

第2章

智能体

我们在此讨论智能体的本质，完美的或不完美的、环境的多样性以及由此产生的智能体类型的集合。

第1章将**理性智能体**（rational agent）的概念确定为研究人工智能的方法的核心。本章将使这个概念更加具体。我们将看到，在任何可以想象的环境中运行的各种智能体都可以应用理性的概念。本书的计划是使用这个概念来制定一小组设计原则，并用于构建成功的智能体，可以合理地称之为**智能系统**。

我们从检查智能体、环境以及它们之间的耦合开始。观察到某些智能体比其他智能体表现得更好，可以自然而然地引出理性智能体的概念，即行为尽可能好。智能体的行为取决于环境的性质。我们将对环境进行粗略分类，并展示环境的属性如何影响智能体的设计。我们描述一些基本的“框架”智能体设计，本书余下的部分将充实相关内容。

2.1 智能体和环境

任何通过**传感器**（sensor）感知**环境**（environment）并通过**执行器**（actuator）作用于该环境的事物都可以被视为**智能体**（agent）。这个简单的想法如图2-1所示。一个人类智能体以眼睛、耳朵和其他器官作为传感器，以手、腿、声道等作为执行器。机器人智能体可能以摄像头和红外测距仪作为传感器，还有各种电动机作为执行器。软件智能体接收文件内容、网络数据包和人工输入（键盘/鼠标/触摸屏/语音）作为传感输入，并通过写入文件、发送网络数据包、显示信息或生成声音对环境进行操作。环境可以是一切，甚至是整个宇宙！实际上，我们在设计智能体时关心的只是宇宙中某一部分的状态，即影响智能体感知以及受智能体动作影响的部分。

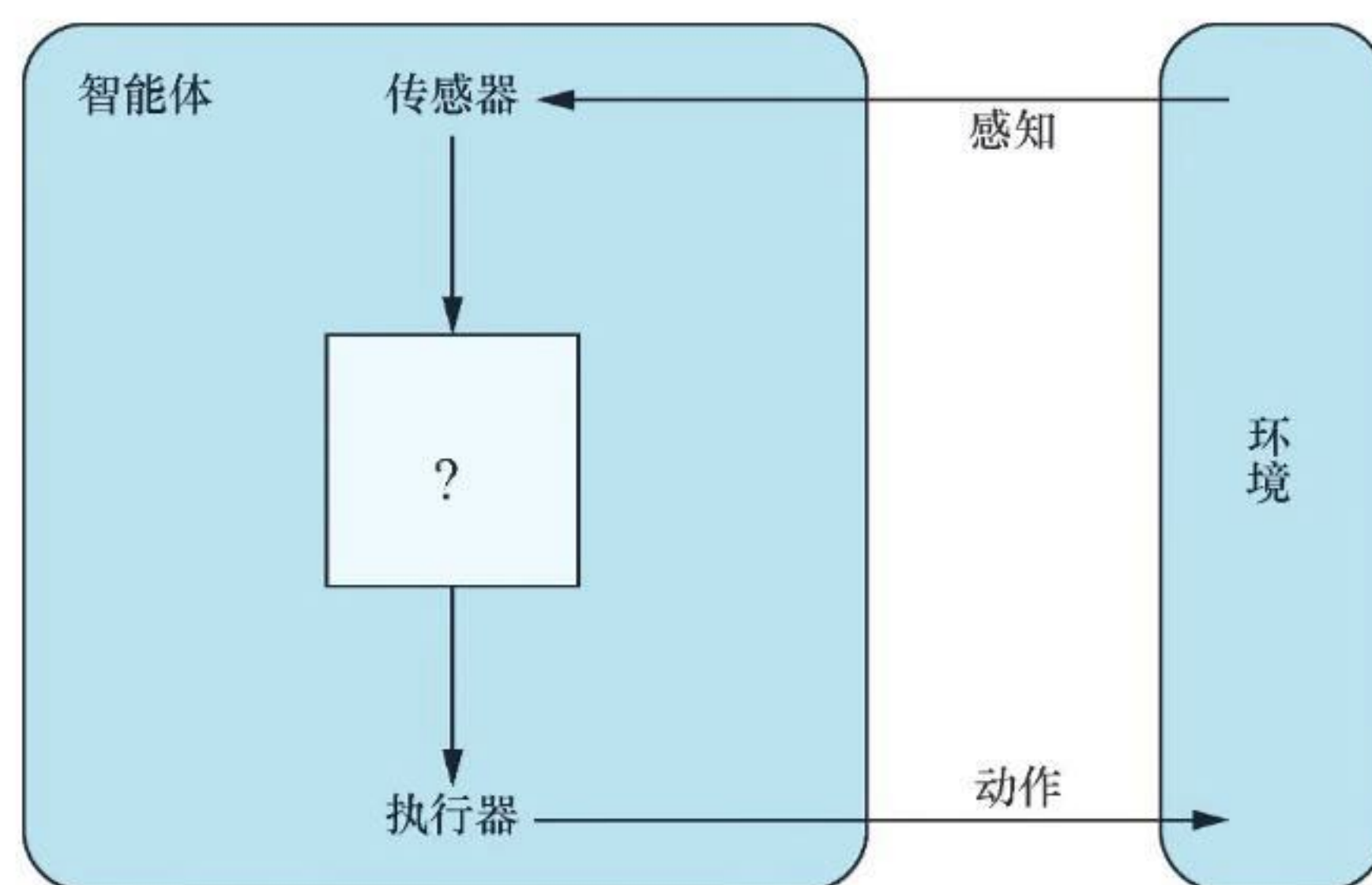


图 2-1 智能体通过传感器和执行器与环境交互

我们使用术语**感知**（percept）来表示智能体的传感器正在感知的内容。智能体的**感知序列**（percept sequence）是智能体所感知的一切的完整历史。一般而言，一个智能体在任何给定时刻的动作选择可能取决于其内置知识和迄今为止观察到的整个感知序列，而不是它未感知到的任何事物。通过为每个可能的感知序列指定智能体的动作选择，我们或多或少地说明了关于智能体的所有内容。从数学上讲，我们说智能体的行为由**智能体函数**（agent function）描述，该函数将任意给定的感知序列映射到一个动作。

可以想象将描述任何给定智能体的智能体函数制成表格。对大多数智能体来说，这将是一个非常大的表，事实上是无限的（除非限制考虑的感知序列长度）。给定一个要进行实验的智能体，原则上，我们可以通过尝试所有可能的感知序列并记录智能体响应的动作来构建此表^①。当然，该表只是该智能体的外部特征。在内部，人工智能体的智能体函数将由**智能体程序**（agent program）实现。区别这两种观点很重要，智能体函数是一种抽象的数学描述，而智能体程序是一个具体的实现，可以在某些物理系统中运行。

为了阐明这些想法，我们举一个简单的例子——真空吸尘器世界。在一个由方格组成的世界中，包含一个机器人真空吸尘器智能体，其中的方格可以是脏的，也可以是干净的。图 2-2 展示了只有两个方格——方格 A 和方格 B——的情况。真空吸尘器智能体可以感知它在哪个方格中，以及方格中是否干净。智能体从方格 A 开始。可选的操作包括向右移动、向左移动、吸尘或什么都不做。^② 一个非常简单的智能体函数如下：如果当前方格是脏的，就吸尘；否则，移动到另一个方格。该智能体函数的部分表格如图 2-3 所示，实现它的智能体程序如图 2-8 所示。

