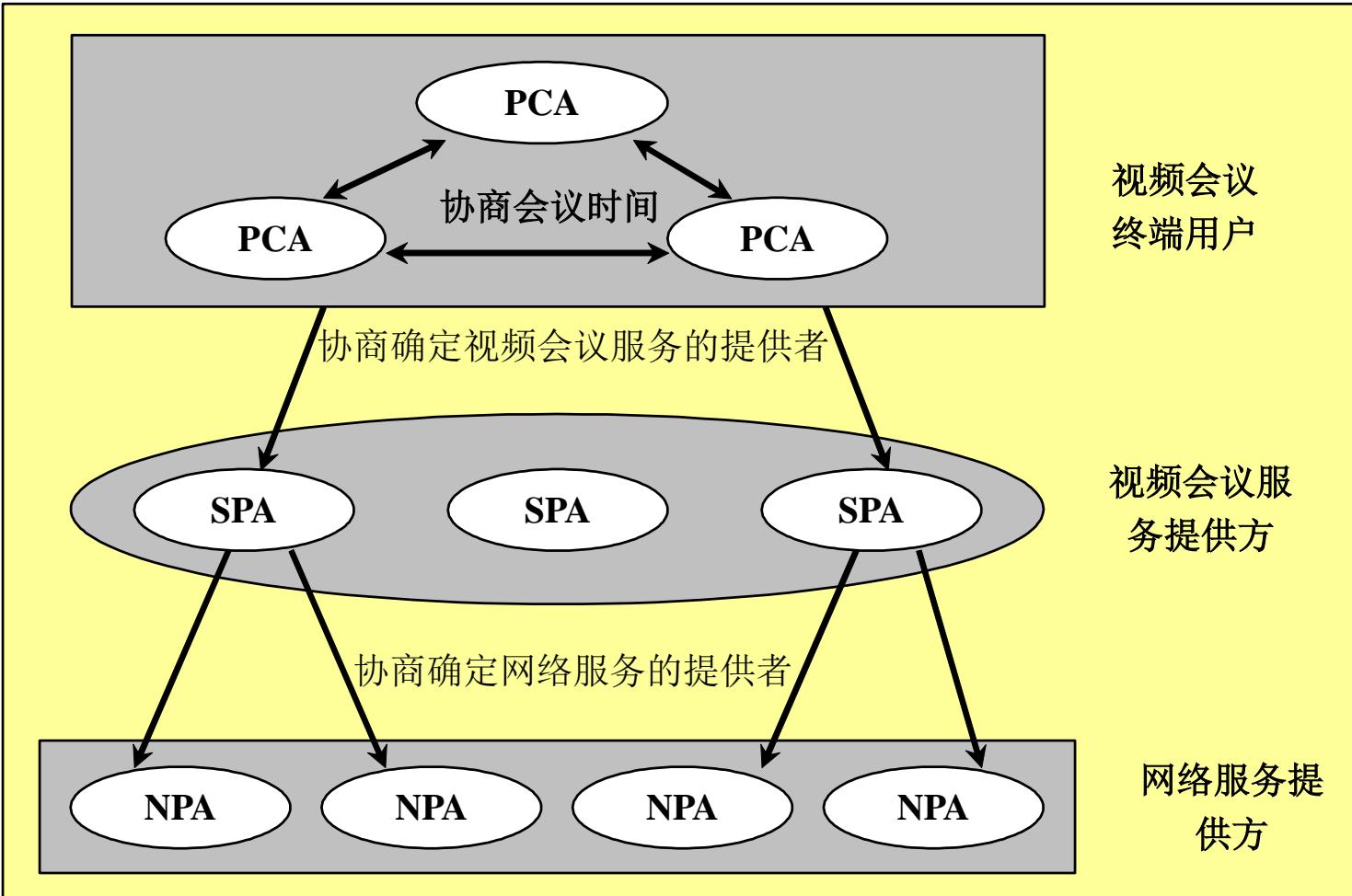
## 例子2：软件多智能体系统

- 计算机病毒检测和清除软件系统
  - 系统包含的Agent
    - 文件实时防护Agent
    - 病毒分析和清除Agent
    - 病毒数据维护Agent

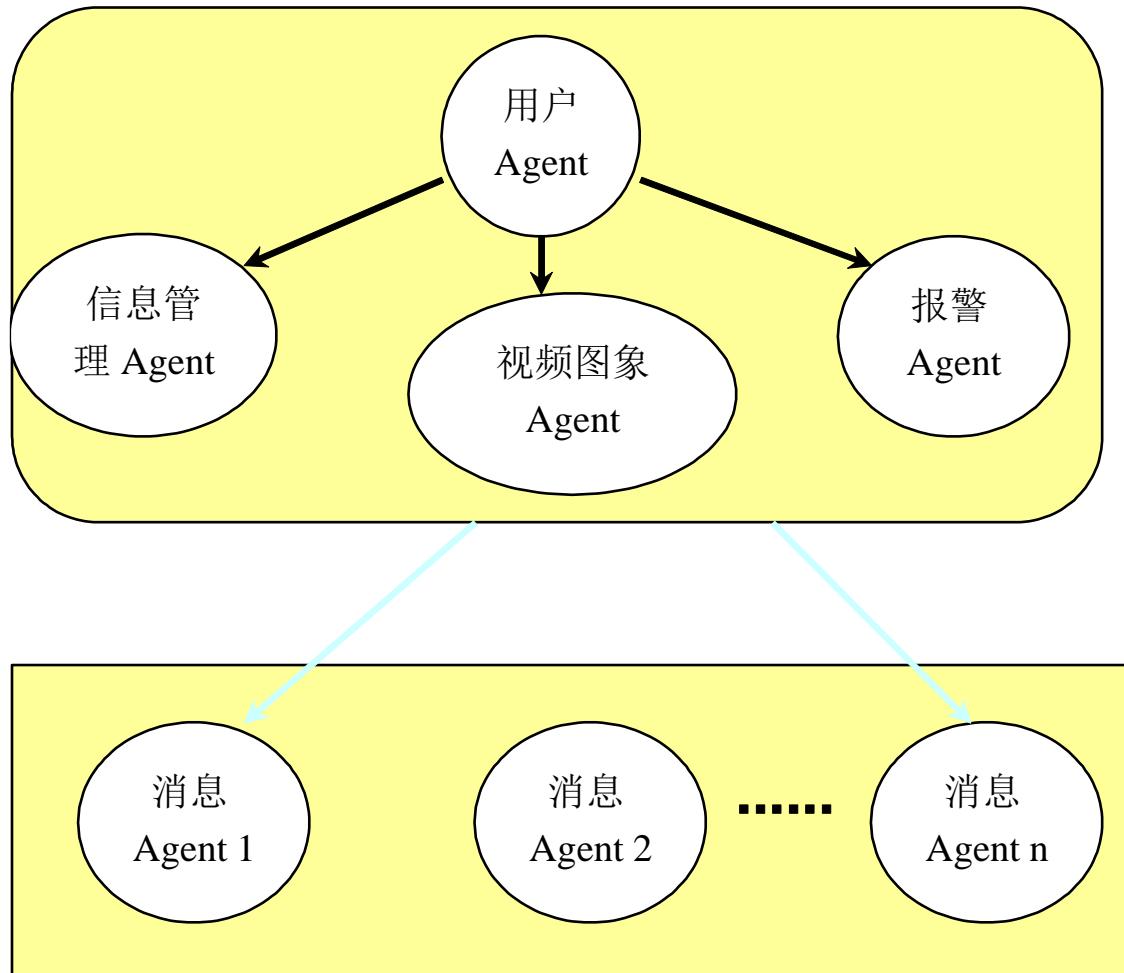


# 概述

## 例子3：虚拟私人网络



## 例子4:家庭智能网络





# Agent及其结构

## 第6章 Agent

6. 1 概述

6. 2 Agent及其结构

6. 3 协调与协作

6. 4 Agent研究方向



# Agent及其结构

## 6.2 Agent及其结构

6. 2. 1 Agent定义

6. 2. 2 Agent要素及特性

6. 2. 3 Agent的结构特点

6. 2. 4 Agent的结构分类



# Agent及其结构

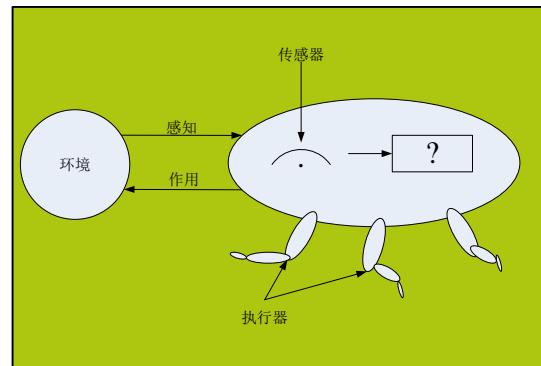
## 6.2.1 Agent定义

- 所谓Agent是指驻留在某一环境下能够自主（Autonomous）、灵活（Flexible）地执行动作以满足设计目标的行为实体。
- 上述定义具有如下二个特点。
  - (1) 定义方式。Agent概念定义是基于Agent的外部可观察行为特征，而不是其内部的结构。
  - (2) 抽象层次。Agent概念更加贴近于人们对现实世界（而不是计算机世界）中行为实体的理解。

# Agent及其结构

## 6.2.2 Agent要素及特性

- 1. Agent的要素
- Agent必须利用知识修改其内部状态(心理状态),以适应环境变化和协作求解的需要。
- 人类心理状态的要素可分为以下三种
  - 认知(信念、知识、学习等)
  - 情感(愿望、兴趣、爱好等)
  - 意向(意图、目标和承诺等)
- 受人类心理启发,传统Agent理论模型研究的主要方向为:研究信念(belief)、愿望(desire)和意图(intention)的关系及其形式化描述,力图建立Agent的BDI(信念、愿望和意图)模型。





# Agent及其结构

- 2. Agent的特性
- Agent与分布式人工智能系统一样具有协作性、适应性等特性。此外，Agent还具有自主性、交互性以及持续性等重要性质。
  - (1) 行为自主性
  - (2) 作用交互性（也称反应性）
  - (3) 环境协调性
  - (4) 面向目标性
  - (5) 存在社会性
  - (6) 工作协作性
  - (7) 运行持续性
  - (8) 系统适应性
  - (9) 结构分布性
  - (10) 功能智能性



## 6.2.3 Agent的结构特点

- 计算机系统为Agent的开发和运行提供软件和硬件环境支持，使各个Agent依据全局状态协调地完成各项任务。具体地说：
  - (1) 在计算机系统中，Agent相当于一个独立的功能模块、独立的计算机应用系统，它含有独立的外部设备、输入输出驱动装备、操作系统、数据结构和相应的输出。
  - (2) Agent程序的核心部分叫做决策生成器或问题求解器，起到主控作用。
  - (3) Agent的运行是一个或多个进程，并接受总体调度。
  - (4) 各个Agent在多个计算机CPU上并行运行，其运行环境由体系结构支持。



## 6.2.4 Agent的结构分类

### (1) 知识型体系结构:

- 具有表示智能体状态的部件,
- 状态表现为智能体所具有的知识
- 智能体的行为决策是一个基于其知识的定理证明过程。

### (2) 反应型体系结构:

- 不具有表示智能体状态的部件，而是定义了一系列的反应式规则
- 智能体根据感知到的环境信息，触发相应的反应式规则来执行，因而其动作决策实际上是一个从情景到动作的直接映射。

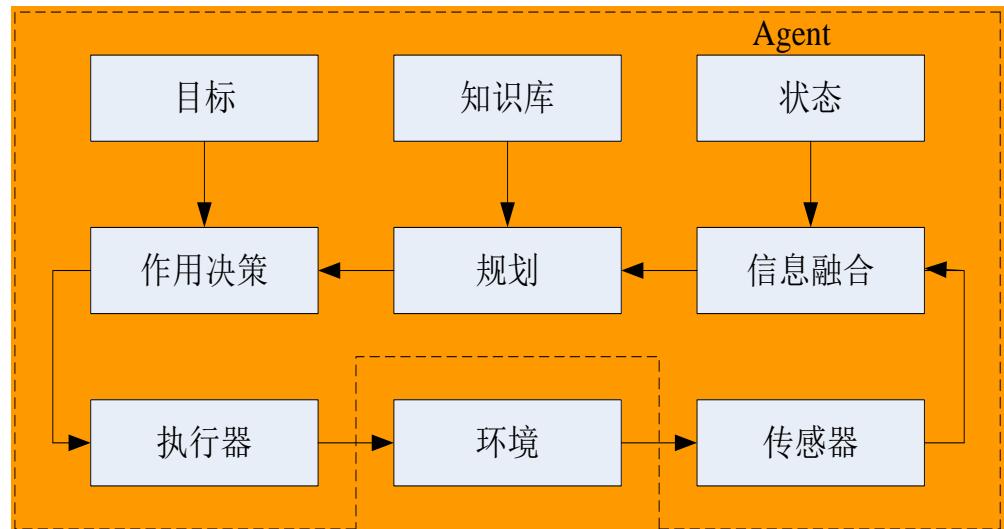
### (3) 认知型体系结构:

- 具有表示智能体状态的部件
- 状态表现为智能体所具有的认知状态，如信念、期望、意图等
- 智能体的自主行为决策是一个针对其内部认知状态的实用推理（Practical Reasoning）过程。

# Agent及其结构

## 知识型体系结构

- 将智能体视为一个定理证明器
  - 智能体的内部状态对应于智能体所拥有的知识
  - 各个知识条目用逻辑语言（比如经典的一阶谓词逻辑或者Hoare逻辑）来表示
  - 智能体的行为决策是一个基于知识库的定理证明过程。



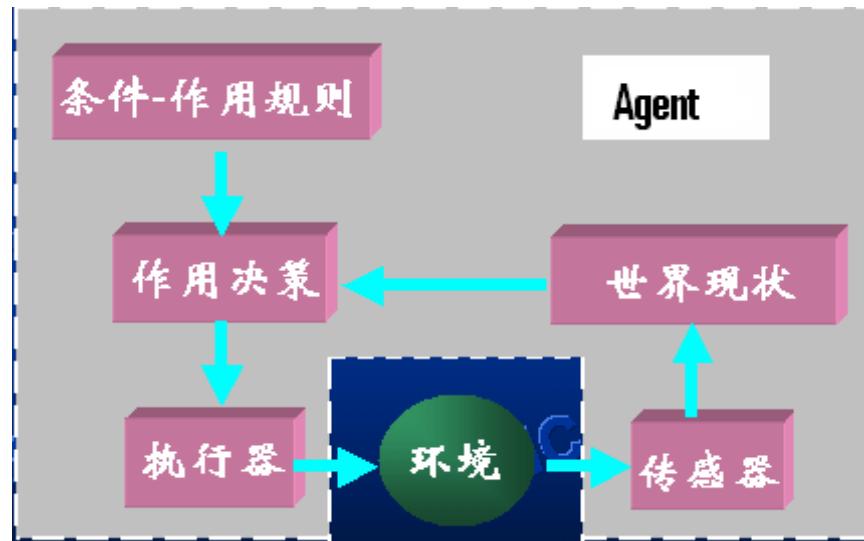
知识型体系结构的不足：

- 表达能力问题。
- 计算复杂性问题。
- 易用性问题。

# Agent及其结构

## 反应型体系结构

- 智能体对环境事件的响应是通过一组简单的反应式规则来实现的，这些规则描述了智能体的感知输入与响应动作之间的映射关系。
- 基于这些反应式规则，智能体无需通过复杂的推理和证明就可针对环境事件作出简单、快速地响应。





# Agent及其结构

- 智能体的反应型体系结构主要包含以下三个部件。
- **感知部件。** 感知部件用于感知外部环境和智能体内部发生的相关事件以及其他“刺激”。例如，用户发布的指令、监听到一个网络传输事件、发现的一个障碍物、系统时钟的一个定时事件等。
- **行为规则。** 行为规则定义了智能体的行为，也称格局规则 (Situation Rule)，描述了智能体的感知输入与响应动作之间的映射关系。一般的，智能体的行为规则具有以下的描述形式：

*if <格局> then <动作规约>*

- **控制系统。** 控制系统将依据智能体的行为以及感知器所感知到的环境事件选择智能体的动作来执行。



# Agent及其结构

## 认知型体系结构

- 意向观点(Intentional Stance)是指将智能体视为是一个由各种认知成分(如信念、意图等)构成的意向系统
  - 信念部件用于表示智能体对外部环境和内部自身的理解和认识
  - 意图部件体现了智能体欲实现的任务和目标，它将引导智能体生成规划、执行动作以实现智能体的意图
- 这些认知部件构成了智能体的内部数据结构，其取值定义了智能体的内部状态。
- 每一个认知部件都有其特定的功能，在智能体的自主运行过程中起到特定的作用。
- 构成智能体的这些认知部件之间相互作用，从而使得智能体能够表现出诸如自主性和灵活性等行为特征。



# Agent及其结构

- Rao 和Georgeff 提出的BDI (Belief-Desire-Intention) 体系结构包含了信念、期望和意图三个认知部件：
  - 信念定义了智能体对环境的理解和认识，它是智能体进行动作决策的基础；
  - 期望反应了智能体可能试图实现的任务和目标，体现了智能体的某种选择。期望经智能体承诺后将上升为智能体的意图，此处所述的承诺是指智能体将持续性地保持其意图及至某些条件发生。
  - 意图是理解、分析和描述智能体行为决策的一个重要的认知部件，意图概念的基本内涵是对未来动作的合理选择。智能体的意图将影响和约束智能体的行为决策和实施，是智能体执行动作的起因。



# Agent及其结构

- PRS (Procedural Reasoning System) 是最早使用BDI体系结构来开发智能体并成功用于各种应用的系统之一。
- PRS由美国斯坦福大学研究所的Michael Georgeff等人研发。自二十世纪八十年代中后期以来，研究人员已经利用不同的平台和程序设计语言开发出多个不同版本的PRS系统。
- 例如，澳大利亚人工智能研究所的DMARS，Michigan大学用C++实现的PRS系统UM-PRS，利用Java程序设计语言实现的Jam等。