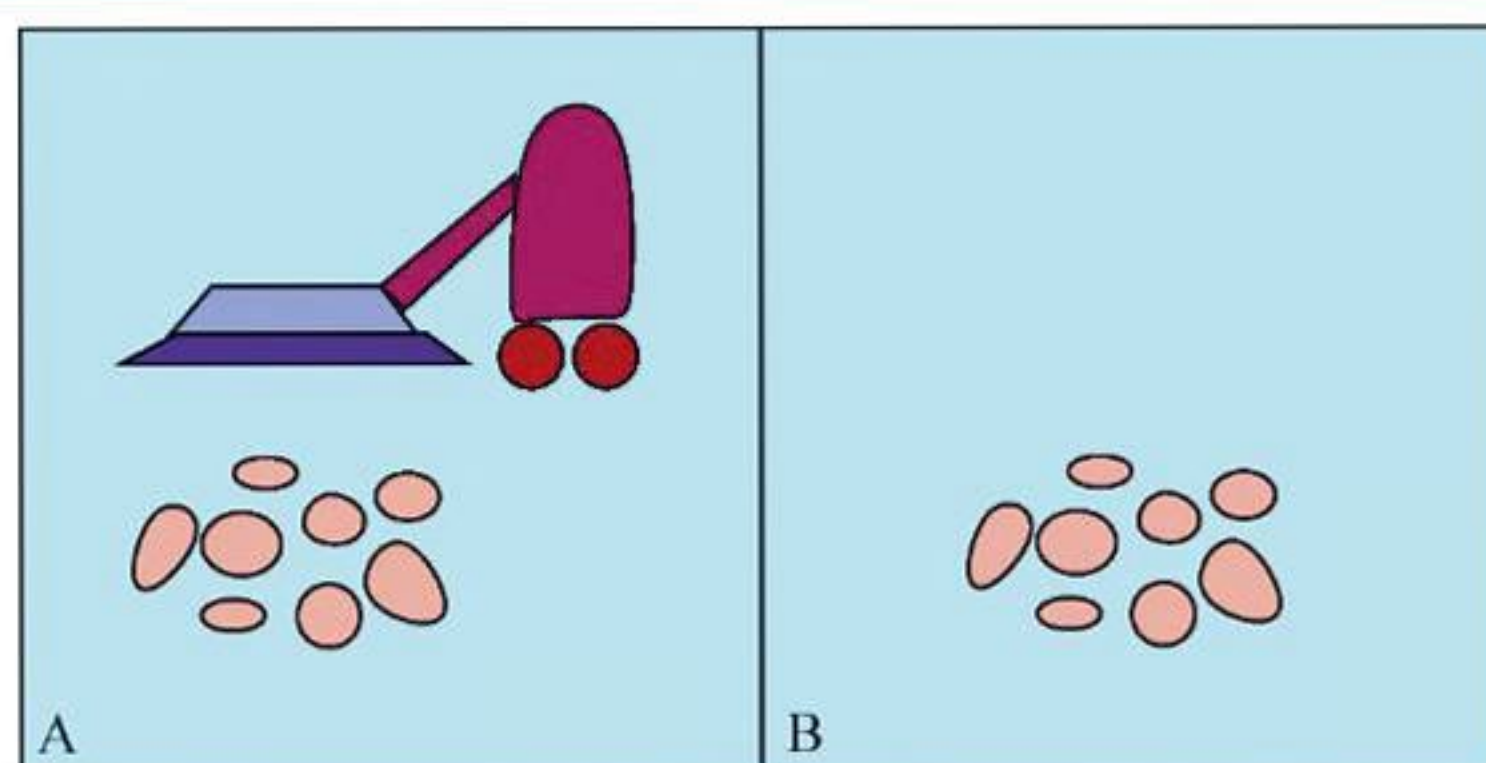


图 2-2 一个只有两个方格的真空吸尘器世界。每个位置可以是干净的，也可以是脏的，智能体可以向左移动或向右移动，可以清理它所占据的方格。不同版本的真空吸尘器世界允许不同的规则，例如智能体可以感知什么，它的动作是否总是成功等

从图 2-3 中可以看到，通过以各种方式填充右边的列可以简单地定义各种真空世界的智能体。那么，显而易见的问题是：填充表格的正确方法是什么？换句话说，是什么使智能体表现好或坏、聪明或愚蠢？我们将在 2.2 节中回答这些问题。

在结束本节之前，我们应该强调，智能体这一概念旨在成为分析系统的工具，而不是将世界划分为智能体和非智能体的绝对表征。人们可以将手持计算器视为一个智能体，它在给定感知序列“ $2+2=$ ”时选择显示“4”的动作，但这样的分析很难帮助我们理解计算器。在某种意义上，工程的所有领域都可以被视为设计与世界互动的人工制品，人工智能运行在（作者认为是）这个系列最有趣的一端，在这一端，人工制品具有重要的计算资源，任务环境需要非凡的决策。

- ① 如果智能体在选择其动作时使用一些随机化，那就必须多次尝试每个序列来确定每个动作的概率。有人可能会认为随机地行动是相当愚蠢的，但本章后面展示出它可能非常聪明。
- ② 真正的机器人不太可能会有“向右移动”和“向左移动”这样的动作，而是采用“向前旋转轮子”和“向后旋转轮子”这样的动作。我们选择的动作更易于在书本上理解，而不是为了在实际的机器人中易于实现。

感知序列	动作
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

图 2-3 图 2-2 所示的真空吸尘器世界的简单智能体函数的部分表项。如果当前方格是脏的，智能体就会进行清理，否则它将移到另一个方格。注意，除非限制可能感知序列的长度，否则该表的大小是无限的

2.2 良好行为：理性的概念

理性智能体（rational agent）是做正确事情的事物。显然，做正确的事情比做错误的事情要好，但是做正确的事情意味着什么呢？

2.2.1 性能度量

道德哲学发展了几种不同“正确事情”的概念，但人工智能通常坚持一种称为**结果主义**（consequentialism）的概念：我们通过结果来评估智能体的行为。当智能体进入环境时，它会根据接受的感知产生一个动作序列。这一动作序列会导致环境经历一系列的状态。如果序列是理想的，则智能体表现良好。这种可取性的概念由**性能度量**（performance measure）描述，该度量评估任何给定环境状态的序列。

人类有自己的欲望和偏好，因此，人类有适用于自身的理性概念。这一概念与成功地选择产生环境状态序列的行动有关，这些环境状态序列从人类的角度来看是可取的。但是，机器没有自己的欲望和偏好，至少在最初，性能度量是在机器设计者的头脑中或者是在机器受众的头脑中。我们将看到，一些智能体设计具有性能度量的显式表示（一个版本），而在其他设计中，性能度量完全是隐式的，智能体可能会做正确的事情，但它不知道为什么。

回顾诺伯特·维纳的警告，以确保“施以机器的目的是我们真正想要的目的”（1.5 节），注意，正确地制定性能度量可能非常困难。例如，考虑 2.1 节中的真空吸尘器智能体，我们可能会建议用单个 8 小时班次中清理的灰尘量来度量性能。当然，有了理性的智能体，你所要求的就是你所得到的。然而一个理性的智能体可以通过清理灰尘，然后将其全部倾倒在地板上，然后再次清理，如此反复，从而最大化这一性能度量。更合适的性能度量是奖励拥有干净地板的智能体。例如，在每个时间步中，每个干净的方格都可以获得 1 分（可能会对耗电和产生的噪音进行惩罚）。作为一般规则，更好的做法是根据一个人在环境中真正想要实现的目标，而

不是根据一个人认为智能体应该如何表现来设计性能度量。

即使避免了明显的缺陷，一些棘手的问题仍然存在。例如，上一段中“干净地板”的概念是基于一段时间内的平均整洁度。然而，两个不同的智能体可以达到相同的平均整洁度，其中一个智能体工作始终保持一般水平，而另一个智能体短时间工作效率很高但需要长时间的休息。哪种工作方式更可取似乎是保洁科学的好课题，但实际上这是一个具有深远影响的深刻哲学问题。大起大落、不计后果的生活，和安全但单调的生活，哪个更好？一个人人都生活在中度贫困的经济体，和一个有些人生活富裕而另一些人非常贫困的经济体，哪个更好？我们把这些问题的留给勤奋的读者作为习题。

对于本书的大部分内容，我们将假设性能度量可以正确地指定。然而，出于前面所述原因，我们必须接受这样一种可能性：我们可能会将错误的目的施加给机器，确切地说，就是 1.5 节描述的迈达斯国王问题。此外，当设计一款软件（其副本将属于不同的用户）时，我们无法预测每个用户的确切偏好。因此，我们可能需要构建相应的智能体，它能够反映真实性能度量的初始不确定性，并随着时间的推移对其了解更多，第 16 章、第 18 章和第 22 章介绍了此类智能体。

2.2.2 理性

在任何时候，理性取决于以下 4 方面：

- 定义成功标准的性能度量；
- 智能体对环境的先验知识；
- 智能体可以执行的动作；
- 智能体到目前为止的感知序列。

这引出了**理性智能体的定义**：

对于每个可能的感知序列，给定感知序列提供的证据和智能体所拥有的任何先验知识，理性智能体应该选择一个期望最大化其性能度量的动作。

考虑一个简单的真空吸尘器智能体，如果一个方格是脏的就清理它，如果不脏就移动到另一个方格，这就是图 2-3 中给出的智能体函数。它是理性智能体吗？这需要看情况了！首先，我们需要说明性能度量是什么，对环境了解多少，以及智能体具有哪些传感器和执行器。我们假设：

- 在 1000 个时间步的“生命周期”内，性能度量在每个时间步为每个干净的方格奖励 1 分；
- 环境的“地理信息”是先验的（图 2-2），但灰尘的分布和智能体的初始位置不是先验的，干净的方格会继续保持干净，吸尘（*Suck*）动作会清理当前方格，向左（*Left*）或向右（*Right*）的动作使智能体移动一个方格，如果该动作会让智能体移动到环境之外，智能体将保持在原来的位置；
- 可用的动作仅有向右（*Right*）、向左（*Left*）和吸尘（*Suck*）；
- 智能体能够正确感知其位置以及该位置是否有灰尘。

在这种情况下，智能体确实是理性的，它的预期性能至少与任何其他智能体一样。

显而易见，同一个智能体在不同的情况下可能会变得不理性。例如，一旦清除了所有灰尘，该智能体将会毫无必要地反复来回；如果性能度量考虑对每个动作罚 1 分，那么智能体的表现就会很差。在确定所有方格都干净的情况下，一个更好的智能体不会做任何事情。如果干净的方格可能再次变脏，智能体应该偶尔检查，并在必要时重新清理。如果环境的地理信息是未知的，智能体则需要对其进行探索（*explore*）。习题 2.VACR 要求在这些情况下设计智能体。

2.2.3 全知、学习和自主

我们需要仔细区分理性和**全知** (omniscience)。全知的智能体能预知其行动的实际结果，并能据此采取行动，但在现实中，全知是不可能的。考虑这样一个例子：有一天我正沿着香榭丽舍大街散步，我看到街对面的一位老朋友。附近没有车流，我也没有别的事要做，所以理性上，我会开始过马路。与此同时，在 10 千米的高空，一架飞过的客机上有一扇货舱门脱落下来^①，在我到达马路对面之前，我就被压扁了。我过马路是不理性的吗？我的讣告上不太可能写“试图过马路的白痴”。

这个例子表明，理性不等同于完美。理性使期望性能最大化，而完美使实际性能最大化。不要求完美不仅仅是对智能体公平的问题。关键是，如果我们期望一个智能体做事后证明是最好的行动，就不可能设计一个符合规范的智能体，除非我们改进占卜水晶球或时间机器的性能。

因此，我们对理性的定义并不需要全知，因为理性决策只取决于迄今为止的感知序列。我们还必须确保我们没有无意中允许智能体进行低智的行动。例如，如果一个智能体在穿过繁忙的道路之前没有向两边看，那么它的感知序列将不会告诉它有一辆大卡车正在以高速接近。我们对理性的定义是不是说现在就可以过马路了？绝非如此！

首先，考虑到这种缺乏信息的感知序列，过马路是不理性的：不观察路况就过马路发生事故的风险太大。其次，理性智能体在上街之前应该选择“观察”动作，因为观察有助于最大化期望性能。采取行动来改变未来的感知，有时被称为**信息收集** (information gathering)，这是理性的一个重要组成部分，将在第 16 章中详细介绍。信息收集的另一个例子是真空吸尘器在最初未知的环境中必须进行的**探索** (exploration)。

我们的定义要求理性智能体不仅要收集信息，还要尽可能多地从它所感知到的东西中**学习** (learn)。智能体的初始配置可以反映对环境的一些先验知识，但随着智能体获得经验，这可能会被修改和增强。在一些极端情况下，环境完全是先验已知的和完全可预测的。在这种情况下，智能体不需要感知或学习，只需正确地运行。

当然，这样的智能体是脆弱的。如卑微的粪甲虫例子，在挖出巢穴产卵后，它会从附近的一堆粪中取出一团粪来堵住入口。如果粪球在途中被截下，粪甲虫根本不会注意到粪球已经不见了，仍会继续它的任务，并滑稽地用不存在的粪球堵住巢穴。进化已经在粪甲虫的行为中建立了一个假设，当它被违反时，不成功的行为就会产生。

稍微聪明一点的是掘土黄蜂。雌性掘土黄蜂会挖一个洞，出去刺一只毛毛虫并把它拖到洞口，再次进入洞里检查一切是否正常，然后把毛毛虫拖进洞里再去产卵。当蜂卵孵化时，毛毛虫会充当食物来源。到目前为止还不错，但如果昆虫学家在掘土黄蜂检查洞穴时将毛毛虫移动几厘米远，它将回到其规划中的“把毛毛虫拖到洞口”步骤，即使经过数十次移动毛毛虫的干预，它仍然继续执行该规划而不进行修改，不断地重新检查洞穴。掘土黄蜂无法知道其固有规划正在失败，因此不会改变规划。

如果在某种程度上，智能体依赖于其设计者的先验知识，而不是其自身的感知和学习过程，我们就说该智能体缺乏**自主性** (autonomy)。一个理性的智能体应该是自主的，它应该学习如何弥补部分或不正确的先验知识。例如，能学习预测何时何地会出现额外灰尘的真空吸尘器比不能学习预测的要好。

实际上，我们很少从一开始就要求智能体完全自主：除非设计者提供一些帮助，否则当智能体几乎没有经验或完全没有经验时，它将不得不随机行动。正如进化为动物提供了足够的内

^① 参见 N. Henderson, “波音 747 大型喷气式飞机迫切需要新门锁”，华盛顿邮报，1989 年 8 月 24 日。

建反射，使其能够生存足够长的时间来学习一样，为人工智能体提供一些初始知识和学习能力也是合理的。在充分体验相应环境后，理性智能体的行为可以有效地独立于其先验知识。因此，结合学习能够让我们设计单个理性智能体，它能在各种各样的环境中取得成功。

2.3 环境的本质

既然已经有了理性的定义，考虑构建理性智能体的准备几乎已经完成。然而，还必须考虑**任务环境**（task environment），它本质上是“问题”，理性智能体是“解决方案”。我们首先展示如何指定任务环境，并用一些示例说明该过程。然后，展示任务环境的多种形式。任务环境的性质直接影响智能体程序的恰当设计。

2.3.1 指定任务环境

在讨论简单的真空吸尘器智能体的理性时，我们必须指定性能度量、环境以及智能体的执行器和传感器。我们将所有这些都归在任务环境的范畴下，基于首字母缩写规则，我们称其为**PEAS**（Performance, Environment, Actuator, Sensor）描述。在设计智能体时，第一步必须始终是尽可能完整地指定任务环境。

真空吸尘器世界是一个简单的例子，让我们考虑一个更复杂的问题：自动驾驶出租车司机。图 2-4 总结了出租车任务环境的 PEAS 描述。我们将在以下段落中更详细地讨论每个元素。