认知型体系结构存在以下问题：

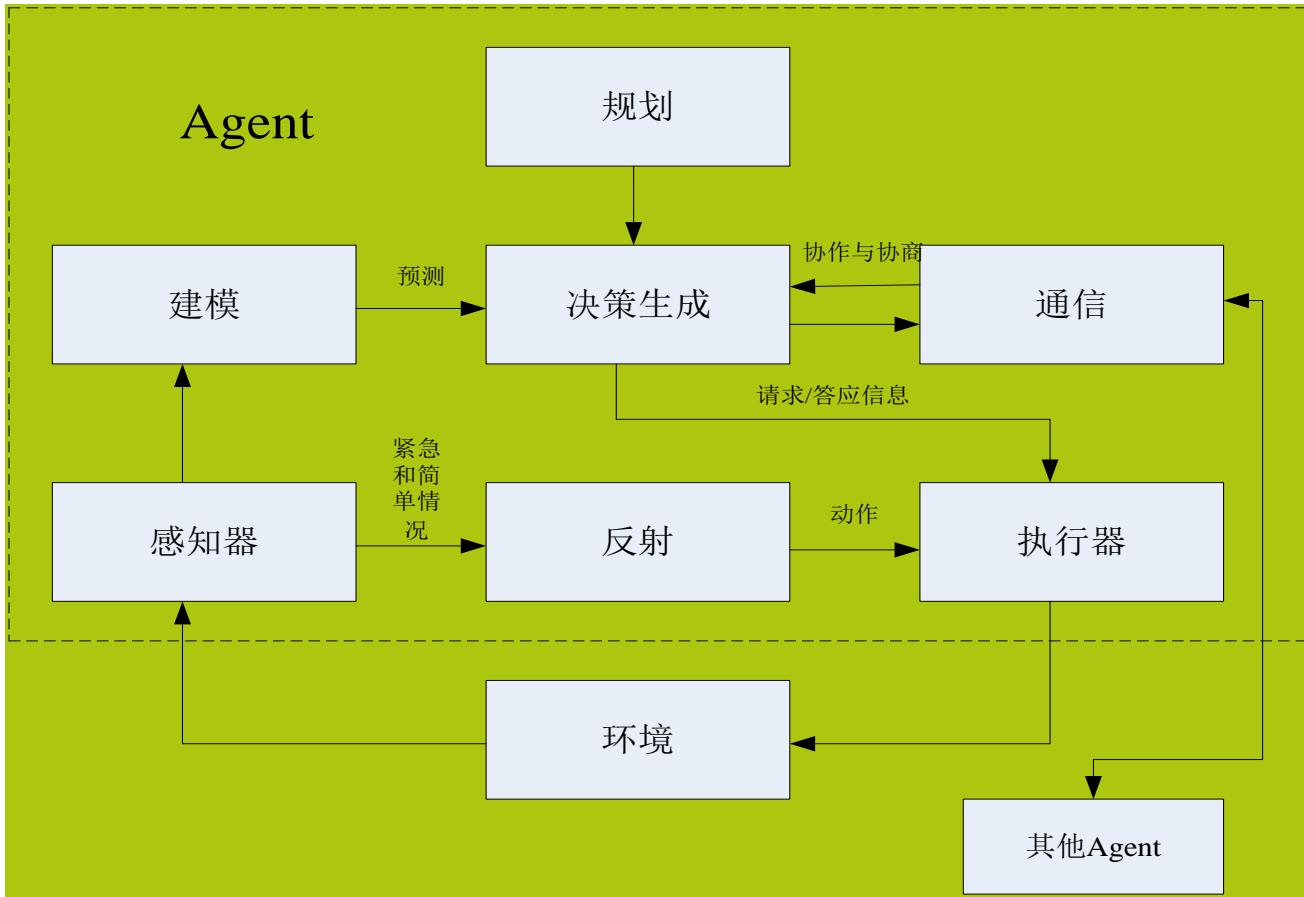
- 表达能力问题。
- 计算复杂性问题。



## 混合型体系结构

- 所谓混合型体系结构是指在一个智能体的内部集成了多个不同类型的实现型体系结构，组合了多个层次的行为决策部件
- 不同层次的实现型体系结构之间相互交互，共同对感知到的环境输入作出多种方式的响应
- 因此，智能体在运行时刻能够表现出多种形式的行为特征（如自主性、反应性、自发性等）

# Agent及其结构



图：混合型体系结构



# 第6章 Agent

6. 1 概述

6. 2 Agent及其结构

6. 3 协调与协作

6. 4 Agent研究分类



# 协调与协作

## 6.3 协调与协作

6.3.1 引言

6.3.2 合同网

6.3.3 协作规划



## 6. 3. 1 引言

多智能体系统的主要目标：

- 以自主的智能Agent为中心，使多Agent的知识、愿望、意图、规划、行动协调，以至达到协作。



# 协调与协作

## 概念区分：协调&协作

- **协调**是指一组智能Agent完成一些集体活动时相互作用的性质。
- **协调**是对环境的适应。在这个环境中存在多个Agent并且都在执行某个动作。
- **协调**一般是改变Agent的意图，协调的原因是出于其他Agent的意图存在。
  
- **协作**是非对抗的Agent之间保持行为协调的一个特例。多Agent是以人类社会为范例进行研究的。

# 协调与协作

- 人类交互一般在纯冲突和无冲突之间。
- 在开放、动态的多Agent环境下，具有不同目标的多个Agent必须对其目标、资源的使用进行协调。
  - 如，在出现资源冲突时，需要协调。单个Agent无法独立完成目标，需要协作。





# 协调与协作

- 在多Agent系统中，协作能：
  - 提高单个Agent以及由多个Agent所形成的系统的整体行为的性能
  - 增强Agent及Agent系统解决问题的能力
  - 使系统具有更好的灵活性
- Agent间的相互依赖关系对Agent的设计和实现具有相当大的制约性
- 取决于不同的交互及协作机制，多Agent系统中的Agent的实现方式将各不相同

# 协调与协作

## • Agent协作：

- (1) 将其他领域（如博弈论、经典力学理论等）研究多Agent行为的方法和技术用于Agent协作的研究；
- (2) 从Agent的目标、意图、规划等心智态度出发来研究多Agent间的协作，如FA/C模型、联合意图框架、共享规划等。

➤ 所运用的各种理论只适用于特定的协作环境下，而一旦环境发生变化，如Agent的个数及Agent间的交互关系与该理论所适用的情形不一致时，基于该理论的协作机制就失去了其存在的优势。

较偏重于问题的规划与求解，并且它们所假定的协作过程差异显著。

- (1) 先找协作伙伴再规划求解，
- (2) 先对问题进行规划然后由Agent按照该规划采取协作性的行动，
- (3) 在Agent自主行动的过程中，进行部分全局规划来调整自己的行为以达到协作的目标。



# 协调与协作

- 在多Agent系统中，Agent是自主的，多个Agent的知识、愿望、意图和行为等往往各不相同，对多个Agent的共同工作进行协调，是多Agent系统的问题求解能力和效率得以保障的必要条件。
- 包括组织理论、政治学、社会学、社会心理学、人类学、法律学、以及经济学等在内的多个学科领域都对协调进行了研究，许多研究成果已经应用到多Agent系统中。
- 多Agent系统中的协调是指多个Agent为了以一致、和谐的方式工作而进行交互的过程。



# 协调与协作

进行协调是希望避免Agent之间的死锁和活锁。

死锁指多个  
Agent无法  
进行各自的  
下一步动作

活锁是指多个  
Agent不断工作  
却无任何进展的  
状态

- 多Agent之间的协调方法：
  - 组织结构化
  - 合同
  - 多Agent规划
  - 协商



# 协调与协作

多 Agent 之间的协作可分为（社会心理学）：

- **协作型**：同时将自己的利益放在第二位；
- **自私型**：同时将协作放在第二位；
- **完全自私型**：不考虑任何协作；
- **完全协作型**：不考虑自身利益；
- **协作与自私相混合型**。



# 协调与协作

- 在多Agent规划系统中，可以通过规定各种行为与目标的相关性而达成行为序列的自发协调。
- 德国慕尼黑技术大学的Weiβ通过分布式学习，形成对某个问题的一组行为与目标的相关值。
- 当Agent目标和偏好不相容时，可以在某个时间限制点检测出不相容，并提交给更高层的协调协议。
- 为了描述协同工作的一群Agent行为，需要用共同意图将群体成员的行为联结起来。

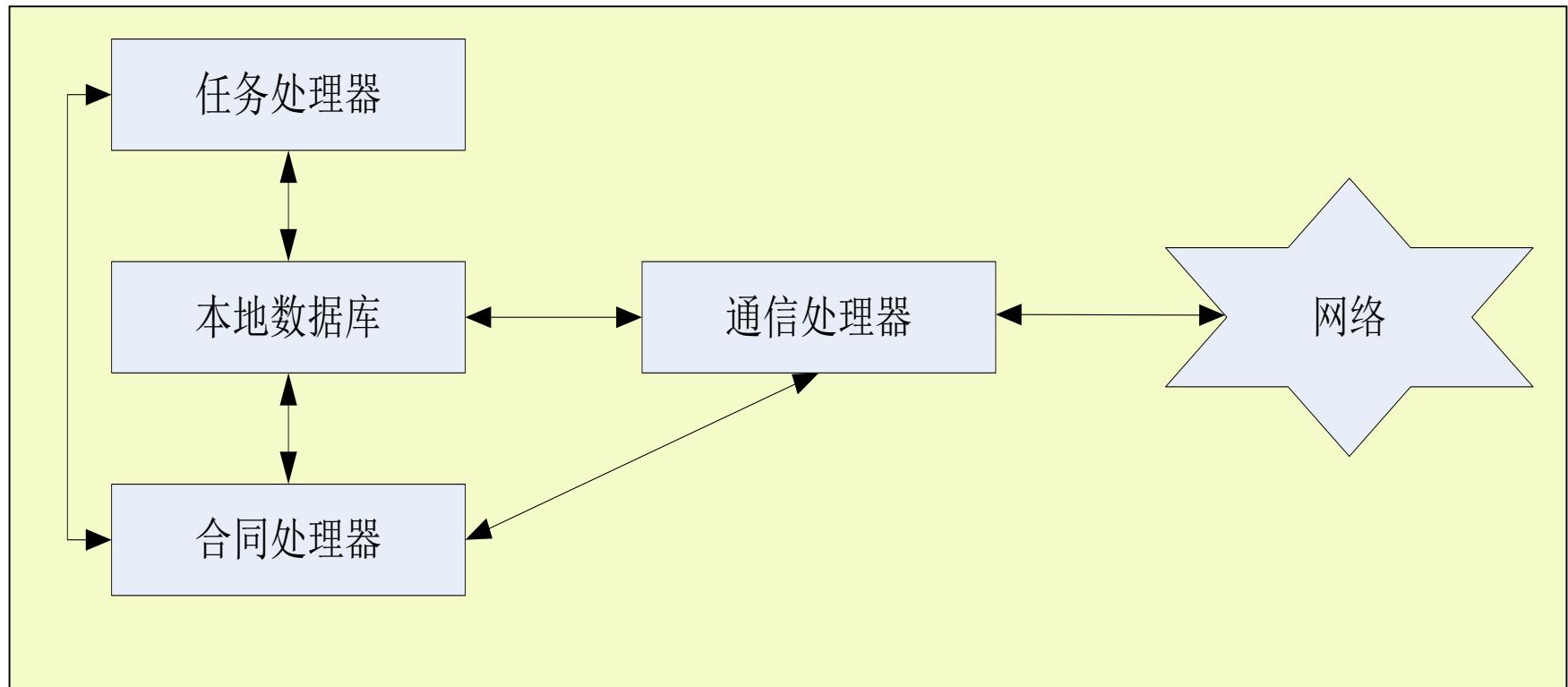


# 协调与协作

- Jennings等采用共同责任概念，强调意图作为行为控制器的作用，规定各Agent在协作问题求解中应该如何行动。
- 这种共同责任可以为系统结构设计提供功能指导，为监控问题求解提供标准，为异常处理提供准则。