智能体类型	性能度量	环境	执行器	传感器
自动驾驶出租车司机	安全、速度快、合法、舒适旅程、最大化利润、对其他道路用户的影响最小化	道路、其他交通工具、警察、行人、客户、天气	转向器、加速器、制动、信号、喇叭、显示、语音	摄像头、雷达、速度表、GPS、发动机传感器、加速度表、麦克风、触摸屏

图 2-4 自动驾驶出租车司机任务环境的 PEAS 描述

首先，我们希望自动驾驶追求的**性能度量**（performance measure）是什么？理想的标准包括到达正确的目的地，尽量减少油耗和磨损，尽量减少行程时间或成本，尽量减少违反交通法规和对其他驾驶员的干扰，最大限度地提高安全性和乘客舒适度，最大化利润。显然，其中一些目标是相互冲突的，因此需要权衡。

接下来，出租车将面临什么样的**驾驶环境**（environment）？任何出租车司机都必须能够在各种道路上行驶，如乡村车道、城市小巷以及 12 车道的高速公路。道路上有其他交通工具、行人、流浪动物、道路工程、警车、水坑和坑洼。出租车还必须与潜在以及实际的乘客互动。另外，还有一些可选项。出租车可以选择在很少下雪的南加利福尼亚州或者经常下雪的阿拉斯加运营。它可能总是靠右行驶，或者我们可能希望它足够灵活，在英国或日本时可以靠左行驶。显然，环境越受限，设计问题就越容易解决。

自动驾驶出租车的**执行器**（actuator）包括可供人类驾驶员使用的器件，例如通过加速器控制发动机以及控制转向和制动。此外，它还需要输出到显示屏或语音合成器，以便与乘客进行对话，或许还需要某种方式与其他车辆进行礼貌的或其他方式的沟通。

出租车的基本**传感器**（sensor）将包括一个或多个摄像头以便观察，以及激光雷达和超声波传感器以便检测其他车辆和障碍物的距离。为了避免超速罚单，出租车应该有一个速度表，

而为了正确控制车辆（特别是在弯道上），它应该有一个加速度表。要确定车辆的机械状态，需要发动机、燃油和电气系统的传感器常规阵列。像许多人类驾驶者一样，它可能需要获取GPS信号，这样就不会迷路。最后，乘客需要触摸屏或语音输入才能说明目的地。

图2-5中简要列举了一些其他智能体类型的基本PEAS元素。更多示例参见习题2.PEAS。这些示例包括物理环境和虚拟环境。注意，虚拟任务环境可能与“真实”世界一样复杂。例如，在拍卖和转售网站上进行交易的**软件智能体**（software agent），或称软件机器人或**软机器人**（softbot），为数百万其他用户和数十亿对象提供交易，其中许多对象具有真实的图片。

智能体类型	性能度量	环境	执行器	传感器
医学诊断系统	治愈患者、降低费用	患者、医院、工作人员	用于问题、测试、诊断、治疗的显示器	用于症状和检验结果的触摸屏/语音输入
卫星图像分析系统	正确分类对象和地形	轨道卫星、下行链路、天气	场景分类显示器	高分辨率数字照相机
零件选取机器人	零件在正确箱中的比例	零件输送带、箱子	有关节的手臂和手	摄像头、触觉和关节角度传感器
提炼厂控制器	纯度、产量、安全	提炼厂、原料、操作员	阀门、泵、加热器、搅拌器、显示器	温度传感器、气压传感器、流量传感器、化学传感器
交互英语教师	学生的考试分数	一组学生、考试机构	用于练习、反馈、发言的显示器	键盘输入、语音

图 2-5 智能体类型及其 PEAS 描述的示例

2.3.2 任务环境的属性

人工智能中可能出现的任务环境范围显然非常广泛。然而，我们可以确定相当少的维度，并根据这些维度对任务环境进行分类。这些维度在很大程度上决定了恰当的智能体设计以及智能体实现的主要技术系列的适用性。首先我们列出维度，然后分析几个任务环境，阐明思路。这里的定义是非形式化的，后面的章节提供了每种环境的更精确的陈述和示例。

完全可观测的（fully observable）与**部分可观测的**（partially observable）：如果智能体的传感器能让它在每个时间点都能访问环境的完整状态，那么我们说任务环境是完全可观测的。如果传感器检测到与动作选择相关的所有方面，那么任务环境就是有效的完全可观测的，而所谓的相关又取决于性能度量标准。完全可观测的环境很容易处理，因为智能体不需要维护任何内部状态来追踪世界。由于传感器噪声大且不准确，或者由于传感器数据中缺少部分状态，环境可能部分可观测。例如，只有一个局部灰尘传感器的真空吸尘器无法判断其他方格是否有灰尘，自动驾驶出租车无法感知其他司机的想法。如果智能体根本没有传感器，那么环境是**不可观测的**（unobservable）。在这种情况下，有人可能会认为智能体的困境是无解的，但是正如我们在第4章中讨论的那样，智能体的目标可能仍然可以实现，有时甚至是确定可以实现的。

单智能体的（single-agent）与**多智能体的**（multiagent）：单智能体和多智能体环境之间的区别似乎足够简单。例如，独自解决纵横字谜的智能体显然处于单智能体环境中，而下国际象棋的智能体则处于二智能体环境中。然而，这里也有一些微妙的问题。首先，我们已经描述了如何将一个实体视为智能体，但没有解释哪些实体必须视为智能体。智能体A（例如出租车司机）是否必须将对象B（另一辆车）视为智能体，还是可以仅将其视为根据物理定律运行的对

象，类似于海滩上的波浪或随风飘动的树叶？关键的区别在于 B 的行为是否被最佳地描述为一个性能度量的最大化，而这一性能度量的值取决于智能体 A 的行为。

例如，国际象棋中的对手实体 B 正试图最大化其性能度量，根据国际象棋规则，这将最小化智能体 A 的性能度量。因此，国际象棋是一个**竞争性**（competitive）的多智能体环境。但是，在出租车驾驶环境中，避免碰撞使所有智能体的性能度量最大化，因此它是一个部分**合作的**（cooperative）多智能体环境。它还具有部分竞争性，例如，一个停车位只能停一辆车。

多智能体环境中的智能体设计问题通常与单智能体环境下有较大差异。例如，在多智能体环境中，通信通常作为一种理性行为出现；在某些竞争环境中，随机行为是理性的，因为它避免了一些可预测性的陷阱。

确定性的（deterministic）与**非确定性的**（nondeterministic）：如果环境的下一个状态完全由当前状态和智能体执行的动作决定，那么我们说环境是确定性的，否则是非确定性的。原则上，在完全可观测的确定性环境中，智能体不需要担心不确定性。然而，如果环境是部分可观测的，那么它可能是非确定性的。

大多数真实情况非常复杂，以至于不可能追踪所有未观测到的方面；出于实际目的，必须将其视为非确定性的。从这个意义上讲，出租车驾驶显然是非确定性的，因为人们永远无法准确地预测交通行为。此外，轮胎可能会意外爆胎，发动机可能会在没有警告的情况下失灵。我们描述的真空吸尘器世界是确定性的，但变化可能包括非确定性因素，如随机出现的灰尘和不可靠的吸力机制（参考习题 2.VFIN）。

最后注意一点，**随机的**（stochastic）一词被一些人用作“非确定性”的同义词，但我们会区分这两个术语。如果环境模型显式地处理概率（例如，“明天的降雨可能性为 25%”），那么它是随机的；如果可能性没有被量化，那么它是“非确定性的”（例如，“明天有可能下雨”）。

回合式的（episodic）与**序贯的**（sequential）：在回合式任务环境中，智能体的经验被划分为原子式的回合。在每一回合中，智能体接收一个感知，然后执行单个动作。至关重要的是，下一回合并不依赖于前几回合采取的动作。许多分类任务是回合式的。例如，在装配流水线上检测缺陷零件的智能体需要根据当前零件做出每个决策，而无须考虑以前的决策；而且，当前的决策并不影响下一个零件是否有缺陷。但是，在序贯环境中，当前决策可能会影响未来所有决策。^① 国际象棋和出租车驾驶是序贯的：在这两种情况下，短期行为可能会产生长期影响。因为在回合式环境下智能体不需要提前思考，所以要比序贯环境简单很多。

静态的（static）与**动态的**（dynamic）：如果环境在智能体思考时发生了变化，我们就说该智能体的环境是动态的，否则是静态的。静态环境很容易处理，因为智能体在决定某个操作时不需要一直关注世界，也不需要担心时间的流逝。但是，动态环境会不断地询问智能体想要采取什么行动，如果它还没有决定，那就等同于什么都不做。如果环境本身不会随着时间的推移而改变，但智能体的性能分数会改变，我们就说环境是**半动态的**（semidynamic）。驾驶出租车显然是动态的，因为驾驶算法在计划下一步该做什么时，其他车辆和出租车本身在不断移动。在用时钟计时的情况下国际象棋是半动态的。填字游戏是静态的。

离散的（discrete）与**连续的**（continuous）：离散/连续的区别适用于环境的状态、处理时间的方式以及智能体的感知和动作。例如，国际象棋环境具有有限数量的不同状态（不包括时钟）。国际象棋也有一组离散的感知和动作。驾驶出租车是一个连续状态和连续时间的问题，出租车和其他车辆的速度和位置是一系列连续的值，并随着时间平稳地变化。出租车的驾驶动

^① “sequential”（串行）一词在计算机科学中也被用作“parallel”（并行）的反义词，与此处的含义在很大程度上是不相关的。

作也是连续的（转向角等）。严格来说，来自数字照相机的输入是离散的，但通常被视为表示连续变化的强度和位置。

已知的（known）与**未知的**（unknown）：严格来说，这种区别不是指环境本身，而是指智能体（或设计者）对环境“物理定律”的认知状态。在已知环境中，所有行动的结果（如果环境是非确定性的，则对应结果的概率）都是既定的。显然，如果环境未知，智能体将不得不了解它是如何工作的，才能做出正确的决策。

已知和未知环境之间的区别与完全可观测和部分可观测环境之间的区别不同。一个已知的环境很可能是部分可观测的，例如，在纸牌游戏中，知道规则但仍然无法看到尚未翻转的牌。相反，一个未知环境可以是完全可观测的，如一个全新的电子游戏，屏幕可能会显示整个游戏状态，但在尝试之前并不知道各个按钮的作用。

如 2.2.1 节所述，性能度量本身可能是未知的，这可能是由于设计者不确定如何正确地描述，也可能是因为最终用户（其偏好很重要）是未知的。例如，出租车司机通常不知道新乘客是喜欢悠闲还是快速的旅程，是喜欢谨慎还是激进的驾驶风格。虚拟个人助理一开始对新主人的个人喜好一无所知。在这种情况下，智能体可以基于与设计者或用户的进一步交互来了解更多关于性能度量的信息。继而，这表明，任务环境必须被视为一个多智能体环境。

最困难的情况是部分可观测的、多智能体的、非确定性的、序贯的、动态的、连续的且未知的。驾驶出租车除了驾驶员的环境大多是已知的，在所有其他方面都很难。在一个陌生的国家驾驶租来的汽车，那里有不熟悉的地理环境、不同的交通法规以及焦虑的乘客，这令人更加紧张。

图 2-6 列出了许多熟悉环境的属性。注意，这些属性并不总是一成不变的。例如，因为将患者的患病过程作为智能体建模并不适合，所以我们将医疗诊断任务列为单智能体，但是医疗诊断系统还可能必须应对顽固的病人和多疑的工作人员，因此环境还具有多智能体的方面。此外，如果我们将任务设想为根据症状列表进行诊断，那么医疗诊断是回合式的；如果任务包括提出一系列测试、评估治疗过程中的进展、处理多个患者等，那么则是序贯的。

任务环境	可观测	智能体	确定性	回合式	静态	离散
填字游戏	完全	单	确定性	序贯	静态	离散
限时国际象棋	完全	多	确定性	序贯	半动态	离散
扑克	部分	多	非确定性	序贯	静态	离散
西洋双陆棋	完全	多	非确定性	序贯	静态	离散
驾驶出租车	部分	多	非确定性	序贯	动态	连续
医疗诊断	部分	单	非确定性	序贯	动态	连续
图片分析	完全	单	确定性	回合式	半动态	连续
零件选取机器人	部分	单	非确定性	回合式	动态	连续
提炼厂控制器	部分	单	非确定性	序贯	动态	连续
交互英语教师	部分	多	非确定性	序贯	动态	离散

图 2-6 任务环境的例子及其特征

因为如前所述，“已知的/未知的”不是严格意义上的环境属性，所以图 2-6 中没有包含此列。对于某些环境，例如国际象棋和扑克，很容易为智能体提供完整的规则知识，但考虑智能体如何在没有这些知识的情况下学会玩这些游戏仍然是有趣的。

与本书相关的代码库包括多个环境实现以及用于评估智能体性能的通用环境模拟器。实验通常不是针对单个环境进行的，而是针对从**环境类**（environment class）中抽象的许多环境进

行的。例如，要在模拟交通中评估出租车司机，我们需要运行具有不同的交通状况、照明和天气条件的多次模拟。我们关注智能体在环境类上的平均性能。

2.4 智能体的结构

到目前为止，我们通过描述行为（即在任意给定的感知序列之后执行的动作）讨论了智能体。现在我们必须迎难而上来讨论智能体内部是如何工作的。人工智能的工作是设计一个**智能体程序**（agent program）实现智能体函数，即从感知到动作的映射。假设该程序将运行在某种具有物理传感器和执行器的计算设备上，我们称之为**智能体架构**（agent architecture）：

$$\text{智能体} = \text{架构} + \text{程序}$$

显然，我们选择的程序必须是适合相应架构的程序。如果程序打算推荐步行这样的动作，那么对应的架构最好有腿。架构可能只是一台普通 PC，也可能是一辆带有多台车载计算机、摄像头和其他传感器的机器人汽车。通常，架构使程序可以使用来自传感器的感知，然后运行程序，并将程序生成的动作选择反馈给执行器。尽管本书第 25 章和第 26 章涉及传感器和执行器，但其余大部分内容都是关于设计智能体程序的。

2.4.1 智能体程序

我们在本书中设计的智能体程序都有相同的框架：它们将当前感知作为传感器的输入，并将动作返回给执行器。^① 注意智能体程序（将当前感知作为输入）和智能体函数（可能依赖整个感知历史）之间的差异。因为环境中没有其他可用信息，所以智能体程序别无选择，只能将当前感知作为输入。如果智能体的动作需要依赖于整个感知序列，那么智能体必须记住历史感知。

我们用附录 B 中定义的简单伪代码语言描述智能体程序。（在线代码库包含真实编程语言的实现。）图 2-7 显示了一个相当简单的智能体程序，它记录感知序列，然后使用它来索引动作表，以决定要执行的动作。动作表（如图 2-3 中给出的真空吸尘器世界示例）明确表示了智能体程序所体现的智能体函数。作为设计者，为了以这种方式构建理性智能体，我们必须构造一个表，该表包含每个可能的感知序列所对应的适当动作。

```

function TABLE-DRIVEN-AGENT(percept) returns 一个动作
    persistent: percepts, 初始为空的序列
               table, 以感知序列为索引的动作表, 初始为完全确定

    将 percept 添加到 percepts 的末尾
    action ← LOOKUP(percepts, table)
    return action

```

图 2-7 每个新感知都会调用 TABLE-DRIVEN-AGENT 程序，并且每次返回一个动作。它在内存中保留了完整的感知序列

表驱动的智能体构建方法注定失败，深入思考这一问题会很有启发性。设 $\mathcal{P}$ 为可能的感知集， T 为智能体的生存期（对应它将接收的感知总数），查找表将包含 $\sum_{t=1}^T |\mathcal{P}|^t$ 条记录。考虑自动驾驶出租车：来自单个摄像头（通常是 8 个摄像头）的视觉输入速度约为 70 MB/s（每秒