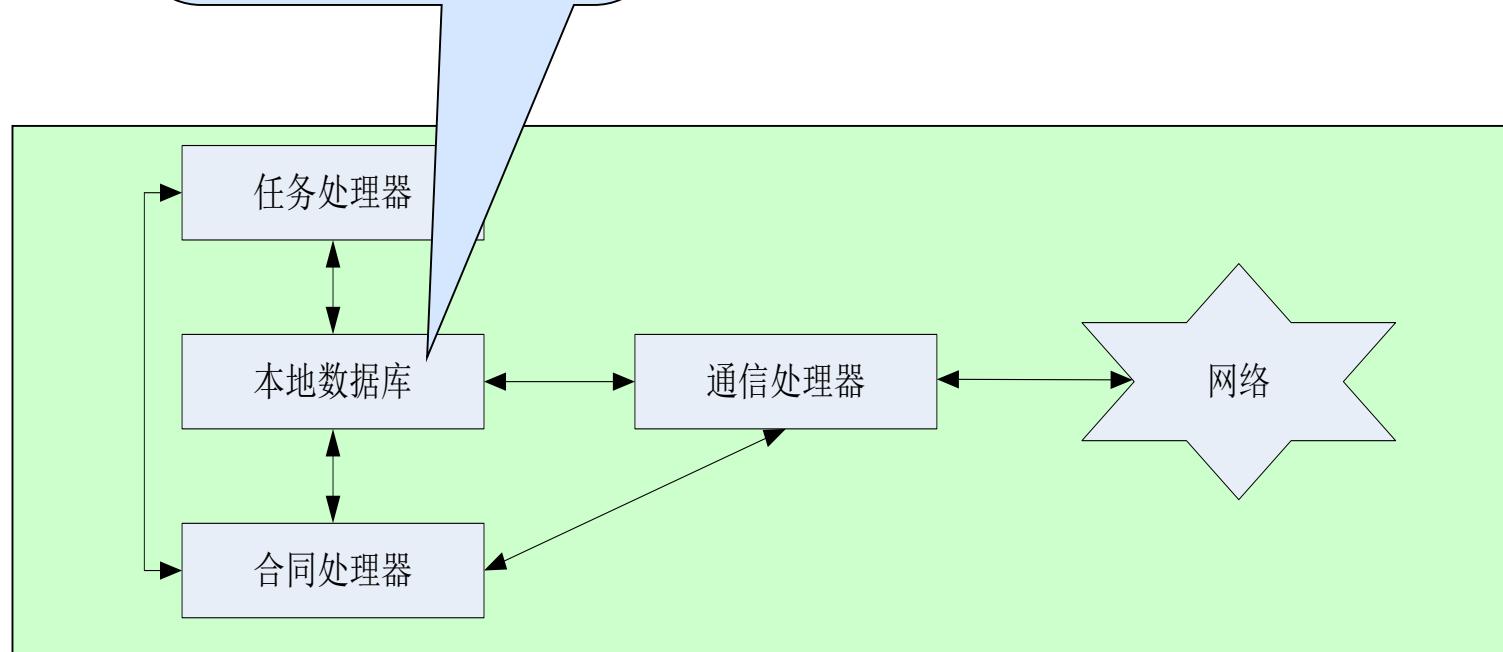
## 6.3.2 合同网

合同网系统基于合同网协议提供一种合同协议，规定任务指派和有关Agent的角色。



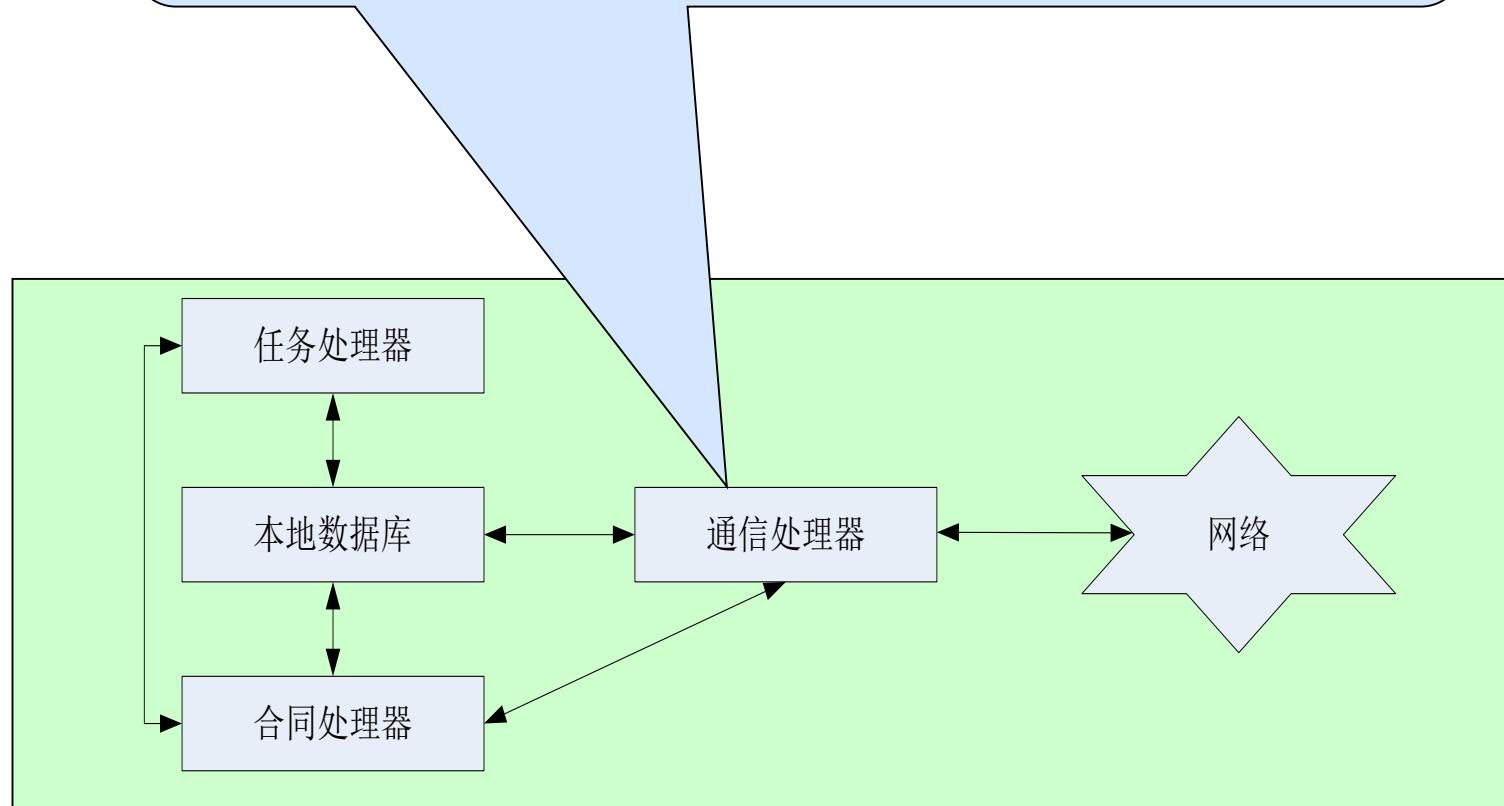
# 协调与协作

本地数据库包括与结点有关的知识库、协作协商当前状态和问题求解过程的信息。

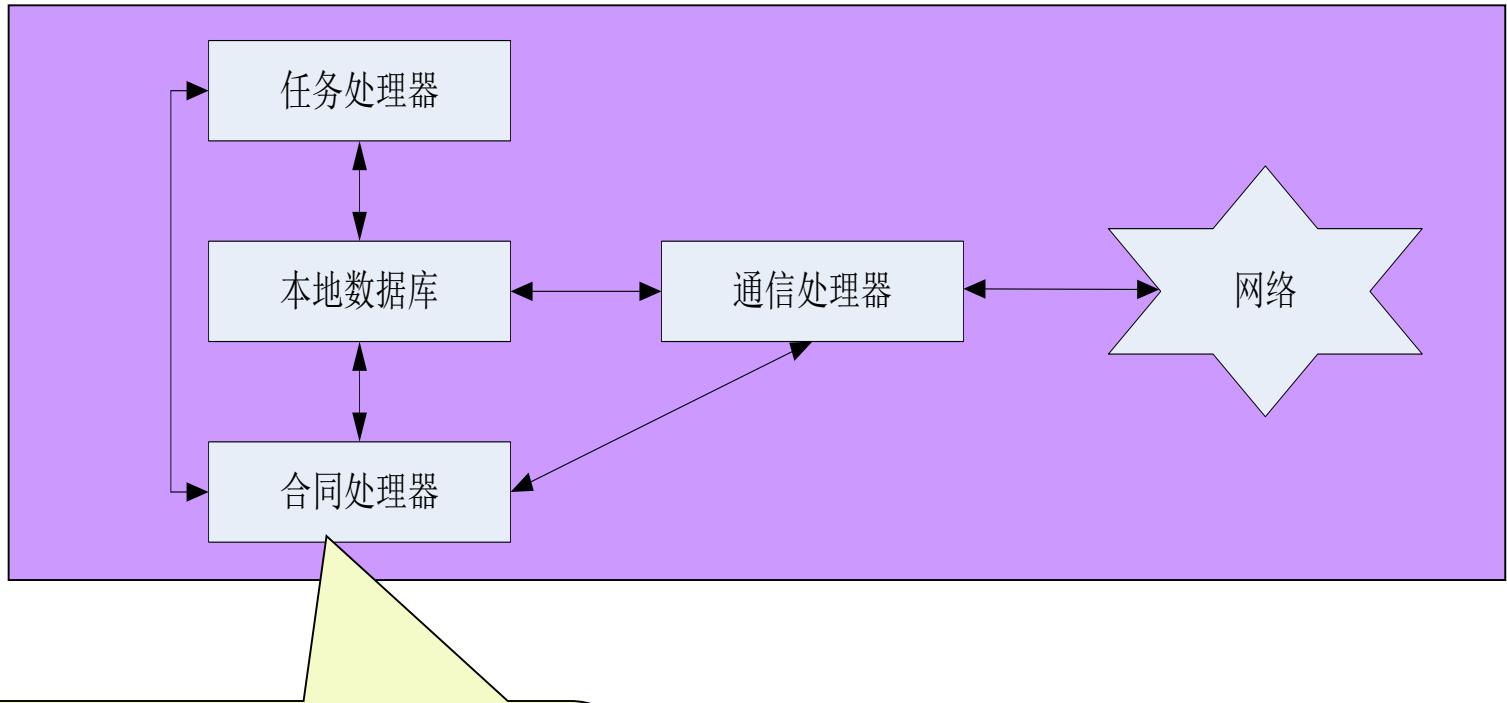


# 协调与协作

通信处理器与其他结点进行通信，结点仅仅通过该部件直接与网络相接。特别是通信处理器应该理解消息的发送和接收。



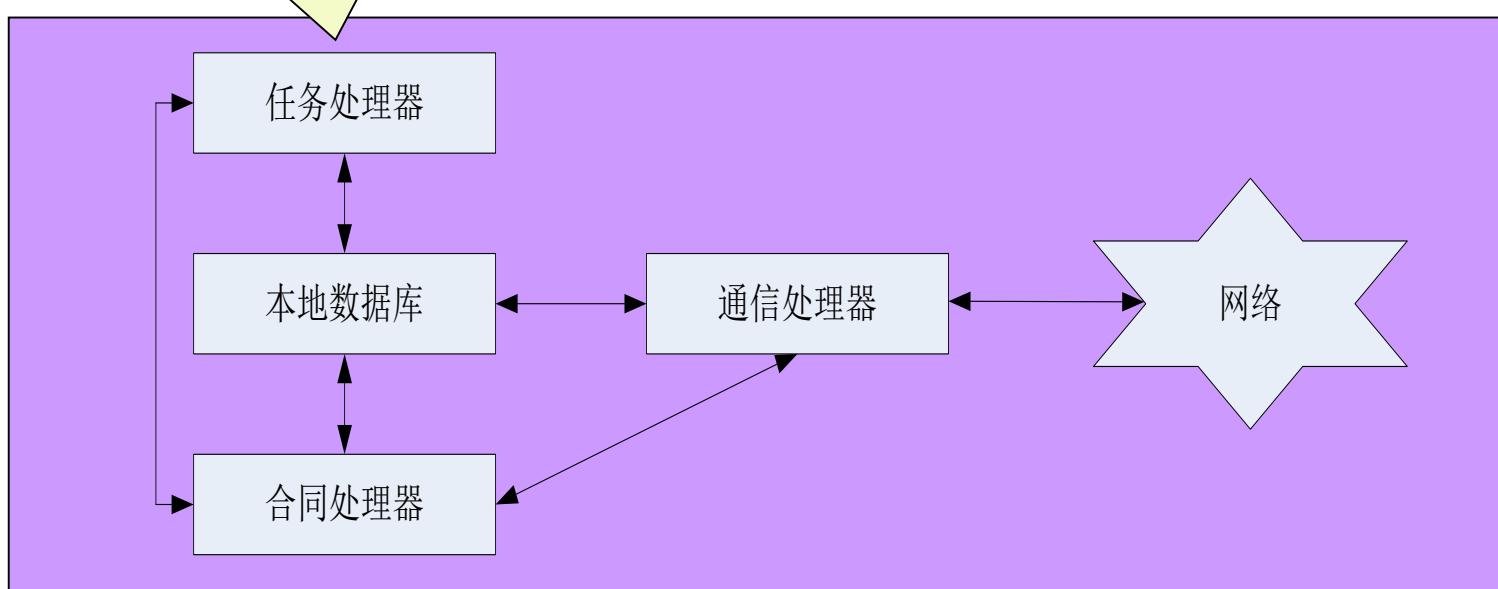
# 协调与协作



合同处理器判断投标所提供的任务，发送应用和完成合同。它也分析和解释到达的消息。合同处理器执行全部结点的协调。

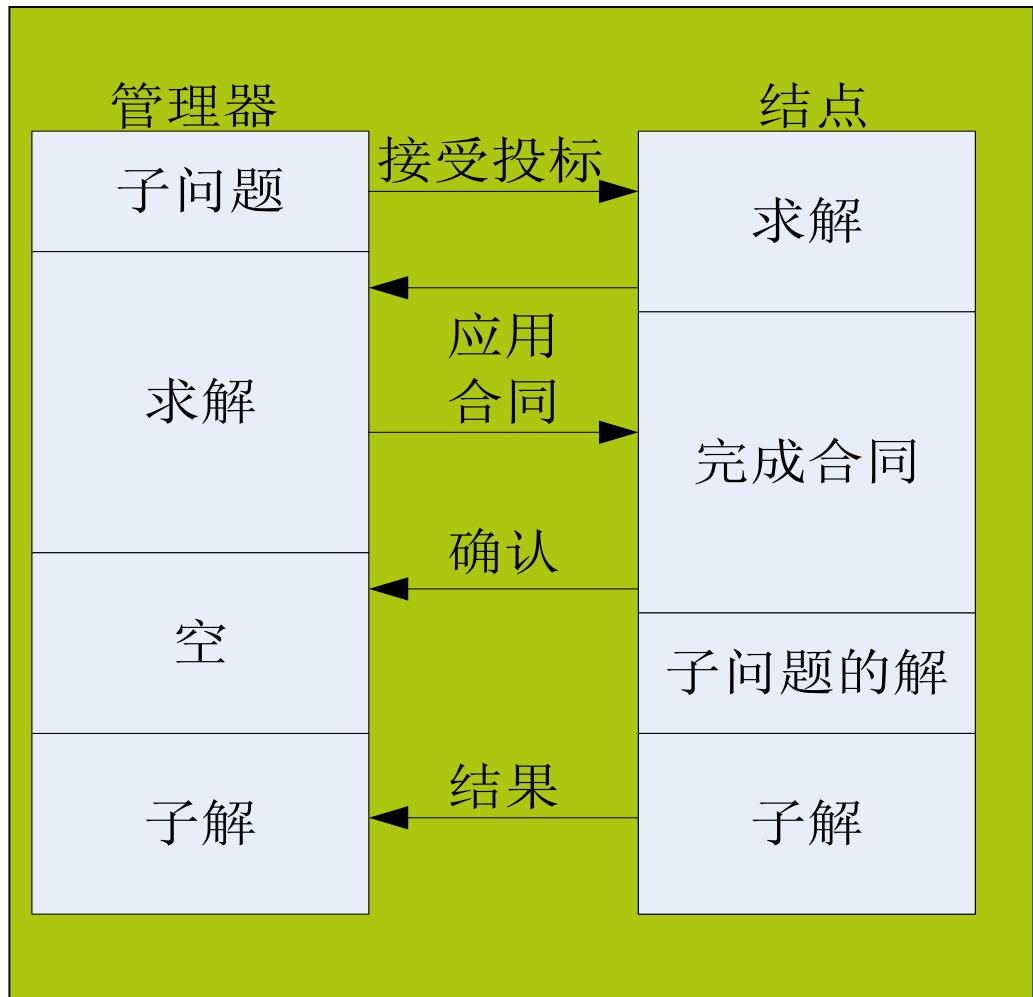
# 协调与协作

任务处理器的任务是实际处理任务赋予它的处理和求解。它从合同处理器接受所要求解的任务，利用本地数据库进行求解，并将结果送到合同处理器。



# 协调与协作

- 合同网工作时，将任务分成一系列子问题。有一个特定的结点称作管理器，它了解子问题的任务。





# 协调与协作

- 管理器提供投标，即要解而尚未求解的子问题合同。它使用合同协议定义的消息结构，例如：

TO :	All nodes
FROM :	Manager
TYPE :	Task bid announcement
ContractID :	xx-yy-zz
Task Abstraction :	<description of the problem>
Eligibility Specification :	<list of the minimum requirements>
Bid Specification :	< description of the requires application information> < latest possible application time>
Expiration time	

# 协调与协作

➤ 标书对所有Agent都是开放的，通过合同处理器进行求解。使用本地数据库求解当前可用的资源和Agent知识。合同处理器决定该公布的任务申请是不是要做。如果要做，它将按下面结构通知管理器：

TO :	Manager
FROM :	Node X
TYPE :	Application
ContractID :	xx-yy-zz
Node Abstraction :	<description of the node's capabilities>



# 协调与协作

- 管理器必须选择结点，使之在应用中最适合于所给的合同。它访问具体的求解知识和方法，选择最好成绩，将合同有关的子问题求解任务交给它。根据合同消息管理器指派合同如下：

TO :	Node X
FROM :	Manager
TYPE :	Contract
ContractID :	xx-yy-zz
Task Specification	<description of the subproblem>



# 协调与协作

- 通信结点发送确认消息到管理器，以规定的形式确认接受合同。
  - 当问题求解阶段完成，已解的问题传给管理器。
  - 承诺的结点完全负责子问题的求解，即完成合同。
- 
- 如果结点认为所安排的任务超过它的能力和资源，那么可以进一步划分子问题，分配子合同到其他结点。这时，它用作管理器角色，提交子问题标书。
  - 形成分层任务结构，每个结点可以同时是管理器、投标申请者和合同成员。

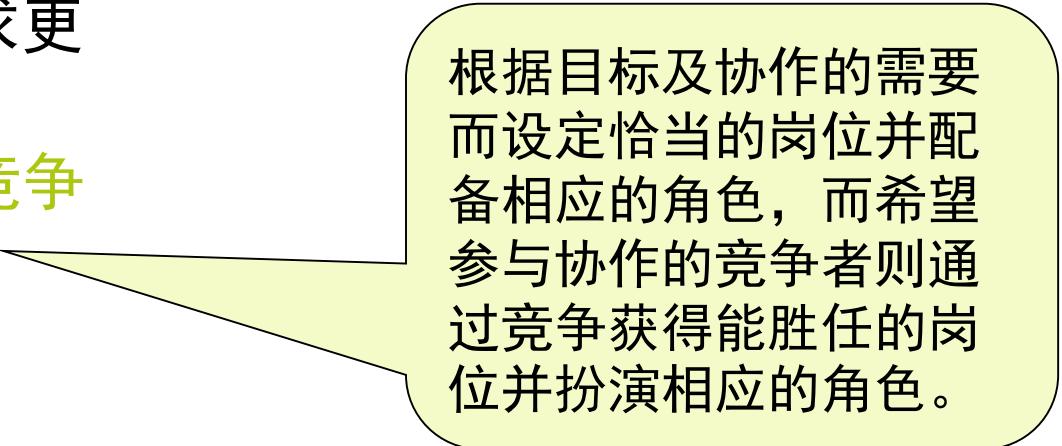


## 6.3.3 协作规划

- 当某个Agent相信通过协作能带来好处时，会产生协作的愿望，进而寻求协作伙伴；
- 或者当多个Agent在交流过程中，发现它们能够通过协作来实现更大的目标时，可能会结成同盟，并采取协作性的行动。

# 协调与协作

- 如，一个企业主为实现某种企业目标而招聘合适的工作人员，多个具有共同利益的企业组成一个大的集团以追求更大的收益等。
- 原则 “**因需设岗，竞争上岗**”



根据目标及协作的需要而设定恰当的岗位并配备相应的角色，而希望参与协作的竞争者则通过竞争获得能胜任的岗位并扮演相应的角色。



# 协调与协作

- 将规划、竞争、约束、协调纳入一个协作框架之中对多Agent间的协作过程进行研究，并将协作过程分为：
  - (1) 产生需求、确定目标
  - (2) 协作规划、求解协作结构
  - (3) 寻求协作伙伴
  - (4) 选择协作方案
  - (5) 实现目标
  - (6) 评估结果



## 1、形式化框架

- 这里所采用的逻辑框架主要是基于时态逻辑的，并在时态逻辑的基础上添加了若干与Agent行为有关的模态算子。

- 时态逻辑语言TL
- 在时态逻辑中存在两类时序连接符：
  - ❖ 过去时的连接符
  - ❖ 现在及将来时的连接符。

# 协调与协作

## 2、协作模型和结构

$$M = \langle Ag, G, P, T, S \rangle$$

**协调Agent**。协作协调者提出协作需求，并对协作目标进行规划；还负责根据竞争者的条件挑选恰当的协作伙伴。

**协作目标**，由协调Agent在特定的情况下产生。  
**任务目标，性能目标**

**协作规划**，构造出问题的协作结构，即完成“因需设岗，竞争上岗”中的第一步，确定完成协作任务所需的角色、角色的相关性质，以及各角色之间的依赖关系。

# 协调与协作

$$M = \langle Ag, G, P, T, S \rangle$$

协作伙伴集合，即参与协作的协作团体， $Ag \in T$ 。协作团体的形成是协调Agent根据各Agent竞争相应的协作角色的有关信息而确定的。

协作方案，它与协作伙伴相对应， $S \in P$ 。



## 3、协作方案

Agent竞争及确定协作伙伴的过程：

- (1) 协作协调者根据协作结构向外界公布可竞争的岗位和角色；
- (2) 各竞争Agent根据自己的能力竞争相应的岗位和角色，并提交相关信息，如有能力胜任何种工作、完成任务可能需要的时间以及实现目标所需的资源等。
- (3) 根据竞争Agent所提交的信息，挑选最合适扮演某一角色的Agent，挑选标准可根据Agent的竞争信息及各要素的权重计算得到。



# 协调与协作

- 协作伙伴在扮演所竞争到的角色即开始进行协作前，
  - (1) 相互信赖，即协作团体中的每个Agent都相信自己和其他Agent都能完成各自所承诺的任务，这样协作才能进行下去。
  - (2) 参与协作的Agent一旦作出承诺，就必须采取行动来实现目标。
- 这样，如果协作伙伴相互信任，且会对自己的目标作出承诺，并保证目标的实现，则多Agent间的协作可以顺利进行。



## 4、协作过程分析

- 以实现协作目标为出发点的多Agent间的协作行为必须满足特定的性质或标准，这些性质包括：
  - (1) Agent间应该相互响应，
  - (2) 所有Agent应对联合行动作出承诺，
  - (3) 每个Agent应承诺对相互之间的行动给予支持；以及协作过程中不发生死锁、能满足特定的环境约束等。



# 协调与协作

在协作过程中，Agent性质表现如下：

- (1) Agent之间相互信赖。
- (2) Agent对协作行动予以承诺。
- (3) 无死锁的协作过程。
- (4) 协作与约束。不仅定义了多Agent应完成的任务，还定义了多Agent在协作过程中应满足的环境限制或约束。
- (5) 协作与协调。定义了一种目标关系图，用来限定Agent行为之间的相互关系。同时，将目标关系进程与目标结构进程组合在一起，从而保证了多Agent行为的协调性。



# 第6章 Agent

6. 1 概述

6. 2 Agent及其结构

6. 3 协调与协作

6. 4 Agent研究方向



## 6.4 Agent研究方向

- 6.4.1 知识表示与推理/规划
- 6.4.2 拍卖理论与非合作博弈
- 6.4.3 资源分配与合作博弈
- 6.4.4 人-Agent交互
- 6.4.5 大模型Agent



## 6.4.1 知识表示与推理/规划

**核心假设：**人工智能可以通过符号推理形式化刻画。

The basic assumption underlying KR (and much of AI) is that **thinking can be usefully understood as mechanical operations over symbolic representations**. This hypothesis is, in fact, quite old, much older than computers, and seems to have originated with the philosopher Leibniz ... Just as there is a calculus of arithmetic, where numerical expressions are formally manipulated in a value-preserving way, so might there be a calculus of thought, where propositional expressions could be formally manipulated in a truth-preserving way.

--Levesque, 1986



# Agent研究分类

**研究目的：**将知识表达为计算机可处理 (computer-tractable) 的形式，以供智能体使用。

知识表示的语言包括：

- ▣ 语法 (syntax) : 语言中合法的语句。例如，一阶逻辑中公式。
- ▣ 语义 (semantics) : 语句在世界中的解释。例如， *het(food)* 表示 *food* 处在被加热后的状态。



# Agent研究分类

智能体在 KR 中用语句 (sentences) 表达世界/环境

- ▣ 语句 (Sentences) 由某种形式化语言来描述
- ▣ 事实 (Facts) 表示世界/环境中真/假的断言

## Sentences

representation in agent

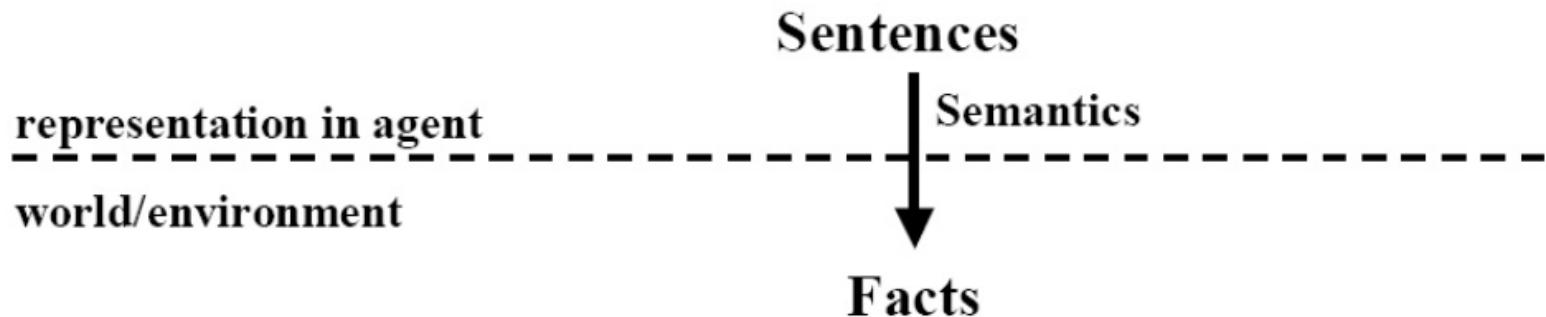
world/environment

## Facts

# Agent研究分类

## 语句表达世界中的事实

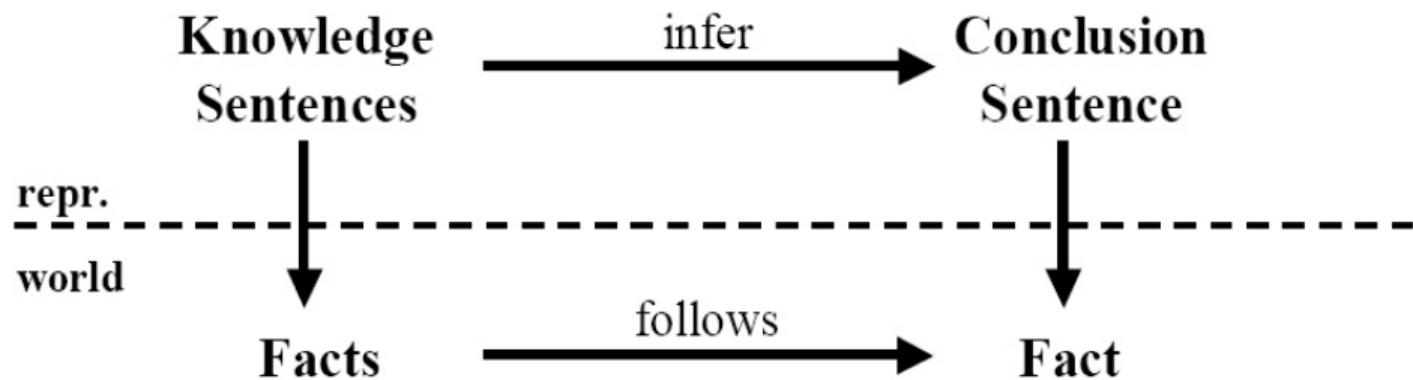
- ▶ 语义 (semantics) 将语句与事实连接起来
- ▶ 某语句为真，则它所代表的事实在相应（真实）世界中也为真