① 智能体程序框架还有其他选择。例如，我们可以让智能体程序作为与环境异步运行的协程。每个这样的协程都有一个输入和输出端口，并由一个循环组成，该循环读取输入端口的感知，并将动作写到输出端口。

30 帧, 每帧 $1080 \text{ 像素} \times 720 \text{ 像素}$, 每个像素包含 24 位颜色信息), 驾驶 1 小时后, 将会生成一张超过 $10^{600\,000\,000\,000}$ 条记录的表。即使是作为真实世界中微小的、表现良好的片段的国际象棋, 其查找表也至少有 10^{150} 条记录。相比之下, 可观测宇宙中的原子数量少于 10^{80} 个。这些表的巨大规模意味着: (a) 这个宇宙中没有任何物理智能体有空间存储表; (b) 设计者没有时间创建表; (c) 任何智能体都无法从其经验中学习所有正确的记录。

尽管如此, 假设表填充正确, TABLE-DRIVEN-AGENT 确实做了我们想要做的事情: 它实现了所需的智能体函数。

人工智能面临的关键挑战是找出编写程序的方法, 尽可能从一个小程序而不是从一个大表中产生理性行为。

历史上有许多例子表明, 在其他领域可以成功地做到这一点: 例如, 20 世纪 70 年代以前, 工程师和学生使用的巨大平方根表格, 现在已经被电子计算器上运行的仅有 5 行代码的牛顿方法所取代。现在问题是, 人工智能能像牛顿处理平方根那样处理一般智能行为吗? 我们相信答案是肯定的。

在本节剩余部分中, 我们将概述 4 种基本的智能体程序, 它们体现了几乎所有智能系统的基本原理:

- 简单反射型智能体;
- 基于模型的反射型智能体;
- 基于目标的智能体;
- 基于效用的智能体。

每种智能体程序以特定的方式组合特定的组件来产生动作。2.4.6 节大致解释了如何将所有这些智能体转换为学习型智能体, 以提高其组件的性能, 从而产生更好的动作。2.4.7 节描述在智能体中表示组件本身的各种方式。这种多样性为这一领域和这本书本身提供了一个主要的组织原则。

2.4.2 简单反射型智能体

最简单的智能体是简单反射型智能体 (simple reflex agent)。这些智能体根据当前感知选择动作, 忽略感知历史的其余部分。例如, 真空吸尘器的智能体函数在图 2-3 所示, 是一种简单反射型智能体, 因为它的决策仅基于当前位置以及该位置是否有灰尘。该智能体的智能体程序如图 2-8 所示。

```
function REFLEX-VACUUM-AGENT([location,status]) returns 一个动作
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

图 2-8 在只有两个位置的真空吸尘器环境中, 简单反射型智能体的智能体程序, 该程序实现图 2-3 中列出的智能体函数

注意, 与之前对应的表相比, 真空吸尘器的程序确实非常轻量。最明显的简化来自忽略感知历史, 这将相关感知序列的数量从 4^T 减少到 4。进一步的简化基于以下事实: 动作不依赖于位置, 只依赖于当前方格是否有灰尘。虽然我们已经使用 if-then-else 语句来编写智能体程序, 但它非常简单, 可以将其实现为布尔电路。

即使在更复杂的环境中, 也会出现简单的反射行为。想象自己是自动驾驶出租车司机。如果前面的汽车刹车并且刹车灯亮起, 那么你应该注意到这一点并开始刹车。换句话说, 你通

过对视觉输入进行一些处理来建立我们称之为“前面的汽车正在刹车”的条件。然后，这会触发智能体程序中的既定联结，对应动作“启动刹车”。我们称这样的联结为**条件-动作规则**（condition-action rule）^①，写作：

如果前面的车正在刹车，则启动刹车。

人类也有许多这样的联结，其中一些是习得反应（如驾驶），而另一些则是先天反射（如有东西接近眼睛时眨眼）。在本书中，我们展示了学习和实现这种联结的几种不同方式。

图 2-8 所示的程序限定于一个特定的真空吸尘器环境。一种更通用、更灵活的方法是，首先为条件操作规则构建通用解释器，然后为特定任务环境创建规则集。图 2-9 给出了通用程序的结构示意图，展示条件-动作规则如何在智能体中建立从感知到动作的联结。如果这看起来普通，不要担心，很快就会变得更加有趣。

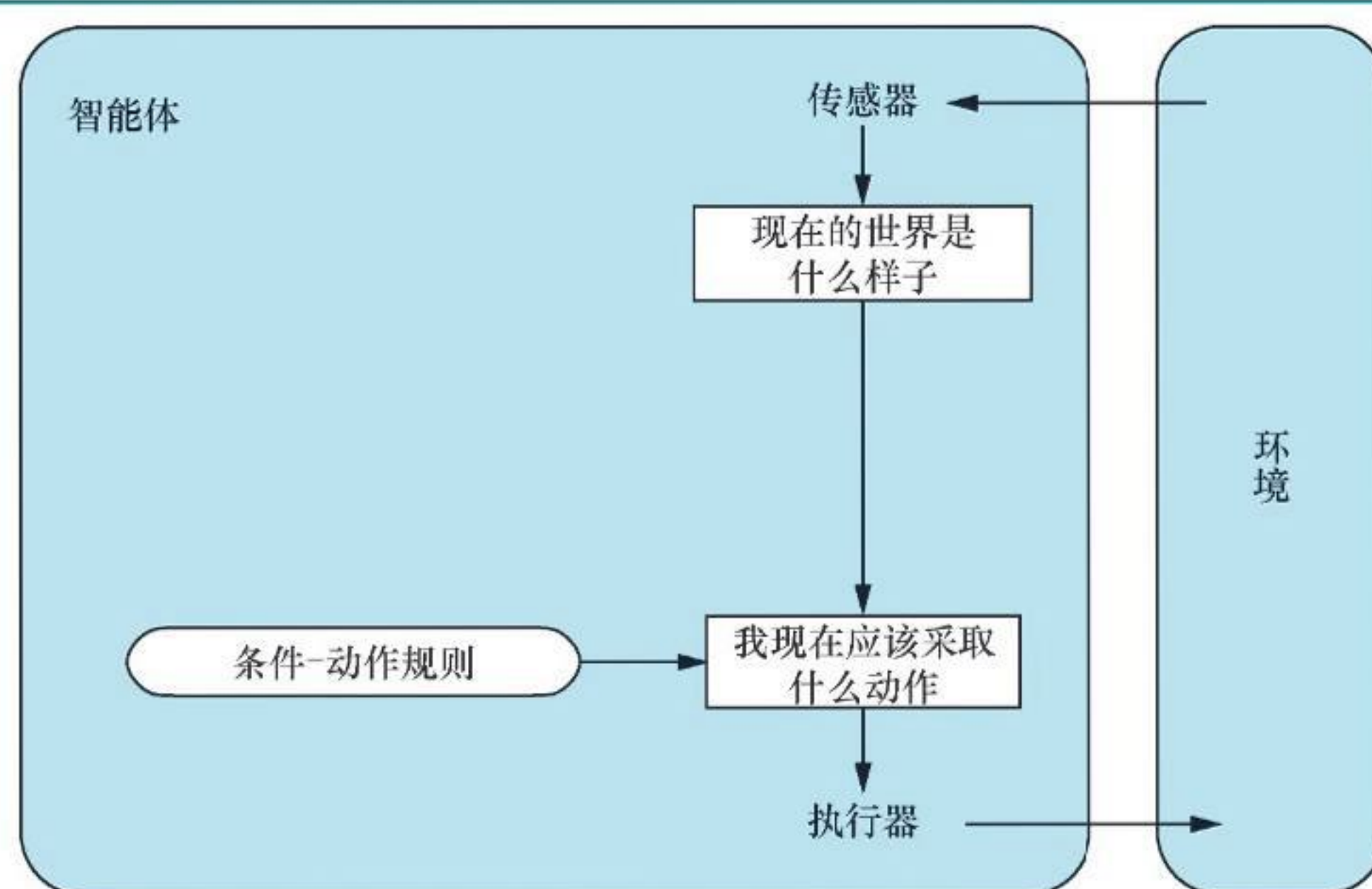


图 2-9 简单反射型智能体的示意图。我们使用矩形表示智能体决策过程的当前内部状态，使用椭圆表示过程中使用的背景信息

图 2-9 中智能体对应的智能体程序如图 2-10 所示。INTERPRET-INPUT 函数根据 *percept* 生成当前状态的抽象描述。给定状态描述，RULE-MATCH 函数返回规则集中匹配的第一条规则。注意，关于“规则”和“匹配”的描述纯粹是概念性的。如上所述，实际实现可以像实现布尔电路的逻辑门集合一样简单。或者，也可以使用“神经”电路，其中逻辑门由人工神经网络中的非线性单元代替（见第 21 章）。