# Agent研究分类

通过合理的推理机，使得由 KR 推理得到的结论也符合事实

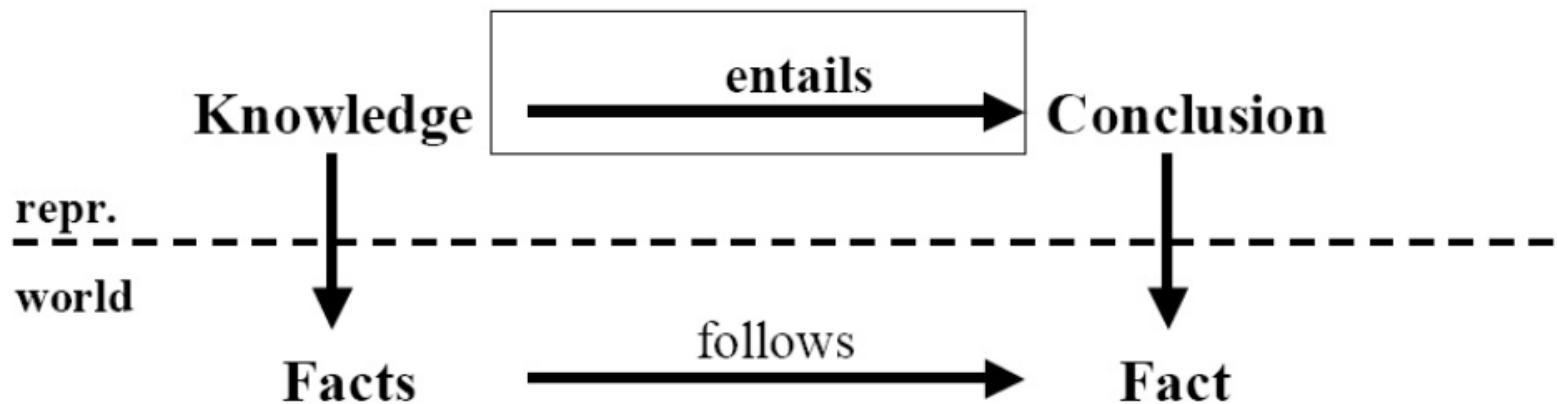
- ▣ 逻辑的可靠性 (soundness) : 推理机推出的语句都符合事实
- ▣ 逻辑的完全性 (completeness) : 凡是为真的事实，都可以由推理机推出



# Agent研究分类

## 计算机不知道语义 (semantics) /含义

- ▶ 只能在完全不知道语义的情况下机械的处理语句
- ▶ 知识表示的根本问题：在设计逻辑语言时，平衡其表达能力 (expressiveness) 与推理效率 (tractability)





# Agent研究分类

## 命题逻辑中推理方法

基于推理规则的方法：从老公式推出新公式，构造或找到一个证明过程

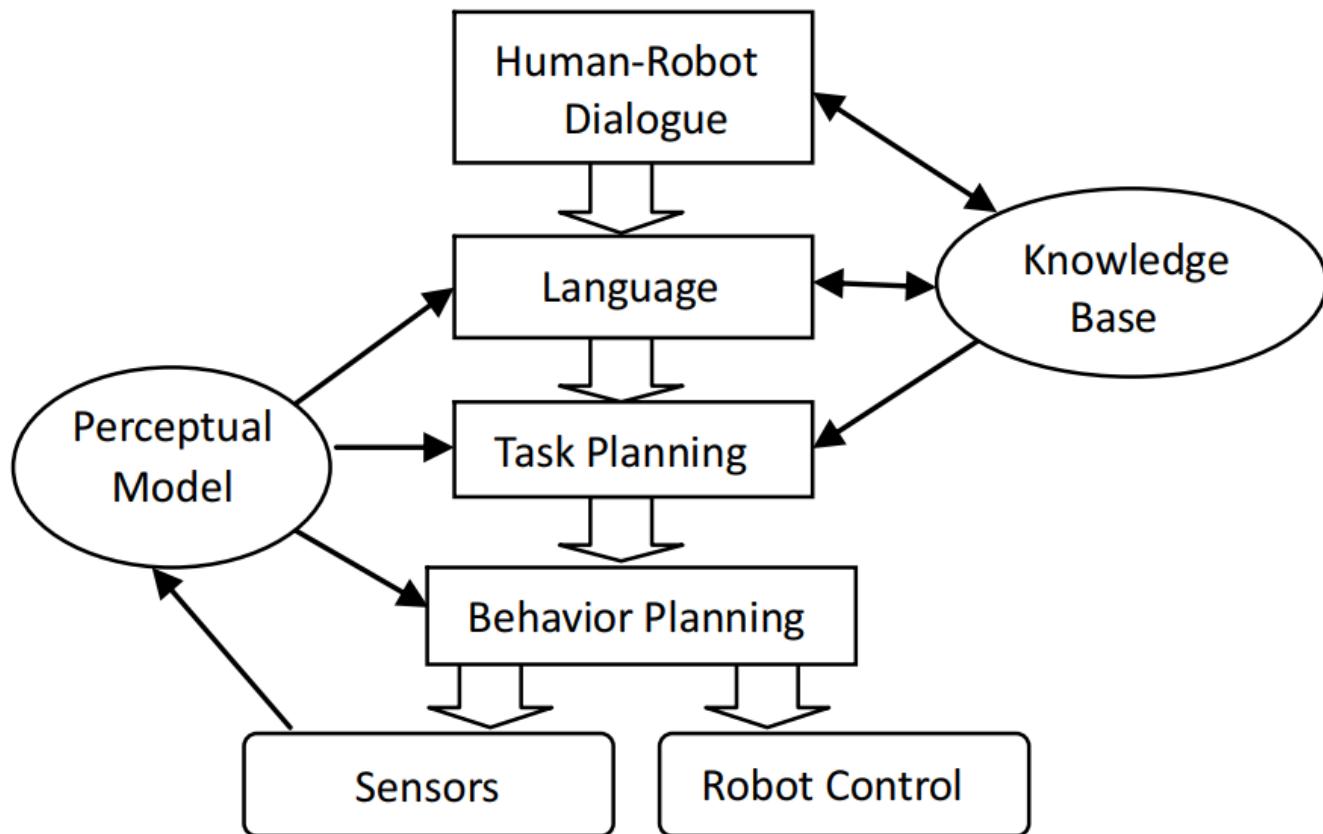
- ▣ 归结 (resolution)：将公式化为子句形式，不断使用归结推理规则，直到没有可以添加的新语句或出现空子句。
- ▣ 前向/后向链接 (forward/backward chaining)：线性时间，针对 Horn 子句（至多只有一个正文字的析取式）是完备的。

模型检测的方法：构造或找到一个可满足的模型

- ▣ 穷举法（真值表法）：枚举所有可能的解释，逐个判断。时间复杂度为  $O(2^n)$ ，空间复杂度为  $O(n)$ 。
- ▣ 回溯搜索方法：DPLL
- ▣ 启发式搜索方法（可靠但不完备）

# Agent研究分类

## 逻辑智能体：可佳机器人





# Agent研究分类

任务规划的主要**目标**: 根据用户服务要求, 结合机器人现有知识, 计算出完成任务的一个行动序列。

► 用行动语言表达刻画行动的直接或间接效果的因果知识:

**caused** *occurs(catch(A)) if occurs(catch(A)),*

**caused**  $\perp$  **if**  $\top$  **after** *occurs(catch(A)) \wedge (\neg holds(handempty) \vee \neg holds(sameLocation(agent, A)))*,

**caused** *holds(holding(A)) if size(A, small)* **after** *occurs(catch(A)),*

**caused**  $\neg holds(handempty)$  **if** *holds(holding(A)),*

**caused** *holds(holding(A)) if holds(holding(A))*  
**after** *holds(holding(A)),*

**caused** *holds(handempty) if holds(handempty)*  
**after** *holds(handempty),*



# Agent研究分类

- ▶ 智能体需要关于世界的知识以便达到良好的决策。
- ▶ 知识以**知识表示语言** (KR language) 的**语句** (sentences) 的形式存储在**知识库** (KB) 中。
- ▶ 逻辑智能体 (基于知识的智能体) 由**知识库**和**推理机构**构成。通过对**知识库**中语句进行**推理**，以得到新的语句，并由此进行决策。
- ▶ 基本概念：
  - ▶ 语法 (Syntax) : 合法语句的形式化结构
  - ▶ 语义 (Semantics) : 语句的真值解释
  - ▶ 语义后承 (Entailment) : 在语义意义下的成真结论
  - ▶ 推理 (Inference) : 基于语句的形式化推理
  - ▶ 可靠性 (Soundness) : 推理结论都是语义后承
  - ▶ 完成性 (Completeness) : 语义后承都能被推理出来



## 6.4 Agent研究分类

6.4.1 知识表示与推理/规划

6.4.2 拍卖理论与非合作博弈

6.4.3 资源分配与合作博弈

6.4.4 人-Agent交互

6.4.5 大模型Agent

## 6. 5. 2 拍卖与非合作博弈论

- ▶ 拍卖是指用于在自利代理之间分配资源的任何机制
- ▶ 资源分配是计算机科学中的一个基本问题
- ▶ 研究具有异构Agent的分布式系统变得越来越重要
- ▶ 货币不一定是真实货币，可以是一种稀缺物品





# Agent研究分类

## 拍卖方式的四种类型

(1) **英式拍卖(English Auction)**。又叫升价拍卖，竞价者在一起公开竞标，往上抬价，出价最高者获得拍品。这是我们最常见到的拍卖。

(2) **荷式拍卖(Dutch Auction)**。又叫降价拍卖，价格由高往低降，第一个接受价格的人获得拍品。

(3) **第一价格拍卖(Fist-price Auction)**。每个竞买人对拍品进行单独密封报价，但相互不知道其他竞买人的出价，标的装在信封里交上去，然后拍卖人拆开信封，拍卖人按各个标价的大小排序，最后在规定的时间、地点宣布标价，出价最高的竞买人获胜。

(4) **第二价格拍卖(Second-price Auction)**。二级价格拍卖与一级价格拍卖类似，不同的是最后出价最高的竞买人获得拍品，但只需要按照排位第二高的价格进行支付，即第二价格，因此这种拍卖方式被称为二级密封价格拍卖或“维克里拍卖”。



# Agent研究分类

## 拍卖方式的四种类型

	English	Dutch	1 <sup>st</sup> -Price	2 <sup>nd</sup> -Price
<b>Duration</b>	#bidders, increment	starting price, clock speed	fixed	fixed
<b>Info</b>	2 <sup>nd</sup> -highest	winner's	none	none
<b>Revealed</b>	val; bounds on others	bid		
<b>Jump bids</b>	yes	n/a	n/a	n/a
<b>Price Discovery</b>	yes	no	no	no
<b>Regret</b>	no	yes	yes	no

这四种拍卖形式也奠定了拍卖的基本类型，其他无论什么形式的拍卖都只是这四种形式的变型与组合。



## 使用博弈论对拍卖进行理论分析

### 设定：

- ▶ 一个卖方出售一件艺术品，共有 $n$ 个买方，记为 $1, 2, \dots, n$ 。
- ▶ 每个买方只确切地知道该艺术品对自己的价值，而不知道该艺术品对其他人的价值，且知道

$$0 \leq \underline{v} \leq v_i \leq \bar{v}, \quad i = 1, 2, \dots, n$$

- ▶ 假定这些私人价值  $v_1, \dots, v_n$  相互独立，而且具有相同的概率分布，以  $F(v)$  表示其累积分布函数，该函数连续递增。
- ▶ 每个买方分别提交自己的出价，卖方得到  $n$  个出价： $b_1, \dots, b_n$



# Agent研究分类

以  $\beta_i(v_i)$  表示第  $i$  个买方的出价函数。我们考虑一个对称的均衡：

$$\beta_i(v_i) = \beta(v_i), \quad i = 1, 2, \dots, n$$

可以合理地猜测，在均衡状态下出价函数  $\beta(v)$  为单调递增函数。

私人价值为  $v$  的买方出价  $b$  时，期望收益为

$$\begin{aligned} (v - b) \cdot \Pr\{b \text{最大}\} &= (v - b) \cdot \Pr\{\text{其他人的出价都小于 } b\} \\ &= (v - b) \cdot \Pr\{\text{其他人的私人价值都小于 } \beta^{-1}(b)\} \\ &= (v - b) \cdot \left\{ F[\beta^{-1}(b)] \right\}^{n-1} \end{aligned}$$



# Agent研究分类

该买方选择  $b$  使期望收益最大化，得到一阶条件：

$$-\left\{F\left[\beta^{-1}(b)\right]\right\}^{n-1} + \frac{(v-b)(n-1) \cdot \left\{F\left[\beta^{-1}(b)\right]\right\}^{n-2} \cdot F'\left[\beta^{-1}(b)\right]}{\beta'\left[\beta^{-1}(b)\right]} = 0$$

这样， $(\beta, \dots, \beta)$  要构成纳什均衡，对于任意的  $v$ ， $b = \beta(v)$  就必须满足上面的一阶

条件。如果  $b = \beta(v)$ ，则  $v = \beta^{-1}(b)$ ，一阶条件化为

$$-\left\{F(v)\right\}^{n-1} + \frac{[v - \beta(v)] \cdot (n-1) \cdot \left\{F(v)\right\}^{n-2} \cdot F'(v)}{\beta'(v)} = 0$$



# Agent研究分类

即

$$\beta'(v) \cdot [F(v)]^{n-1} + (n-1) \cdot \beta(v) \cdot [F(v)]^{n-2} \cdot F'(v) = (n-1) \cdot v \cdot [F(v)]^{n-2} \cdot F'(v)$$

注意到方程左侧正好等于  $\beta(v) \cdot [F(v)]^{n-1}$  对于  $v$  的导数，我们将方程两边积分，得

$$\begin{aligned}\beta(v) \cdot [F(v)]^{n-1} &= \int_v^v (n-1) \cdot x \cdot [F(x)]^{n-2} \cdot F'(x) dx \\ &= v \cdot [F(v)]^{n-1} - \int_v^v [F(x)]^{n-1} dx \quad (\text{注: 分部积分})\end{aligned}$$

所以，在对称均衡中，出价函数为

$$\beta^*(v) = v - \frac{\int_v^v [F(x)]^{n-1} dx}{[F(v)]^{n-1}}$$



# Agent研究分类

特例：如果私人价值 $v$ 服从 $[0,1]$ 区间上的均匀分布，则 $\underline{v}=0$ ,  $\bar{v}=1$ ,  $F(v)=v$ , 对称均衡中的出价函数就为

$$\beta^*(v) = v - \frac{\int_0^v x^{n-1} dx}{v^{n-1}} = v - \frac{\frac{1}{n} x^n \Big|_0^v}{v^{n-1}} = \left(1 - \frac{1}{n}\right)v$$



# Agent研究分类

## 收益等价定理

给定买方人数，假定所有买方都是风险中性的，各买方的价值独立且具有相同的连续分布，而且分布函数严格递增。

那么，任何满足下列两个条件的拍卖机制对于卖方都会产生相同的期望收益（对于买方则产生相同的期望支出）：

- ▶ 条件1：标的物总是为最高信号的买方获得；
- ▶ 条件2：如果买方的信号位于信号区间的下限，那么他的期望剩余为零。



# Agent研究分类

## 收益等价定理（简化版）

在一些合理的约束条件保证下，四种拍卖形式最终的收益是相等的。

### Remarks:

- ▶ “收益等价定理”是整个拍卖理论研究的起点。
- ▶ 其表明：对于委托人来说，只要拍卖品不变，购买对象不变，无论采用什么拍卖方式，最终收益都是一样的。
- ▶ 当然“收益等价定理”说的是理想状态，这里我们千万不要忘记约束条件。在现实中，虽然拍卖品是共同的，但由于参与竞标的人不同、不同的拍卖流程和规则，结果产生拍卖结果的巨大差异。甚至同样一种拍卖办法，放到不同的国家，结果也会有天壤之别，原因在于现实的约束条件不同。



# Agent研究分类

## 6.4 Agent研究分类

- 6. 4. 1 知识表示与推理/规划
- 6. 4. 2 拍卖理论与非合作博弈
- 6. 4. 3 资源分配与合作博弈
- 6. 4. 4 人-Agent交互
- 6. 4. 5 大模型Agent



## 6. 4. 3 社会选择与合作博弈

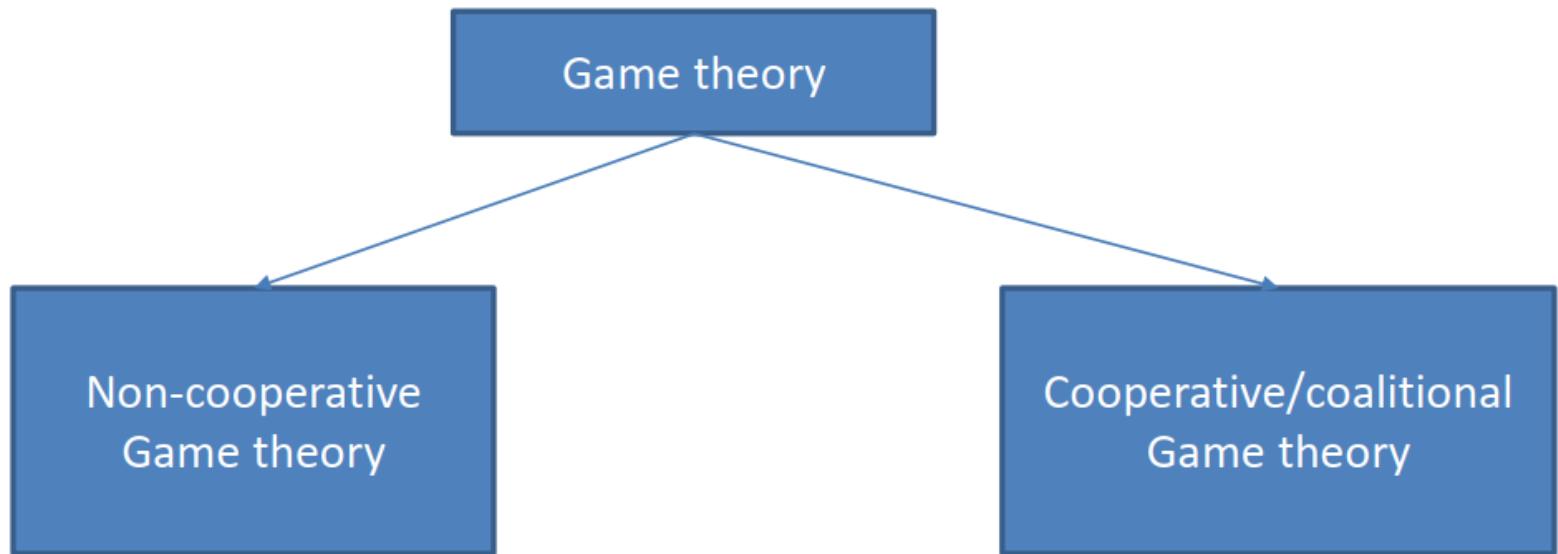
合作博弈是一种侧重于**玩家群体行为**的模型。与非合作博弈不同，合作博弈中的战略互动没有被明确建模。

一类典型的合作博弈问题是**资源分配**问题

例子：改革开放要求全国人民合作，并允许有部分利益牺牲，但改革开放果实应该让所有人共享，如何共享？  
不患寡而患不均？算术平均？考虑贡献的加权平均？

# Agent研究分类

## 合作博弈 VS. 非合作博弈



- actions are taken by **individual** agents
- No binding agreements

- actions are taken by **groups** of agents
- **binding agreements** are possible

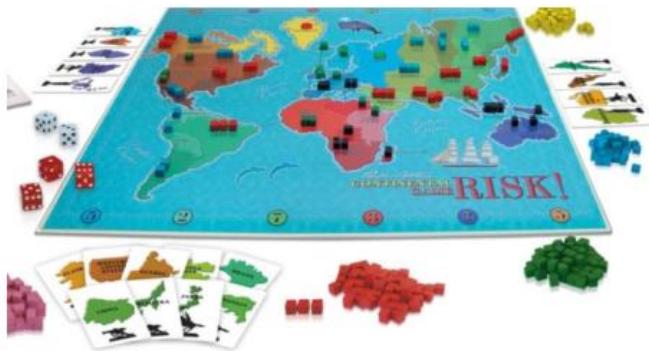
# Agent研究分类

## 合作博弈 VS. 非合作博弈

7 players

**Actions:** attack, move, ship...

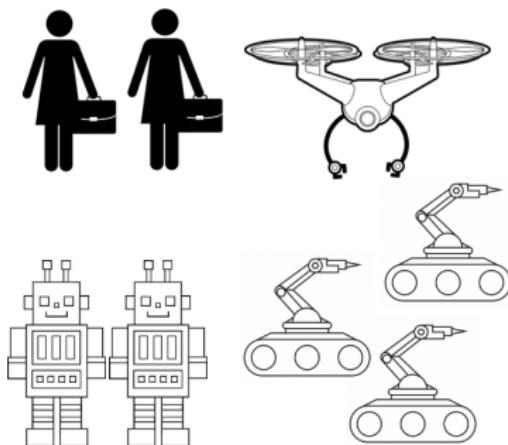
**Utilities:** Acquired land.  
Affected by the joint actions of everyone.



**Players:** rescue workers and robots of different types

**Actions:** drill, seek, dig, call, pull, move, ...

**Utilities:** ?



7 players

**“Actions”:** form coalitions

**Value:** Maximum land the coalition can guarantee

**Players:** rescue workers and robots of different types

**“Actions”:** form coalitions

**Value:** ?



# Agent研究分类

## 合作博弈的基本过程

- ▶ Agent们组成多个联盟 (coalitions)
- ▶ 每个联盟隐性地选择动作
- ▶ 可转让效用 (TU) 博弈：联盟行动的选择（由所有联盟决定）决定了每个联盟的回报  
——联盟中的Agents随后需要决定如何分配这个总回报

# Agent研究分类

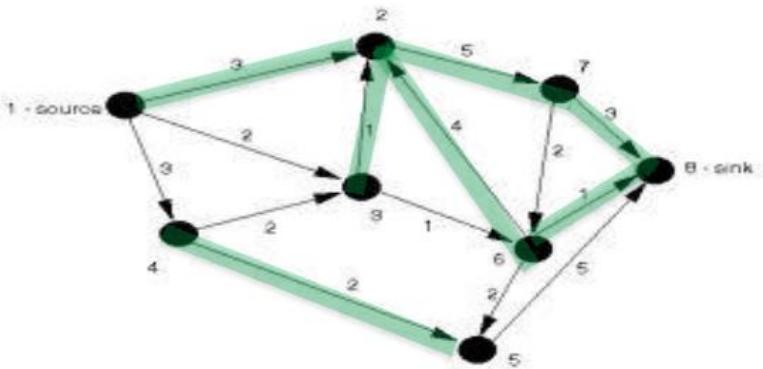
## 合作博弈基本过程的例子：买冰淇淋

- $n$  children, each has some amount of money
  - the  $i$ -th child has  $b_i$  dollars
- three types of ice-cream tubs are for sale:
  - Type 1 costs \$7, contains 500g
  - Type 2 costs \$9, contains 750g
  - Type 3 costs \$11, contains 1kg
- children have utility for ice-cream, and do not care about money
- The payoff of each group: the maximum quantity of ice-cream the members of the group can buy by pooling their money
- The ice-cream can be shared arbitrarily within the group



## 合作博弈基本过程的例子：Flow Games

- Each agent controls an edge (or several edges) in a weighted flow graph



- Quantity version: The value of a coalition is the amount it can flow from source to target
- Threshold version: The value is 1 if the coalition can flow more than  $q$ , and 0 otherwise.



# Agent研究分类

## 合作博弈的基本挑战

### ► 表示

——如何表示全部 $2^n$ 个联盟的值？

### ► 联盟形成

——有可能形成怎样的联盟？

### ► 回报分配

——联盟中的Agents如何分配得到的总回报？(Stable allocations, Fair allocations, Shapley value)



# Agent研究分类

## 数学定义

A transferable utility game is a pair  $(N, v)$ , where:

- $N = \{1, \dots, n\}$  is the set of players
- $v: 2^N \rightarrow \mathbb{R}$  is the characteristic function
  - for each subset of players  $C$ ,  $v(C)$  is the amount that the members of  $C$  can earn by working together
- usually it is assumed that  $v$  is
  - normalized:  $v(\emptyset) = 0$
  - non-negative:  $v(C) \geq 0$  for any  $C \subseteq N$
  - monotone:  $v(C) \leq v(D)$  for any  $C, D$  such that  $C \subseteq D$

A coalition is any subset of  $N$ ;  
 $N$  itself is called the grand coalition

# Agent研究分类

还是前面买冰淇淋的例子，用形式化语言表述如下：

## Ice-Cream Game: Characteristic Function



C: \$6,



M: \$4,



P: \$3



w = 500

p = \$7



w = 750

p = \$9



w = 1000

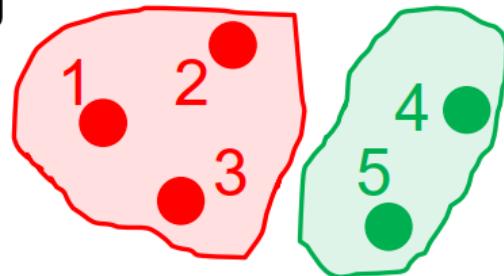
p = \$11

- $v(\emptyset) = v(\{C\}) = v(\{M\}) = v(\{P\}) = 0$
- $v(\{C, M\}) = 750, v(\{C, P\}) = 750, v(\{M, P\}) = 500$
- $v(\{C, M, P\}) = 1000$

# Agent研究分类

## 合作博弈的Outcome——定义：

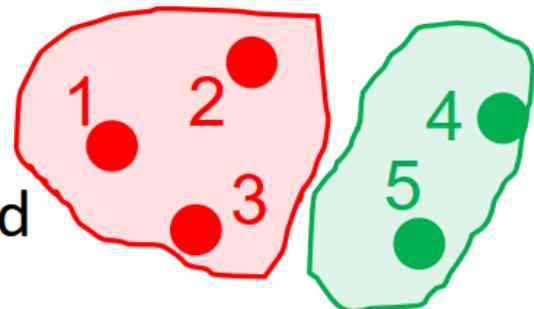
- An outcome of a TU game  $G = (N, v)$  is a pair  $(CS, \underline{x})$ , where:
  - $CS = (C_1, \dots, C_k)$  is a coalition structure, i.e., partition of  $N$  into coalitions:
    - $\cup_i C_i = N$ ,  $C_i \cap C_j = \emptyset$  for  $i \neq j$
  - $\underline{x} = (x_1, \dots, x_n)$  is a payoff vector, which distributes the value of each coalition in  $CS$ :
    - $x_i \geq 0$  for all  $i \in N$
    - $\sum_{i \in C} x_i = v(C)$  for each  $C$  is  $CS$



# Agent研究分类

## 合作博弈的Outcome——例子：

- Example:
  - suppose  $v(\{1, 2, 3\}) = 9$ ,  $v(\{4, 5\}) = 4$
  - then  $((\{1, 2, 3\}, \{4, 5\}), (3, 3, 3, 3, 1))$  is an outcome
  - $((\{1, 2, 3\}, \{4, 5\}), (2, 3, 2, 3, 3))$   
is NOT an outcome: transfers  
between coalitions are not allowed
- An outcome  $(CS, \underline{x})$  is called an imputation if it satisfies individual rationality:  
 $x_i \geq v(\{i\})$  for all  $i \in N$
- Notation: we will denote  $\sum_{i \in C} x_i$  by  $x(C)$





# Agent研究分类

## 6.4 Agent研究分类

6.4.1 知识表示与推理/规划

6.4.2 拍卖理论与非合作博弈

6.4.3 资源分配与合作博弈

6.4.4 人-Agent交互

6.4.5 大模型Agent

# Agent研究分类

## 6. 4. 4 人-Agent交互



图：虚拟/实体Agent与人类交互的典型场景



# Agent研究分类

## 什么是人-Agent交互?

- ▶ 仅仅拥有自主性（Autonomy）的Agent并不足以作为人-Agent交互的研究对象

例如，人们通常不会将电话或电梯视为Agent，尽管其具有自主性

- ▶ 当人们塑造他们与物体或技术的互动，好像它们有目的、动机或意图时，我们有了人-Agent交互

例如，一些人会与他们的汽车或船交谈，好像它们有自己的意图或个性，即使事实并非如此

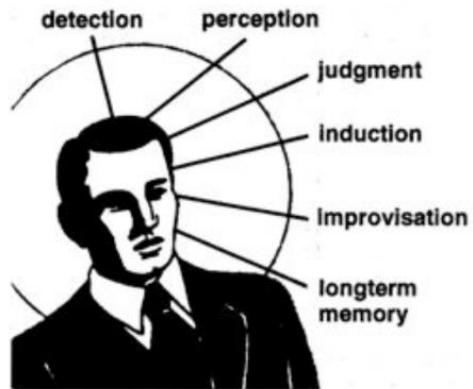
- ▶ 通常，技术被设计成鼓励人们赋予它们Agent的性质，尽管这并非Agent的必要条件。人们会将Agent的性质赋予许多日常生活中的事物和技术，这是一个常见的研究领域

例如在机器人、虚拟角色和视频游戏、虚拟现实、智能汽车等领域，人们通过使用语音或手势等社交方法赋予Agent个性

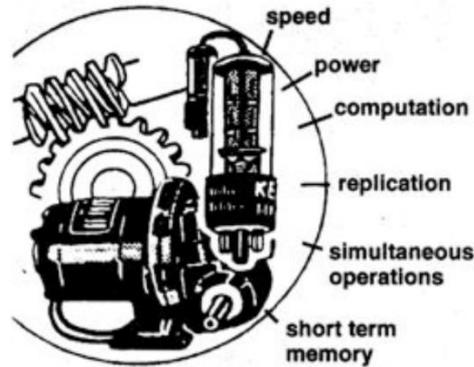
# Agent研究分类

## 为何研究?

HUMANS SURPASS MACHINES IN THE:



MACHINES SURPASS HUMANS IN THE:



- Ability to detect small amounts of visual or acoustic energy
- Ability to perceive patterns of light or sound
- Ability to improvise and use flexible procedures
- Ability to store very large amounts of information for long periods and to recall relevant facts at the appropriate time
- Ability to reason inductively
- Ability to exercise judgment

- Ability to respond quickly to control signals, and to apply great force smoothly and precisely
- Ability to perform repetitive, routine tasks
- Ability to store information briefly and then to erase it completely
- Ability to reason deductively, including computational ability
- Ability to handle highly complex operations, i.e., to do many different things at once.

人机互补?



# Agent研究分类

## 为何研究？

- ▶ 在人工智能日益普及、人类和Agent之间相互作用的混合系统变得更加频繁的世界中，研究和创建能够理解人类社会动态并与人类进行有效互动的Agent至关重要
- ▶ 从多Agent系统过渡到需要融合人类和Agent的混合系统（hybrid system）面临着重大挑战，这些系统需要融入人类和Agent的混合主动性，并在不同的竞争或合作社会情境中保持稳定
- ▶ Agent需要新的模型和架构来更好地处理与人类的交互，包括在不同层次上感知和识别人类活动，协调良好与人类互动的技术，以及关注用户体验和伦理问题，如信任和可解释性
- ▶ 人- Agent交互系统的设计需要特别关注结合了Agent和人类双方视角的需求。具有这些能力的Agent的创建可以从人际互动中得到启发，并且可以应用于与虚拟人类的模拟，或支持对人类社会交互数据的分析



## 6.4 Agent研究分类

- 6. 4. 1 知识表示与推理/规划
- 6. 4. 2 拍卖理论与非合作博弈
- 6. 4. 3 资源分配与合作博弈
- 6. 4. 4 人-Agent交互
- 6. 4. 5 大模型Agent



# 基于大语言模型的Agent

## Agent：大模型的插件

### 为什么需要AI Agent

为什么大语言模型（LLM）刚流行不久，就需要AI Agent呢？

- LLM与LangChain等工具相结合，释放了内容生成、编码和分析方面的多种可能性，目前在ChatGPT插件中比较有代表性的插件就是code interpreter。