```
function SIMPLE-REFLEX-AGENT(percept) returns 一个动作
  persistent: rules, 一组条件-动作规则

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

图 2-10 简单反射型智能体。它根据一条规则进行操作，该规则的条件与感知定义的当前状态相匹配

简单反射型智能体具有值得赞扬的简单特性，但它们的智能有限。图 2-10 中的智能体只有在当前感知的基础上才能做出正确的决策，也就是说，只有在环境完全可观测的情况下才可行。

① 也称为情境-动作规则、产生式系统或 if-then 规则。

即使是轻微的不可观测性也会造成严重的问题。例如，前面给出的刹车规则假设前车正在刹车的条件可以通过当前的感知（视频的单帧）确定。如果前车有一个安装在中间的（因此是唯一可识别的）刹车灯，这是可行的。但是，旧款车型的尾灯、刹车灯和转向灯的配置各不相同，而且从单幅图像中分辨出汽车是在刹车还是仅仅打开了尾灯不是总能做到的。一个简单反射型智能体在这样一辆车后面行驶，要么会连续不必要地刹车，或者更糟的是根本就不刹车。

我们在真空吸尘器世界中也可以看到类似的问题。假设一个简单的真空吸尘器反射型智能体没有位置传感器，只有一个灰尘传感器。这样的智能体只有两种可能的感知：*[Dirty]* 和 *[Clean]*。它可以用吸尘（*Suck*）来响应 *[Dirty]*，它该如何响应 *[Clean]* 呢？如果碰巧从方格 A 开始，向左（*Left*）移动会（永远）失败，如果从方格 B 开始，向右（*Right*）移动会（永远）失败。对在部分可观测环境中工作的简单反射型智能体而言，无限循环通常是不可避免的。

如果智能体可以**随机化**（randomize）其操作，则可以跳出无限循环。例如，如果真空吸尘器智能体感知到 *[Clean]*，它可能会通过抛硬币来选择左右。我们很容易就能证明智能体将平均通过两步到达另一个方格。如果方格是脏的，智能体将清理它，任务就会完成。因此，随机化的简单反射型智能体可能优于确定性的简单反射型智能体。

我们在 2.3 节中提到，在某些多智能体环境中，正确的随机行为是理性的。在单智能体环境中，随机化通常是不理性的。在某些情况下，这是一个有用的技巧，可以帮助简单反射型智能体，但在大多数情况下，我们可以使用更复杂的确定性智能体以做得更好。

2.4.3 基于模型的反射型智能体

处理部分可观测性的最有效方法是让智能体追踪它现在观测不到的部分世界。也就是说，智能体应该维护某种依赖于感知历史的**内部状态**（internal state），从而至少反映当前状态的一些未观测到的方面。对于刹车问题，内部状态范围不仅限于摄像头拍摄图像的前一帧，要让智能体能够检测车辆边缘的两个红灯何时同时亮起或熄灭。对于其他驾驶任务，如变道，如果智能体无法同时看到其他车辆，则需要追踪它们的位置。为了在任何时候都能驾驶，智能体需要追踪其钥匙的位置。

随着时间的推移，更新这些内部状态信息需要在智能体程序中以某种形式编码两种知识。首先，需要一些关于世界如何随时间变化的信息，这些信息大致可以分为两部分：智能体行为的影响和世界如何独立于智能体而发展。例如，当智能体顺时针转动方向盘时，汽车就会向右转；而下雨时，汽车的摄像头就会被淋湿。这种关于“世界如何运转”的知识（无论是在简单的布尔电路中还是在完整的科学理论中实现）被称为世界的**转移模型**（transition model）。

其次，我们需要一些关于世界状态如何反映在智能体感知中的信息。例如，当前面的汽车开始刹车时，前向摄像头的图像中会出现一个或多个亮起的红色区域；当摄像头被淋湿时，图像中会出现水滴状物体并部分遮挡道路。这种知识称为**传感器模型**（sensor model）。

转移模型和传感器模型结合在一起让智能体能够在传感器受限的情况下尽可能地跟踪世界的状态。使用此类模型的智能体称为**基于模型的智能体**（model-based agent）。

图 2-11 给出了基于模型的反射型智能体的结构，它具有内部状态，展示了当前感知如何与旧的内部状态相结合，并基于世界如何运转的模型生成当前状态的更新描述。智能体程序如图 2-12 所示。有趣的部分是函数 UPDATE-STATE，它负责创建新的内部状态描述。模型和状态的表示方式的细节因环境类型和智能体设计中使用的特定技术而异。

无论使用哪种表示，智能体几乎不可能准确地确定部分可观测环境的当前状态。相反，标有“现在的世界是什么样子”（图 2-11）的框表示智能体的“最佳猜测”（或者在具有多种可能

性的情况下的最佳猜测)。例如，一辆自动驾驶出租车可能无法看到停在它前面的大卡车周围的情况，只能猜测是什么导致了拥堵。因此，关于当前状态的不确定性可能是不可避免的，但智能体仍然需要做出决定。