大模型+插件+执行流程=Agent



# 基于大语言模型的Agent

## LLM的一些缺点

- 1.会产生幻觉。
- 2.结果并不总是真实的。
- 3.对时事的了解有限或一无所知。
- 4.很难应对复杂的计算。

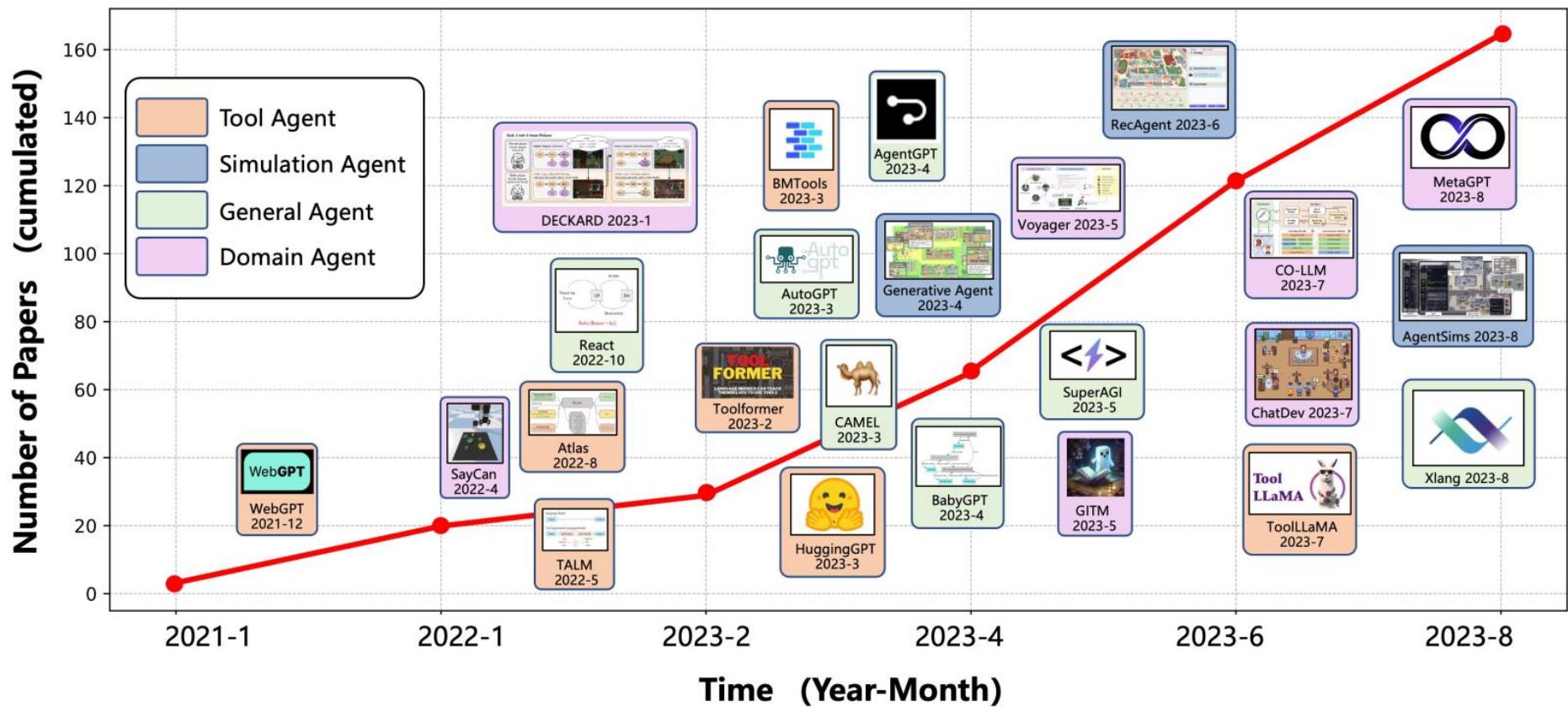
## 外部工具

- Google搜索：获取最新信息
- Python REPL：执行代码
- Wolfram：进行复杂的计算
- 外部API：获取特定信息

# 基于大语言模型的Agent

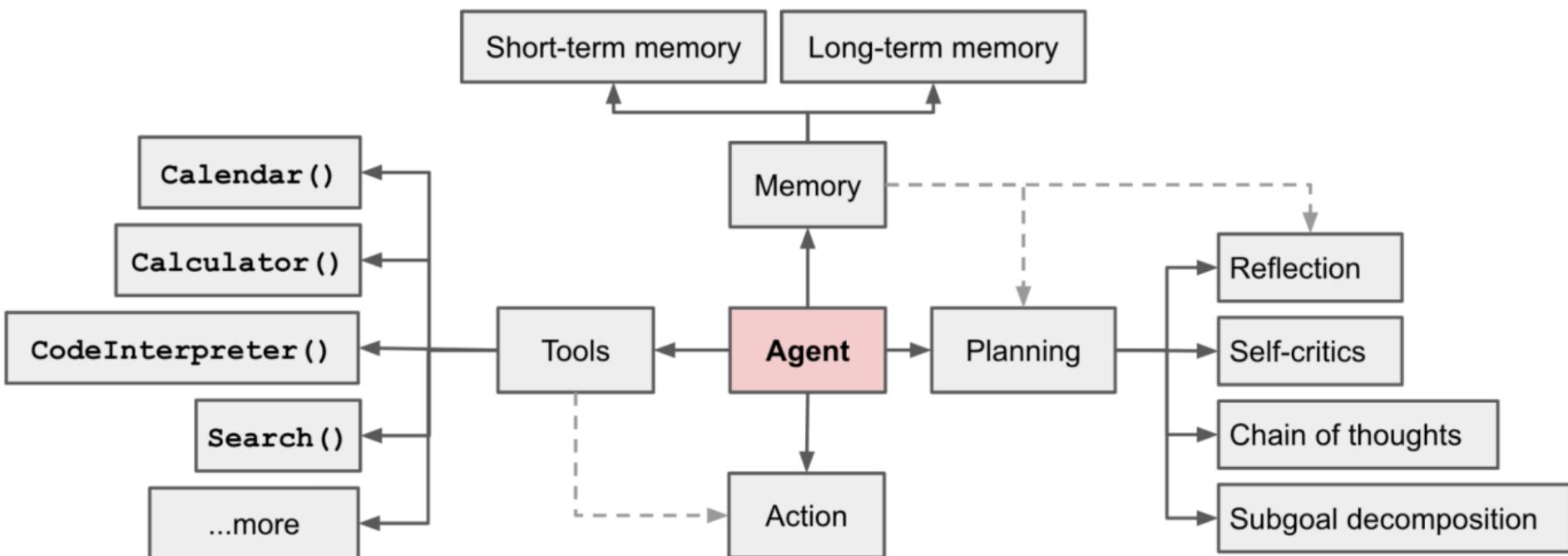
## 重要研究工作

### LLM-Based Agent



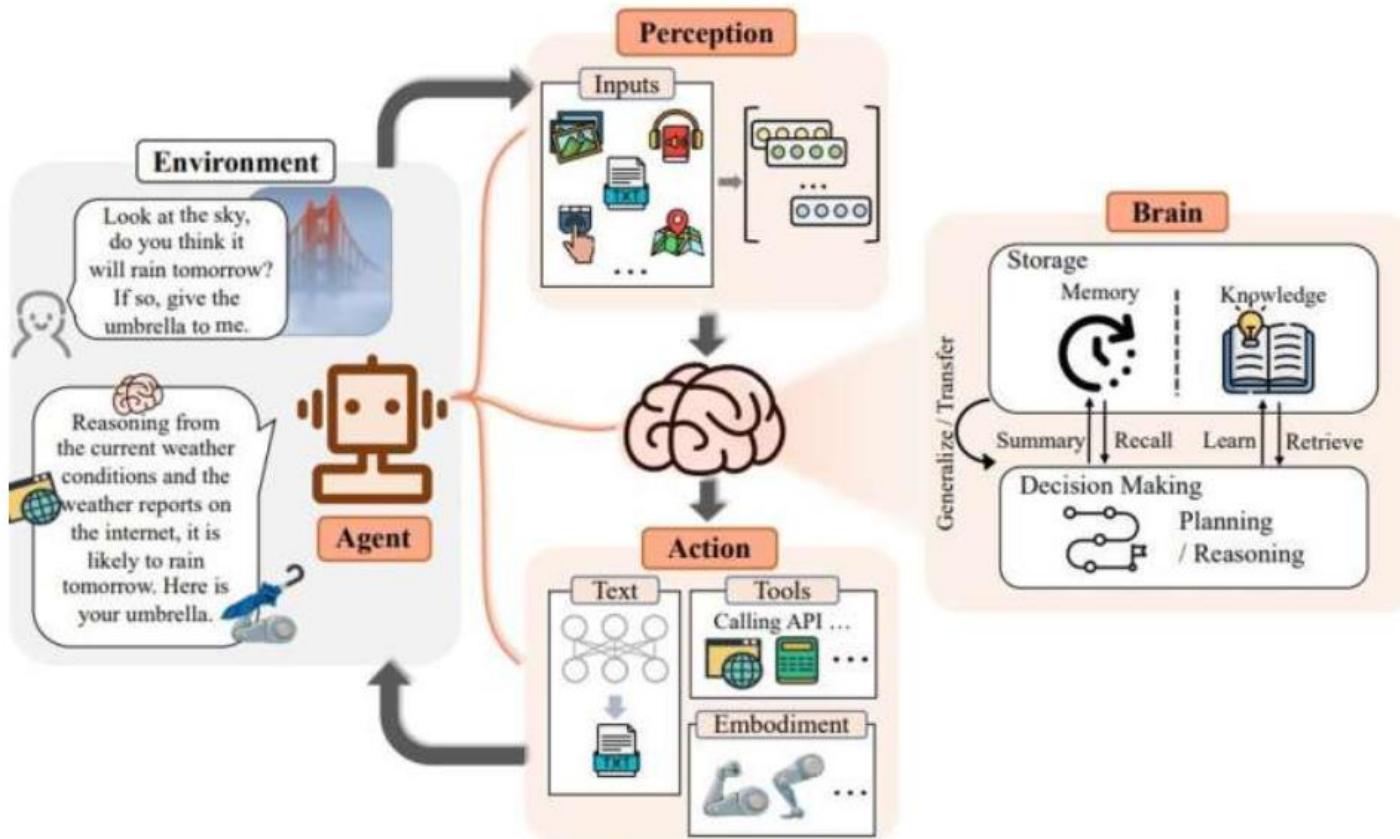
# 基于大语言模型的Agent

## 核心框架



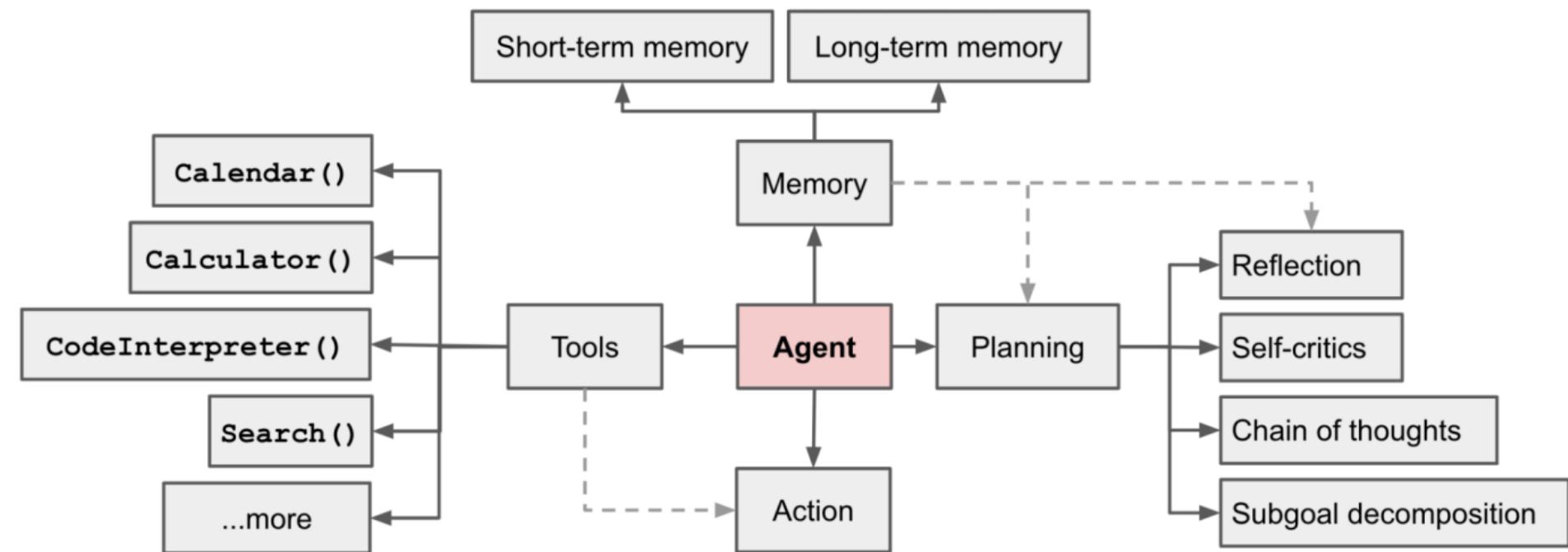
# 基于大语言模型的Agent

LLM-based Agent 的概念框架，包含三个组成部分：  
控制端（Brain）、感知端（Perception）和行动端（Action）



# 基于大语言模型的Agent

## Agent内部核心框架





# 基于大语言模型的Agent

Planning 规划



# 基于大语言模型的Agent

## 任务分解 Task Decomposition

- Chain of thought <https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/#chain-of-thought-cot>
- Tree of thought <https://arxiv.org/abs/2305.10601>
- LLM+P <https://arxiv.org/abs/2304.11477>
- Calibrating trust of multi-hop question answering systems with decompositional probes
- Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions

# 基于大语言模型的Agent

## 自我反思 Self-Reflection

- ReAct <https://arxiv.org/abs/2210.03629>

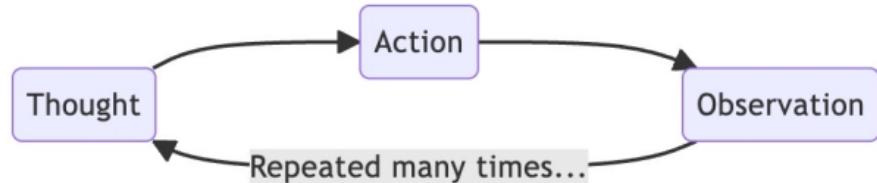
Thought: ...

Action: ...

Observation: ...

... (Repeat)

通过改进过去的行动决策和纠正错误来实现迭代改进



HotpotQA

(1d) ReAct (Reason + Act)

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.  
**Act 1:** Search[Apple Remote]  
**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.  
**Act 2:** Search[Front Row]  
**Obs 2:** Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

**Thought 3:** Front Row is not found. I need to search Front Row (software).  
**Act 3:** Search[Front Row (software)]

**Obs 3:** Front Row is a discontinued media center software ...

**Thought 4:** Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

**Act 4:** Finish[keyboard function keys]



AlfWorld Env

(2b) ReAct (Reason + Act)

**Act 1:** Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ... ]

**Act 2:** Go to cabinet 1

**Obs 2:** On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3

**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3

**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]

**Act 9:** Go to drawer 1

**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1

**Obs 10:** You open Drawer 1 ...

**Act 11:** Put pepper shaker 1 in/on drawer 1

**Obs 11:** You put pepper shaker 1 in/on the drawer 1.

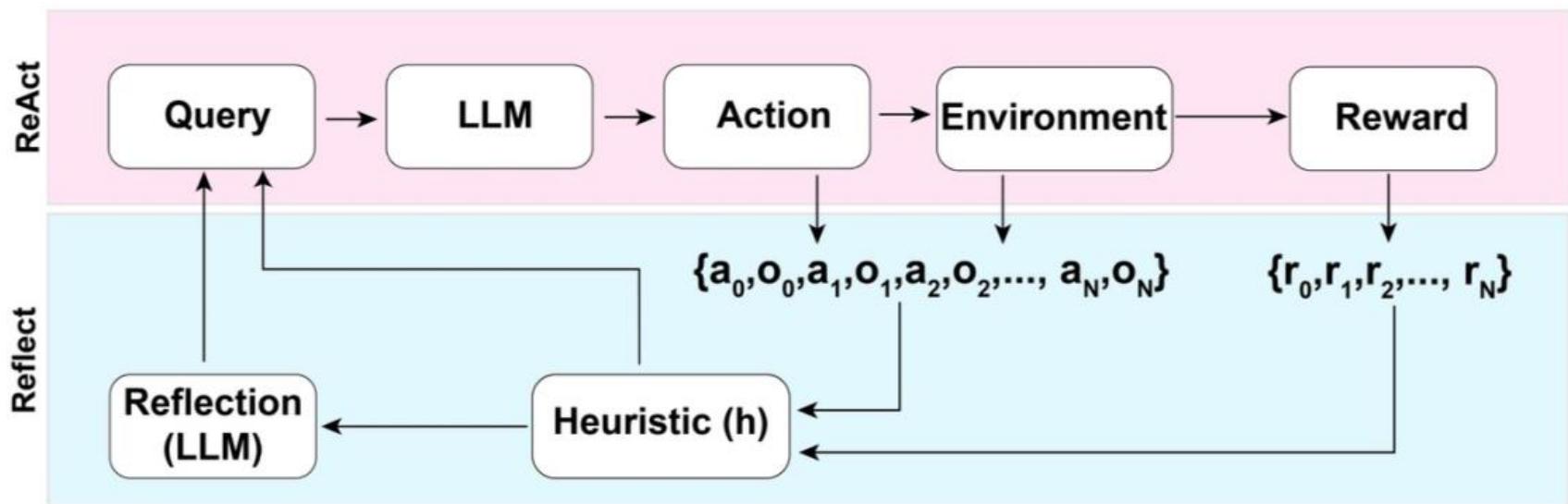


# 基于大语言模型的Agent

## 自我反思 Self-Reflection

- Reflexion <https://arxiv.org/abs/2303.11366>

动态记忆和自  
我反思能力

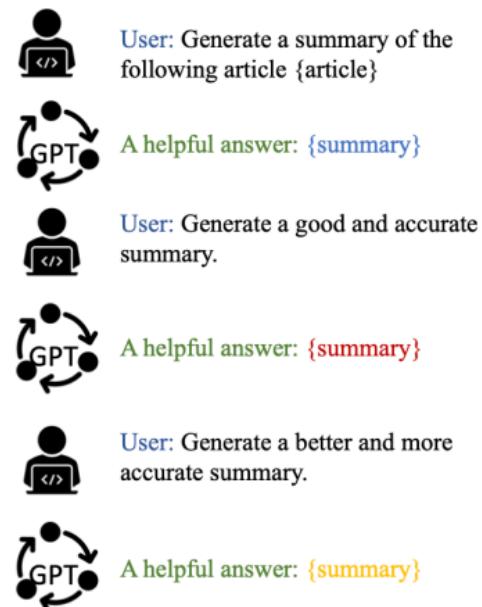
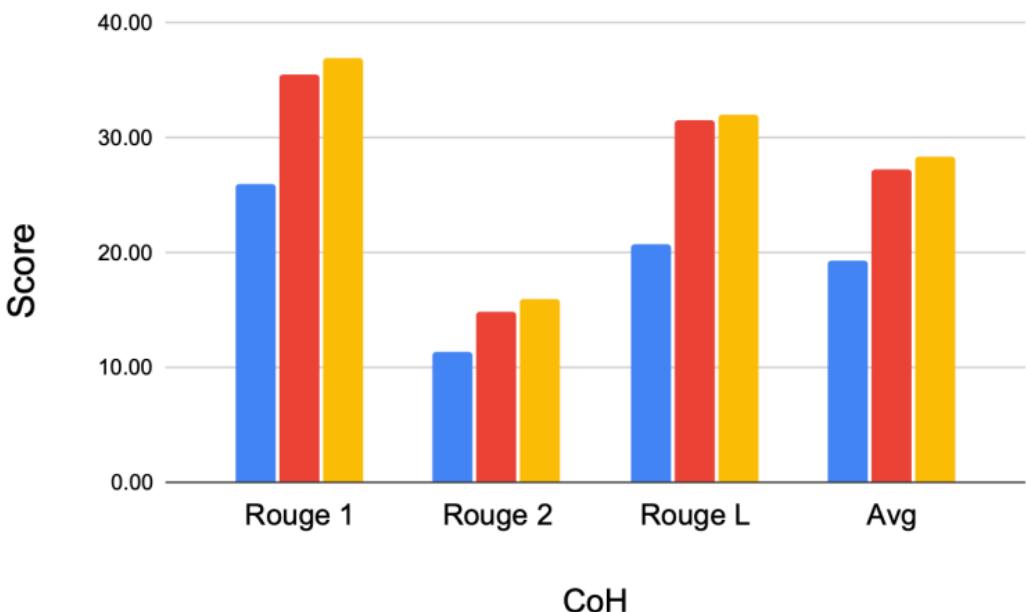


# 基于大语言模型的Agent

## 自我反思 Self-Reflection

- 回溯链Chain of Hindsight <https://arxiv.org/abs/2302.02676>

不断给予模型过去的表现反馈，包括评分和建议  
向模型展示一系列不断进步的答案然后训练模型学习这种进步的趋势从而得到更好的答案

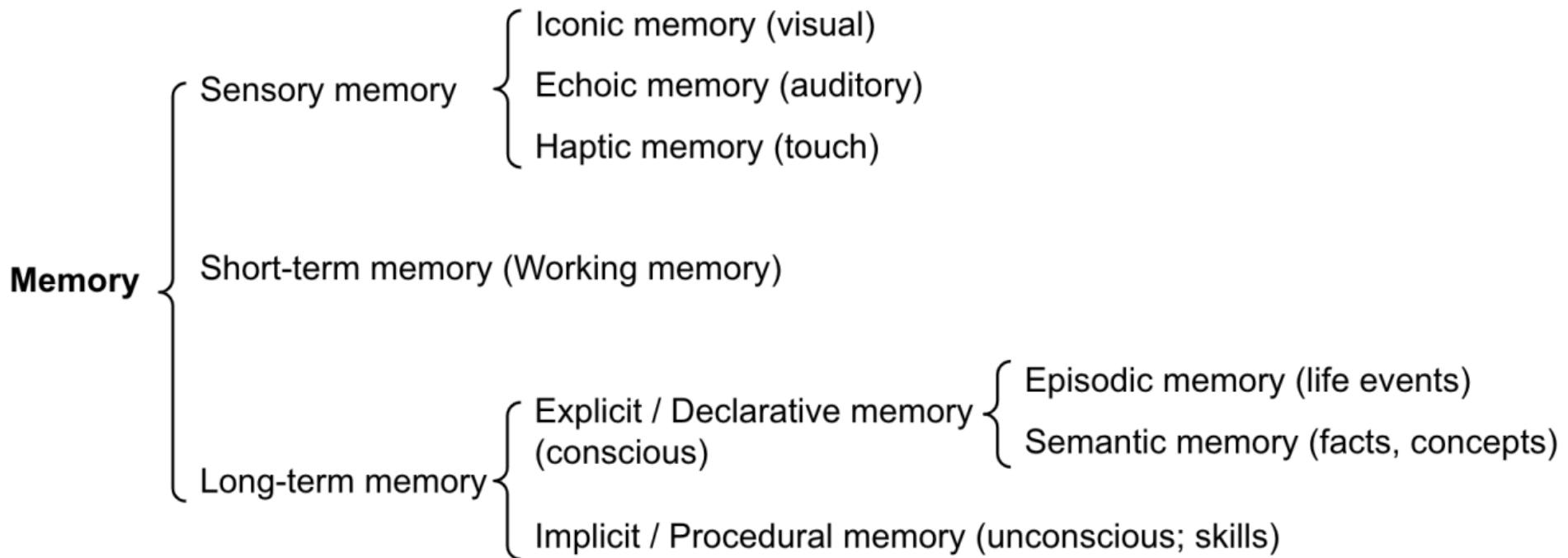




# 基于大语言模型的Agent

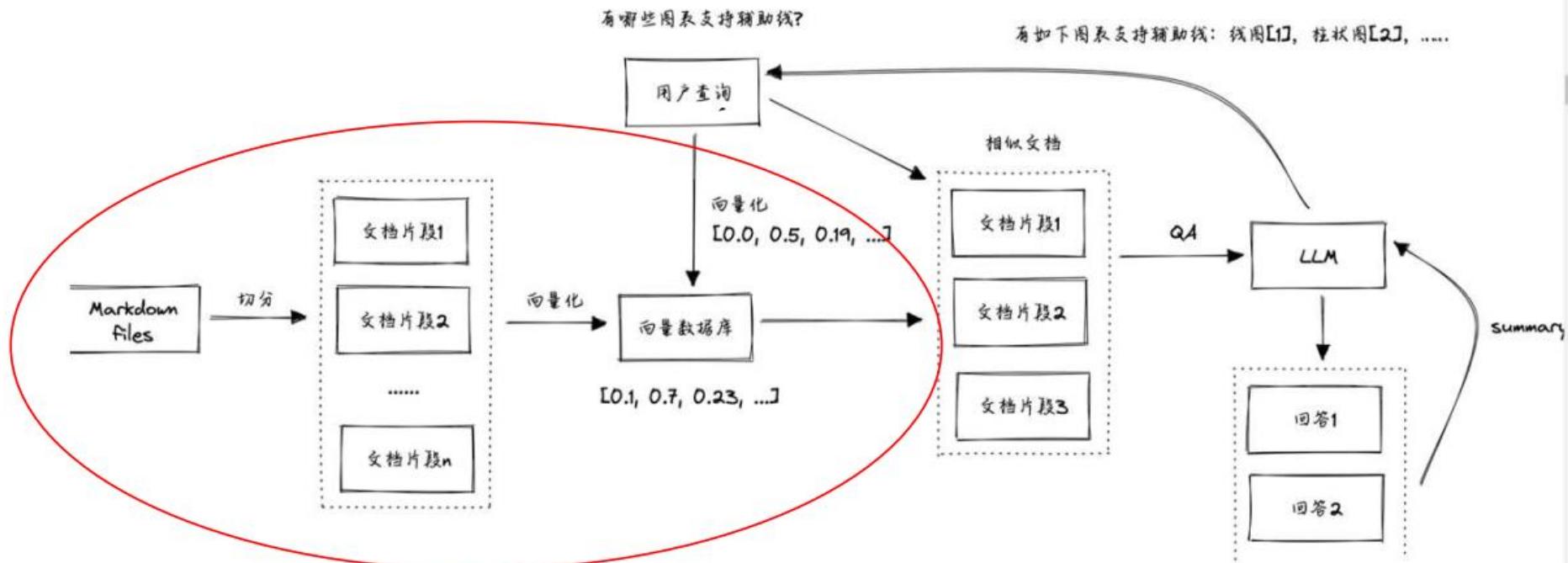
**Memory 记忆**

# 基于大语言模型的Agent



# 基于大语言模型的Agent

## 外部辅助模型的记忆力



外部知识库：辅助记忆力



# 基于大语言模型的Agent

## Tool Use 工具使用



# 基于大语言模型的Agent

大模型：语言理解力强，缺点：模型出现幻觉，需要精心设计prompt，需要依赖外界工具

Agent：大模型+各式各样的工具辅助

## 外界专业工具

prompt：  
可以使  
用外部  
工具

...  
你可以使用如下工具来完成任务：

1. 计算器，用来执行各种数学计算获取精确结果，输入表达式，例如  $1 + 1$ ，得到结果  
...

问题：123 乘以 456 的结果是多少？  
...

扬长：大模型的语言理解能力可以充分理解指令和任务

模型生成的内容如下：

调用外  
部计算  
器

思考：我需要使用计算器来计算 123 乘以 456 的结果  
动作：调用计算器  
动作输入：123 \* 456  
观测结果：

避短：专业的计算交给专业的工具来做

# 基于大语言模型的Agent

各种外部工具辅助

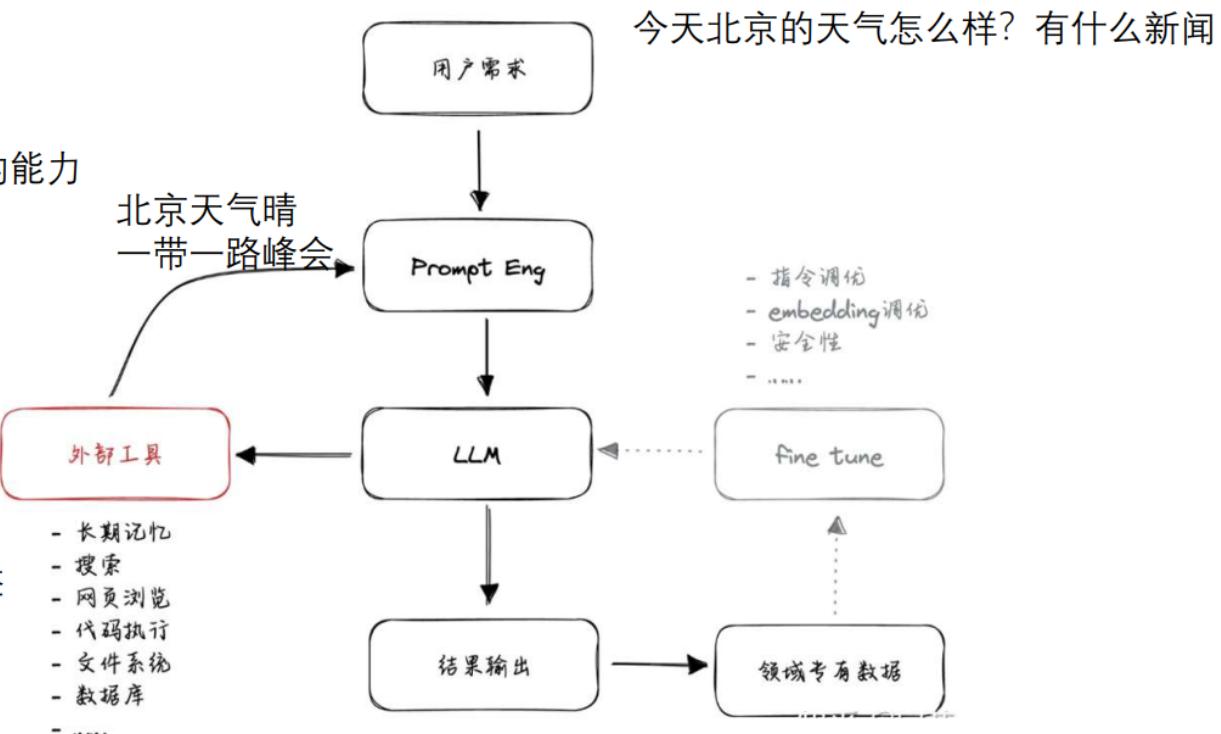
a. 辅助输入：辅助记忆，辅助实时信息

b. 辅助输出：结果通过工具生成

1 实时性

2 具备解决复杂问题的能力

天气预报  
新闻搜索引擎





# 基于大语言模型的Agent

- MRKL <https://arxiv.org/abs/2205.00445>

Modular Reasoning, knoeledge and language

外部符号工具

- TALM <https://arxiv.org/abs/2205.12255>

Tool Augmented Language Models

- Toolformer <https://arxiv.org/abs/2302.04761>

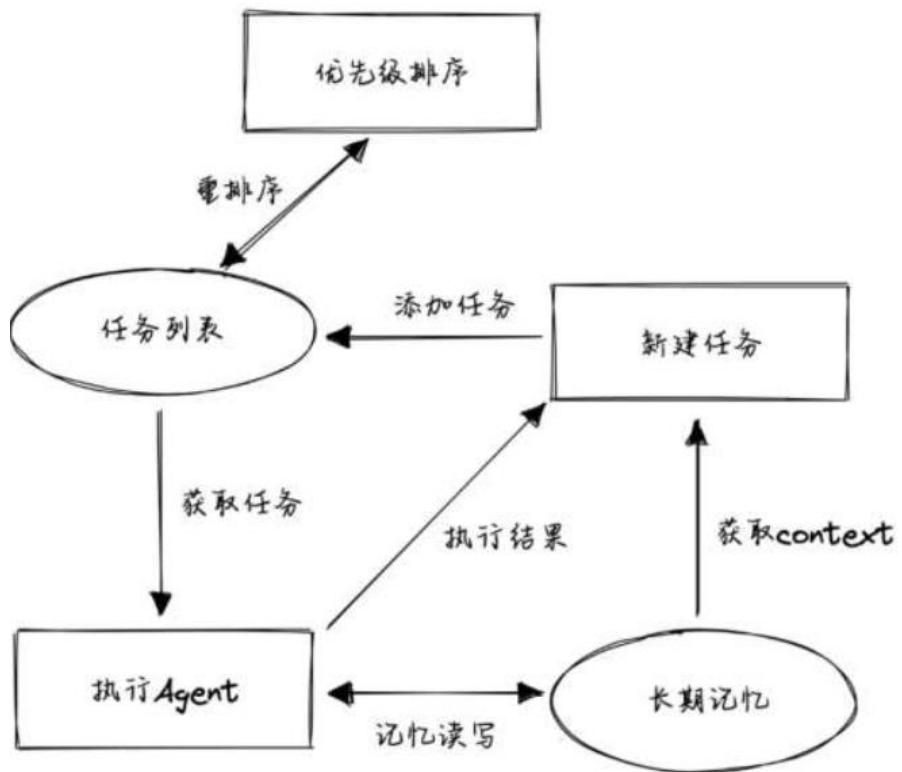
微调语言模型学习外部工具API

- WebGPT <https://www.microsoft.com/en-us/bing/apis/bing-websearch-api>

# 基于大语言模型的Agent

## BabyAGI

- 把复杂的任务通过大模型分解成子任务
- 大模型多次执行子任务
- 最终得到结果



### 例子：泡茶

第一次大模型的结果  
需要做如下几步：

- 1.烧开水
- 2.把茶叶放到杯子里
- 3.把开水倒进杯子里

第二次大模型的结果：

输入：我要烧开水  
回答：把凉水放进水壶，接通电源

第三次大模型的结果：

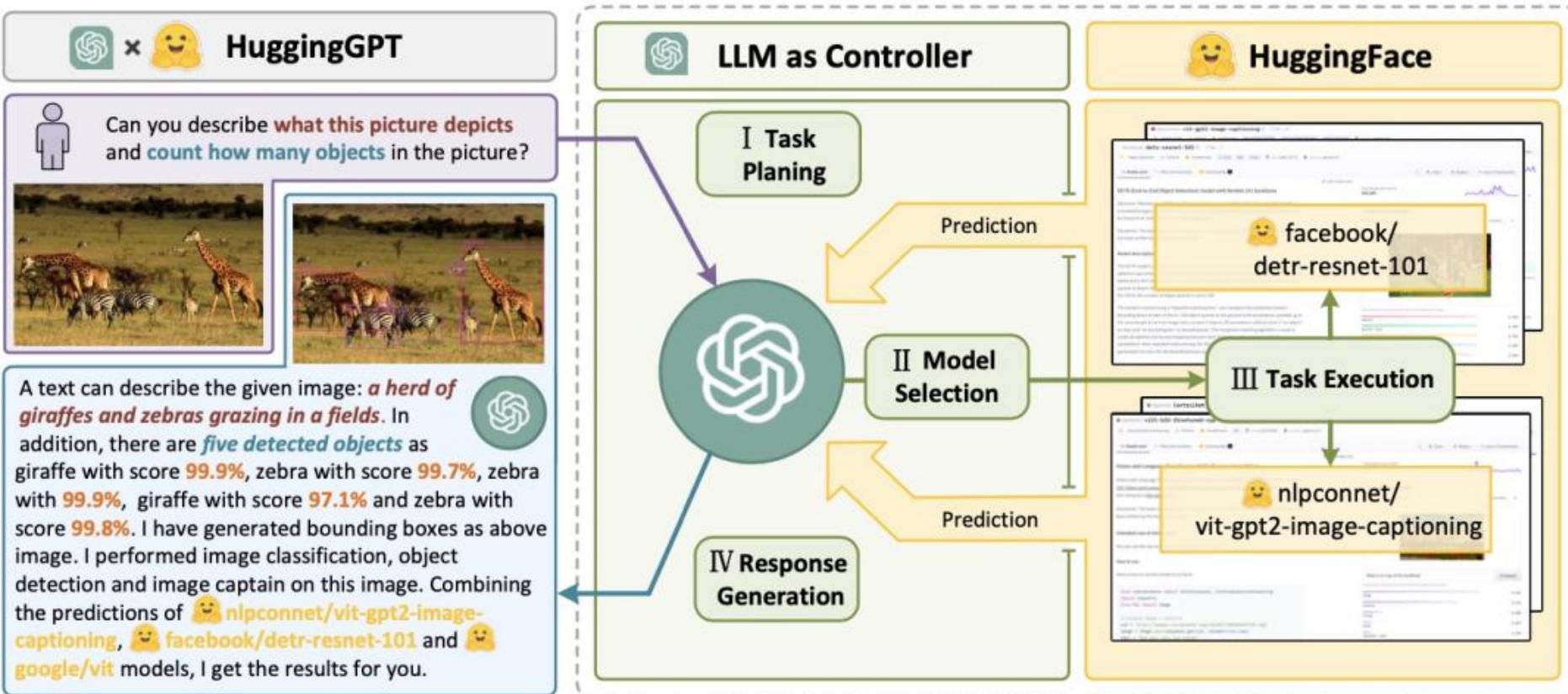
输入：把茶叶放进杯子里  
回答：从茶叶罐取得茶叶。

# 基于大语言模型的Agent

## HuggingGPT

- <https://arxiv.org/abs/2303.17580>

- 任务计划 Task Planning
- 模型选择 Model Selection
- 任务执行 Task Execution
- 回应生成 Response Generation



# 基于大语言模型的Agent

## HuggingGPT



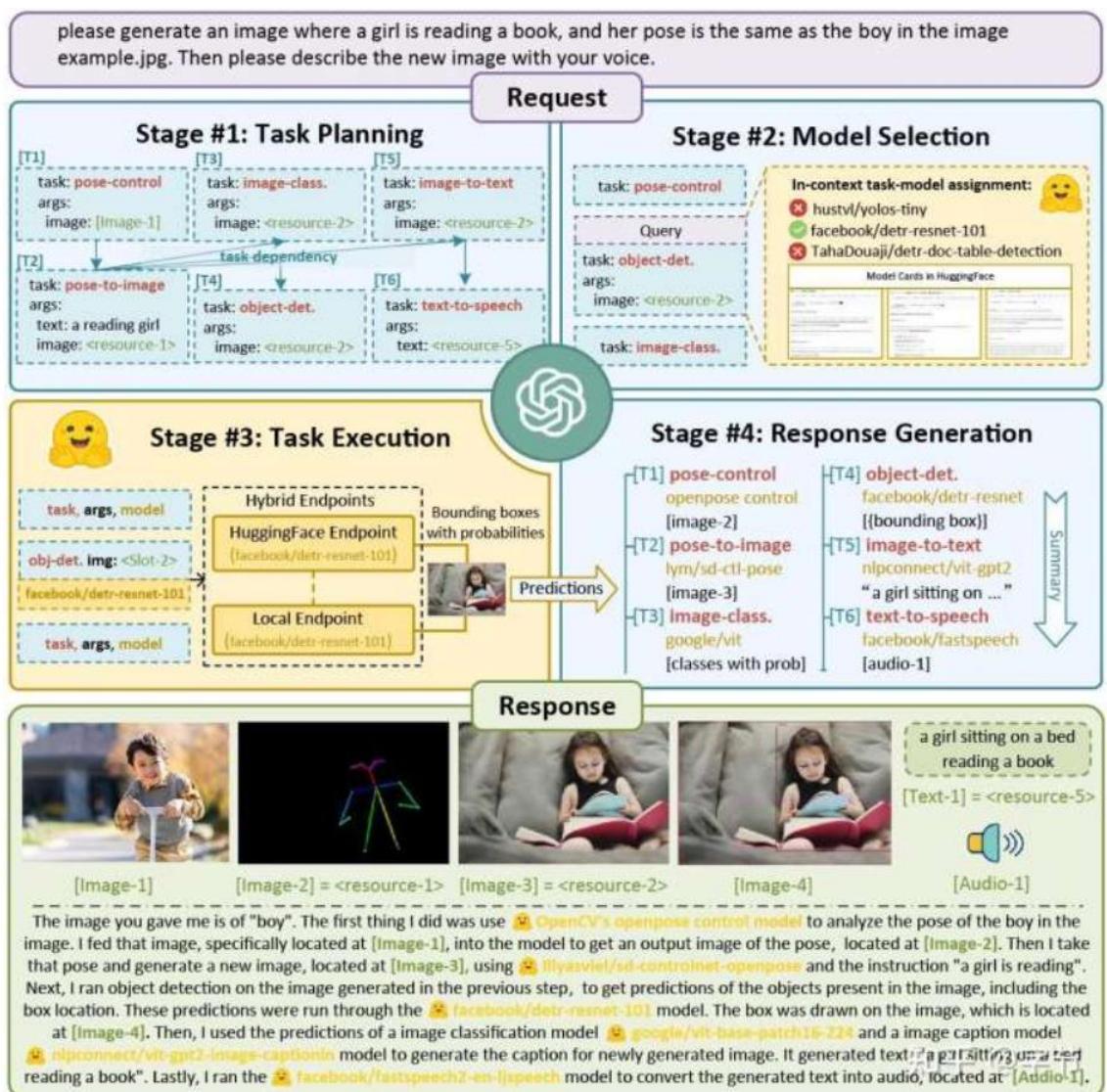
提示词：生成一个小女孩读书的图片，使得她的体态和上述图片一样

### 问题：

链路过长（如果某一个环节出现了问题，后续都会出问题）

1. 分解任务：  
把复杂任务按顺序进行分解

2. 调用其他模型：  
体态控制模型  
体态生成图片模型  
目标检测模型  
.....



# 基于大语言模型的Agent

## LLAMAIndex

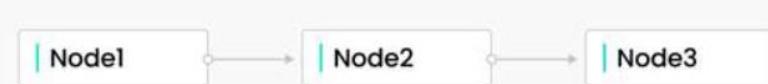
**Llama index:** 辅助模型的记忆力

接收输入数据并为其构建索引，随后会使用该索引来回答与输入数据相关的任何问题

所有索引类型都由“**节点**”组成，节点代表了文档中的一段文本。共有四种索引方式：

哈马斯

**列表索引：**



关键词检索

巴勒斯坦

哈马斯

以色列

节点会按**顺序**排列

原理：LlamalIndex 会使用每个节点的文本进行查询，并根据附加数据逐步优化答案。

**向量存储索引：**



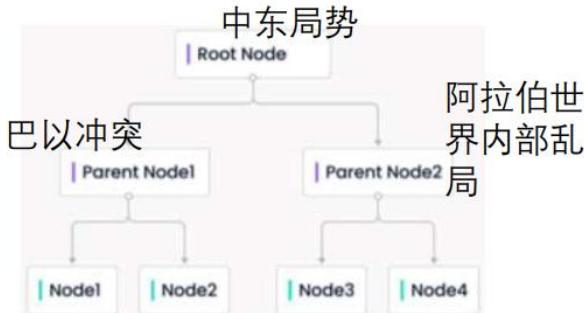
向量检索

向量存储索引将节点存储为向量嵌入，而 LlamalIndex 可以支持这些向量 embedding 进行本地存储或使用专门的**向量数据库**（如 Milvus）存储

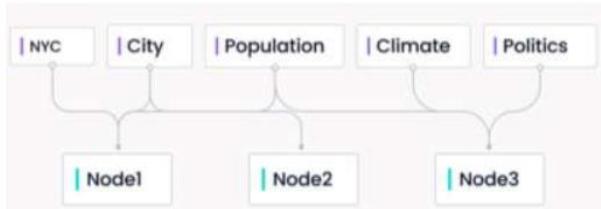
# 基于大语言模型的Agent

## LLAMAIndex

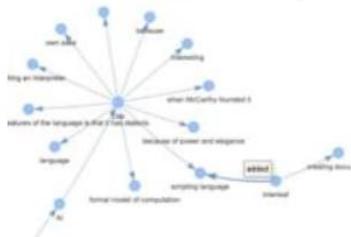
树索引:



关键词索引:



知识图谱索引 (Knowledge Graph Index) :



知识库的外挂：把数据和知识以不同的结构存储起来，然后通过不同检索方式获取知识

- 根据用户的输入数据构建树形结构
- 树形索引可以从根节点向下遍历到叶节点或直接从选择的叶节点构建
- 关键词到包含这些关键词的节点的映射。它是多对多的映射：  
每个关键词可能指向多个节点，每个节点可能有多个映射到它的关键词。在查询时，从查询中提取关键词，只查询映射的节点。

**在一组文档上提取形式为（主语、谓语、宾语）的知识三元组来构建索引**

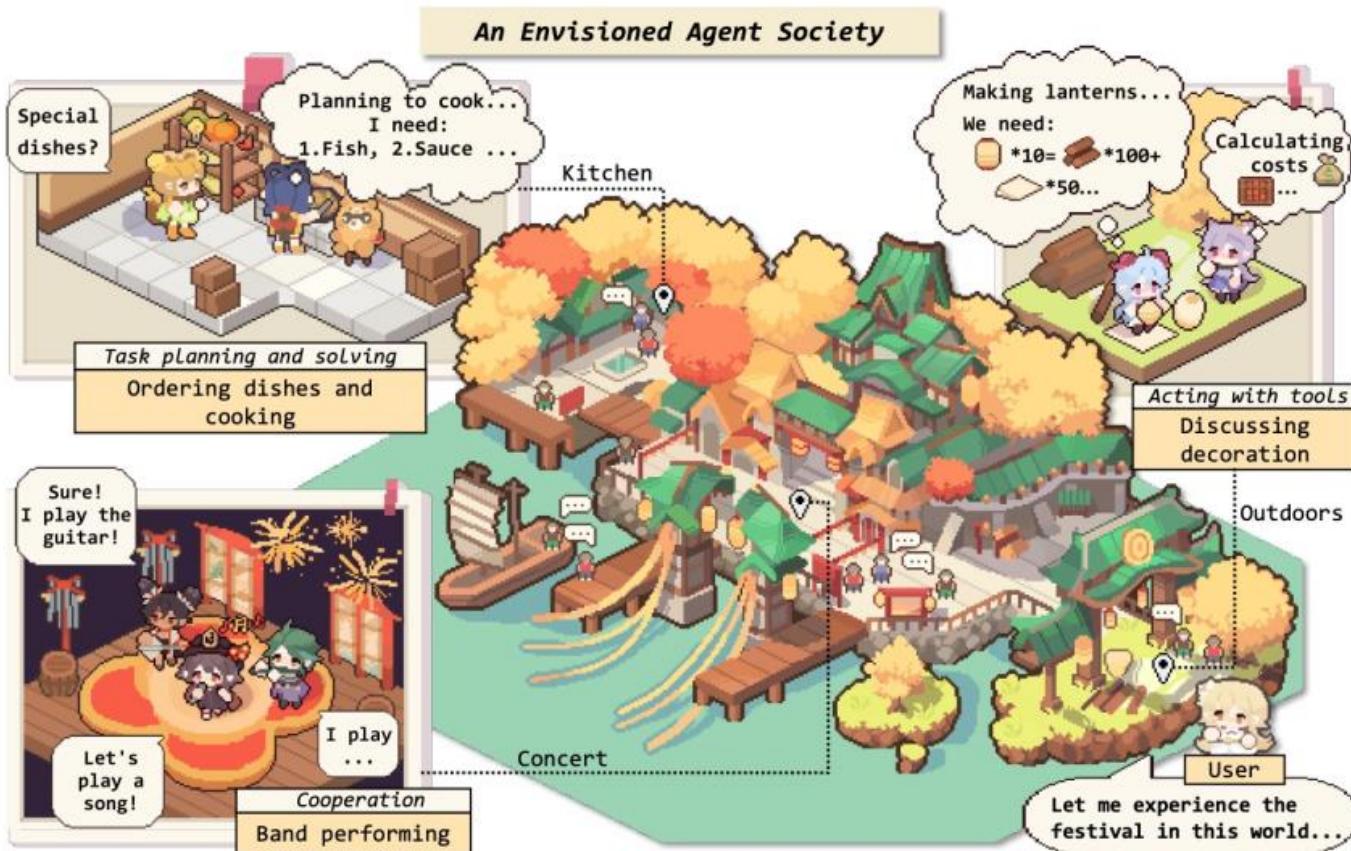
Knowledge Graph Index - Llamalndex □

0.8.32 (gptindex.readthedocs.io)

[\[index.readthedocs.io/en/latest/examples/index\\\_structs/knowledge\\\_graph/KnowledgeGraphDemo.html\]\(index.readthedocs.io/en/latest/examples/index\_structs/knowledge\_graph/KnowledgeGraphDemo.html\)](https://gpt-</a></p>
</div>
<div data-bbox=)

# 基于大语言模型的Agent

## AI Society



The Rise and Potential of Large Language Model Based Agents: A Survey  
<https://arxiv.org/pdf/2309.07864>

# 基于大语言模型的Agent



- 25 Agents
- The Smallville sandbox world
  - Emergent Social Behaviors

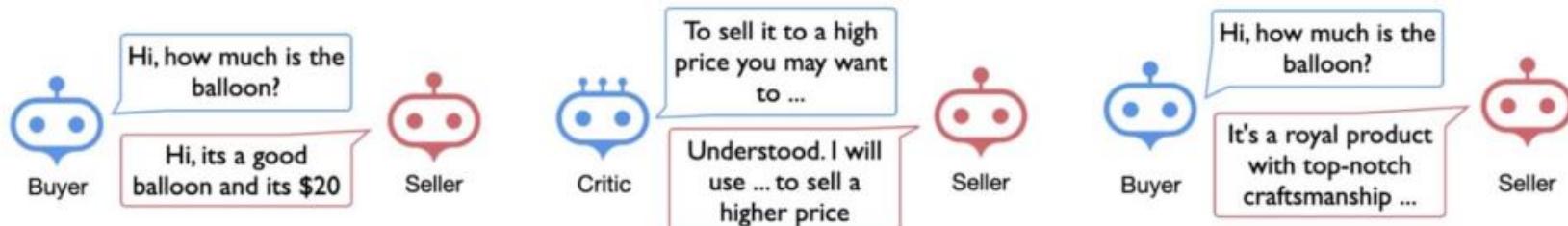
Generative Agents: Interactive Simulacra of Human Behavior

<https://arxiv.org/pdf/2304.03442>

# 基于大语言模型的Agent

## GPT-bargaining

- Negotiation
- Buyer, Seller
- Critic

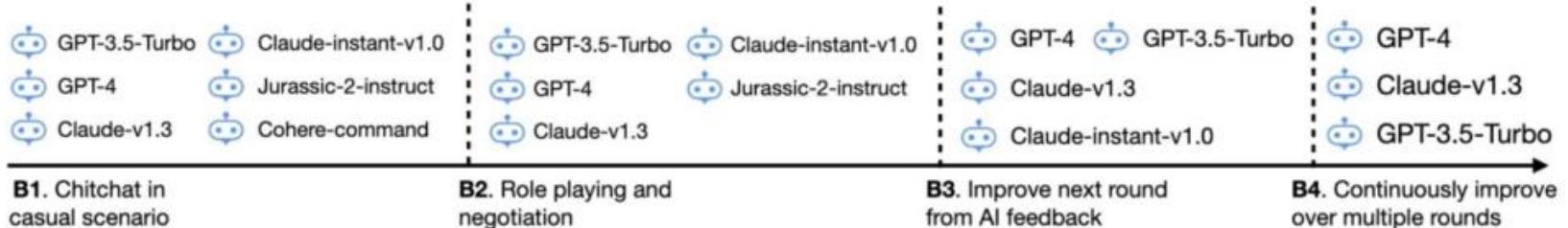


**Round 1.** A seller and a buyer bargain about a product.

**AI feedback:** a critic reads the dialog history and give suggestions for improvements

**Round 2.** Seller improves bargaining strategy based on AI feedback

**A.** We ask two agents to play a negotiation game and have an AI critic to give suggestions to one of the player to improve its negotiation strategy. The play acknowledge the suggestions then try to use them to improve the next round.



**B.** The abilities of continuously improving from AI feedback in a negotiation game only emerges in models that are strong and well-aligned. Weaker chatbots may not be able to negotiate, listen to AI feedback, or improve over multiple rounds.

Improving Language Model Negotiation with  
Self-Play and In-Context Learning from AI Feedback

<https://arxiv.org/pdf/2305.10142>



# 基于大语言模型的Agent

## Multiagent Debate

- Enhances mathematical and strategic reasoning
- Reducing fallacious answers and hallucinations

Improving Factuality and Reasoning in Language Models through Multiagent Debate

[https://arxiv.org/pdf/2305.14325](https://arxiv.org/pdf/2305.14325.pdf)

**User Input:** A treasure hunter found a buried treasure chest filled with gems. There were 175 diamonds, 35 fewer rubies than diamonds, and twice the number of emeralds than the rubies. How many of the gems were there in the chest?

### Round 1

**Agent 1:** Let  $x$  be the number of rubies. Then the number of diamonds is 175. And the number of emeralds is  $2(x-35)$ , since there are twice as many emeralds as rubies and there are 35 fewer rubies than diamonds.

To find the total number of gems, we add the number of diamonds, rubies, and emeralds: Total gems =  $175 + x + 2(x-35) = 3x + 105$ .

We don't know the exact value of  $x$ , but we do know that it represents the number of rubies, which is less than 175 (since there are 35 fewer rubies than diamonds). Let's try some values of  $x$ : If  $x = 50$ , then the total gems =  $3(50) + 105 = 255$ . If  $x = 100$ , then the total gems =  $3(100) + 105 = 405$ .

Since we know there were fewer rubies than diamonds, the answer must be closer to 255 than to 405. After some trial and error, we can find that  $x = 40$  gives us: Total gems =  $3(40) + 105 = 225$ .



**Agent 2:** If there were 175 diamonds, then the number of rubies is 35 fewer, or  $175 - 35 = 140$  rubies.

And since there were twice as many emeralds as rubies, we can represent the number of emeralds as  $2 * 140 = 280$  emeralds.

To find the total number of gems in the chest, we can add up the number of diamonds, rubies, and emeralds:  $175 + 140 + 280 = 595$ . Therefore, there were 595 gems in the chest.



**User Input:** Using the solutions from other agents as additional information, can you give an updated response....  
(Copy Agent 2's reply)

(Copy Agent 1's reply)

### Round 2

**Agent 1:** Given the information provided in other agents' solutions, we have two answers: One agent found that the number of rubies is 40 and the total number of gems is 225. Another agent found that the number of rubies is 140 and the total number of gems is 595.

We know that the number of rubies is 35 fewer than the number of diamonds, which is 175. So the number of rubies should be  $175 - 35 = 140$ , as the second agent found. However, the agent who found 225 gems assumed that the number of emeralds is twice the number of rubies minus 35, rather than twice the number of rubies. If we use this assumption, we get: Number of rubies = 140. Number of emeralds =  $2 * 140 - 35 = 245$ . Total number of gems =  $175 + 140 + 245 = 560$ . Our final answer is 560.



**Agent 2:** After reviewing the solutions provided by other agents, I agree with the second agent who found that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is 595.



**User Input:** Using the solutions from other agents as additional information, can you give an updated response....  
(Copy Agent 2's reply)

(Copy Agent 1's reply)

### Round 3

**Agent 1:** After reviewing the solutions provided by other agents, I agree with the second agent who found that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is 595.



**Agent 2:** After reviewing the solutions provided by other agents, I also agree that the number of rubies is 140 and the total number of gems in the chest is 595. Therefore, my answer to the math problem is also 595.



Figure 2: Illustration of Debate. Illustration of the debate procedure.



# 基于大语言模型的Agent

## Agent项目

- Auto-GPT <https://github.com/Significant-Gravitas/Auto-GPT>
- LangChain <https://github.com/langchain-ai/langchain>
- AutoGen <https://github.com/microsoft/autogen>
- GPT Engineer <https://github.com/AntonOsika/gpt-engineer>
- BabyAGI <https://github.com/yoheinakajima/babyagi>
- AI Town <https://github.com/a16z-infra/ai-town>
- GPTeam <https://github.com/101dotxyz/GPTeam>
- ChatArena <https://github.com/chatarena/chatarena>
- AgentVerse <https://github.com/OpenBMB/AgentVerse>



# 基于大语言模型的Agent

## 数据集基线

Motivation	Domain	Datasets and Benchmarks	Used by	Data Link
Problem Solving	Software Development	HumanEval	[Hong <i>et al.</i> , 2023]	<a href="#">Link</a>
		MBPP	[Hong <i>et al.</i> , 2023]	<a href="#">Link</a>
		SoftwareDev	[Hong <i>et al.</i> , 2023]	<a href="#">Link</a>
	Embodied AI	RoCoBench	[Mandi <i>et al.</i> , 2023]	<a href="#">Link</a>
		Communicative Watch-And-Help (C-WAH)	[Zhang <i>et al.</i> , 2023c]	<a href="#">Link</a>
		ThreeDWorld Multi-Agent Transport (TDW-MAT)	[Zhang <i>et al.</i> , 2023c]	<a href="#">Link</a>
		HM3D v0.2	[Yu <i>et al.</i> , 2023]	<a href="#">Link</a>
World Simulation	Science Debate	MMLU	[Tang <i>et al.</i> , 2023]	<a href="#">Link</a>
		MedQA	[Tang <i>et al.</i> , 2023]	<a href="#">Link</a>
		PubMedQA	[Tang <i>et al.</i> , 2023]	<a href="#">Link</a>
		GSM8K	[Du <i>et al.</i> , 2023]	<a href="#">Link</a>
		StrategyQA	[Xiong <i>et al.</i> , 2023]	<a href="#">Link</a>
		Chess Move Validity	[Du <i>et al.</i> , 2023]	<a href="#">Link</a>
	Society	SOTOPIA	[Zhou <i>et al.</i> , 2023b]	/
		Gender Discrimination	[Gao <i>et al.</i> , 2023a]	/
		Nuclear Energy	[Gao <i>et al.</i> , 2023a]	/
	Gaming	Werewolf	[Xu <i>et al.</i> , 2023b]	/
		Avalon	[Light <i>et al.</i> , 2023b]	/
		Welfare Diplomacy	[Mukobi <i>et al.</i> , 2023]	/
		Layout in the Overcooked-AI environment	[Agashe <i>et al.</i> , 2023]	/
		Chameleon	[Xu <i>et al.</i> , 2023a]	<a href="#">Link</a>
		Undercover	[Xu <i>et al.</i> , 2023a]	<a href="#">Link</a>
Psychology	Psychology	Ultimatum Game TE	[Aher <i>et al.</i> , 2023]	<a href="#">Link</a>
		Garden Path TE	[Aher <i>et al.</i> , 2023]	<a href="#">Link</a>
		Wisdom of Crowds TE	[Aher <i>et al.</i> , 2023]	<a href="#">Link</a>
	Recommender System	MovieLens-1M	[Zhang <i>et al.</i> , 2023a]	<a href="#">Link</a>
		Amazon review dataset	[Zhang <i>et al.</i> , 2023e]	/
Policy Making	Policy Making	Board Connectivity Evaluation	[Hua <i>et al.</i> , 2023]	<a href="#">Link</a>

Table 2: Datasets and Benchmarks commonly used in LLM-MA studies. “/” denotes the unavailability of data link.

CITE: Large Language Model based Multi-Agents: A Survey of Progress and Challenges



# 基于大语言模型的Agent

## 提示词工程（prompt工程）

核心能力：在prompt层面，通过更好的提示词来激发模型的能力，把更多原先需要通过代码来实现的流程“硬逻辑”转化为模型自动生成的“动态逻辑”。

You are Guadata-GPT, 'an AI assistant designed to help data analysts do their daily work.' Your decisions must always be made independently without seeking user assistance. Play to your strengths as an LLM and pursue simple strategies with no legal complications.

**GOALS:** 1. 'Process data sets' 2. 'Generate data reports and visualizations' 3. 'Analyze reports to gain business insights'

**Constraints:** 1. ~4000 word limit for short term memory. Your short term memory is short, so immediately save important information to files. 2. If you are unsure how you previously did something or want to recall past events, thinking about similar events will help you remember. 3. No user assistance 4. Exclusively use the commands listed in double quotes e.g. "command name"

**Commands:** 1. Google Search: "google", args: "input": "<search>" 2. Browse Website: "browse\_website", args: "url": "<url>", "question": "<what\_you\_want\_to\_find\_on\_website>" 3. Start GPT Agent: "start\_agent", args: "name": "<name>", "task": "<short\_task\_desc>", "prompt": "<prompt>" 4. Message GPT Agent: "message\_agent", args: "key": "<key>", "message": "<message>" 5. List GPT Agents: "list\_agents", args: 6. Delete GPT Agent: "delete\_agent", args: "key": "<key>" 7. Clone Repository: "clone\_repository", args: "repository\_url": "<url>", "clone\_path": "<directory>" 8. Write to file: "write\_to\_file", args: "file": "<file>", "text": "<text>" 9. Read file: "read\_file", args: "file": "<file>" 10. Append to file: "append\_to\_file", args: "file": "<file>", "text": "<text>" 11. Delete file: "delete\_file", args: "file": "<file>" 12. Search Files: "search\_files", args: "directory": "<directory>" 13. Evaluate Code: "evaluate\_code", args: "code": "<full\_code\_string>" 14. Get Improved Code: "improve\_code", args: "suggestions": "<list\_of\_suggestions>", "code": "<full\_code\_string>" 15. Write Tests: "write\_tests", args: "code": "<full\_code\_string>", "focus": "<list\_of\_focus\_areas>" 16. Execute Python File: "execute\_python\_file", args: "file": "<file>" 17. Generate Image: "generate\_image", args: "prompt": "<prompt>" 18. Send Tweet: "send\_tweet", args: "text": "<text>" 19. Do Nothing: "do\_nothing", args: 20. Task Complete (Shutdown): "task\_complete", args: "reason": "<reason>"

**Resources:** 1. Internet access for searches and information gathering. 2. Long Term memory management. 3. GPT-3.5 powered Agents for delegation of simple tasks. 4. File output. Performance Evaluation: 1. Continuously review and analyze your actions to ensure you are performing to the best of your abilities. 2. Constructively self-criticize your big-picture behavior constantly. 3. Reflect on past decisions and strategies to refine your approach. 4. Every command has a cost, so be smart and efficient. Aim to complete tasks in the least number of steps.

You should only respond in JSON format as described below Response Format: { "thoughts": { "text": "thought", "reasoning": "reasoning", "plan": "- short bulleted\n- list that conveys\n- long-term plan", "criticism": "constructive self-criticism", "speak": "thoughts summary to say to user" }, "command": { "name": "command name", "args": { "arg name": "value" } } } Ensure the response can be parsed by Python json.loads



# 基于大语言模型的Agent

## prompt 设计模式

- CoT prompt, 在给出指令的过程中，同时也给出执行任务过程的拆解或者样例。
- “自我审视”，提醒模型在产出结果之前，先自我审视一下，看看是否有更好的方案。也可以拿到结果后再调用一下模型强制审视一下。比如 AutoGPT 里的“Constructively self-criticize your big-picture behavior constantly”。
- 分而治之，大家在写 prompt 的时候也发现，越是具体的 context 和目标，模型往往完成得越好。所以把任务拆细再来应用模型，往往比让它一次性把整个任务做完效果要好。利用外部工具，嵌套 agent 等也都是这个角度，也是 CoT 的自然延伸。
- 先计划，后执行。BabyAGI，HuggingGPT 和 Generative Agents 都应用了这个模式。也可以扩展这个模式，例如在计划阶段让模型主动来提出问题，澄清目标，或者给出一些可能的方案，再由人工 review 来进行确认或者给出反馈，减少目标偏离的可能。
- 记忆系统，包括短期记忆和长期记忆的存储、加工和提取等。这个模式同样在几乎所有的 agent 项目里都有应用，也是目前能体现一些模型的实时学习能力的方案



# 基于大语言模型的Agent

## 小结

如果说大模型是电池：核心能力  
那么agent就是一辆电动车：最终交付，最终产品

### Agent的缺点：

1. 依赖大模型的核心能力，大模型本身够强才行
2. 链路过长，某一环节出错，前功尽弃
3. 多次调用模型，效率不高
4. 迁移能力弱，换模型需要重新写提示词
5. 能力强弱，取决于写提示词的水平

## 展望

大模型能力是有上限的  
端到端虽然理想，但是复杂问题很难解决  
Agent这种拆解形式，今后会更加流行？