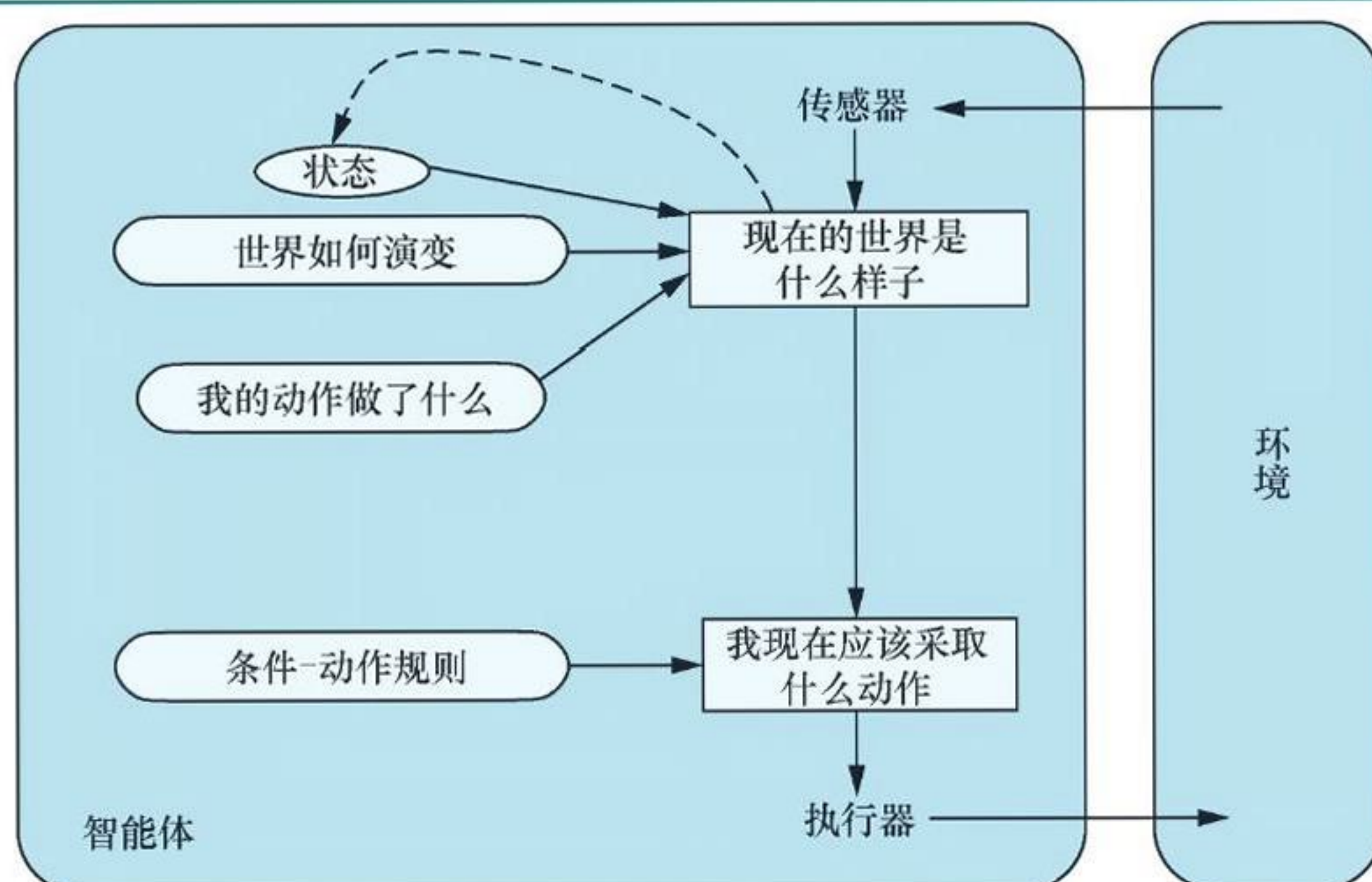


图 2-11 基于模型的智能体

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** 一个动作

persistent: *state*, 智能体对世界状态的当前理解

transition_model, 关于下一个状态如何决定于当前状态和动作的描述

sensor_model, 关于当前世界状态如何反映到智能体感知的描述

rules, 一组条件-动作规则

action, 最近的动作，初始为空

state ← UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)

rule ← RULE-MATCH(*state*, *rules*)

action ← *rule*.ACTION

return *action*

图 2-12 基于模型的反射型智能体。它使用内部模型追踪世界的当前状态，然后以与反射型智能体相同的方式选择动作

2.4.4 基于目标的智能体

了解环境的现状并不总是足以决定做什么。例如，在一个路口，出租车可以左转、右转或直行。正确的决定取决于出租车要去哪里。换句话说，除了当前状态的描述之外，智能体还需要某种描述理想情况的**目标**信息，例如设定特定的目的地。智能体程序可以将其与模型（与基于模型的反射型智能体中使用的信息相同）相结合，并选择实现目标的动作。图 2-13 展示了基于目标的智能体结构。

有时，基于目标的动作选择很直接，例如，单个动作能够立刻实现目标的情况。有时会更棘手，例如，智能体为了找到实现目标的方法而不得不考虑很长的复杂序列。**搜索**（第 3 ~ 5 章）和**规划**（第 11 章）是人工智能的子领域，专门用于寻找实现智能体目标的动作序列。

注意，这类决策从根本上不同于前面描述的条件-动作规则，因为它涉及对未来的考虑，包括“如果我这样做会发生什么？”和“这会让我快乐吗？”在反射型智能体设计中，这种信

息并没有被明确地表示出来，因为内置规则直接从感知映射到动作。反射型智能体在看到刹车灯时刹车，但它不知道为什么。基于目标的智能体在看到刹车灯时会刹车，因为这是它预测的唯一动作，这个动作可以实现不撞到其他汽车的目标。

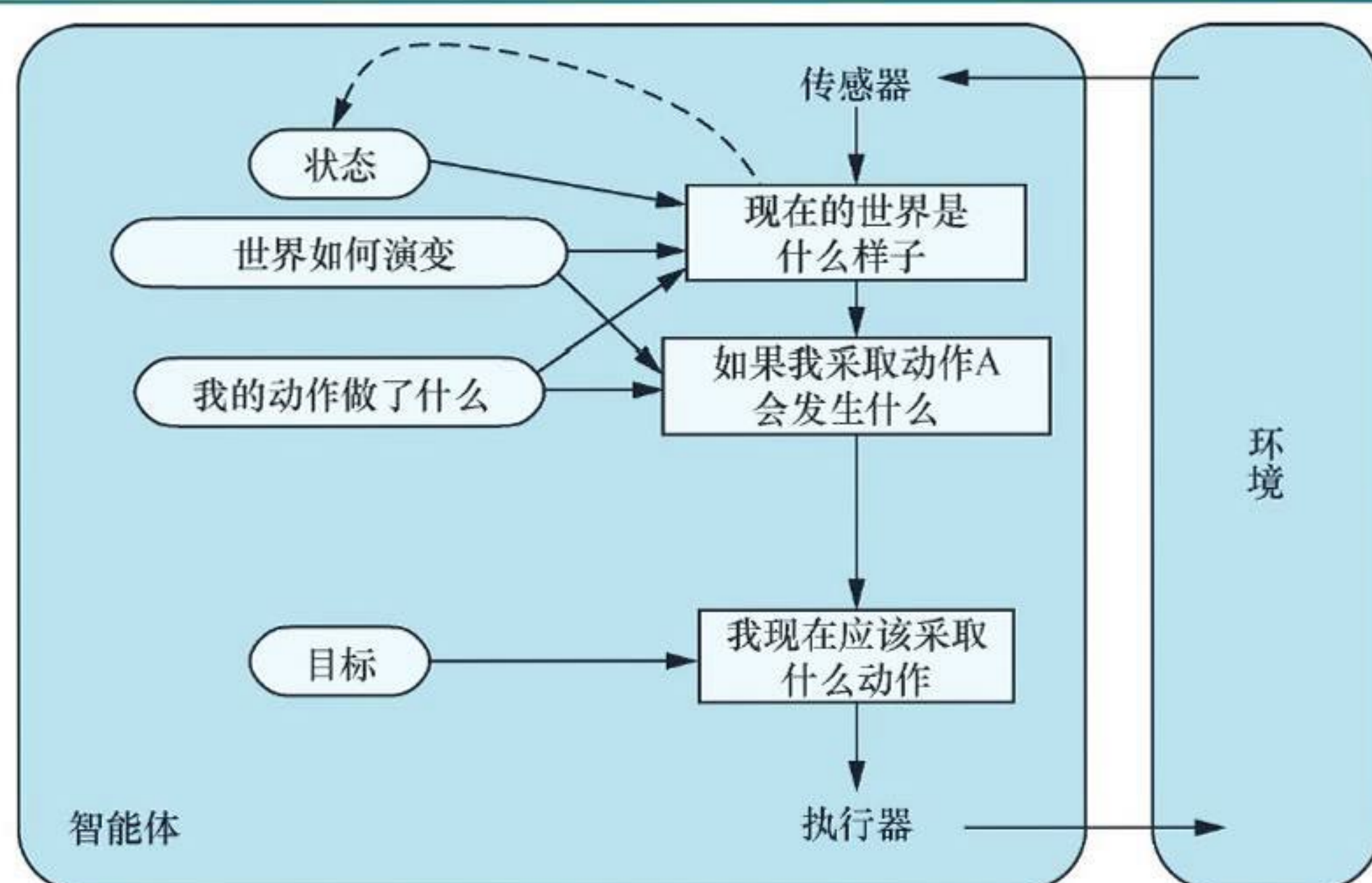


图 2-13 基于模型、基于目标的智能体。它追踪世界状态以及它试图实现的一系列目标，并选择一项最终能够实现目标的动作

尽管基于目标的智能体看起来效率较低，但它更灵活，因为支持其决策的知识是显式表示的，并且可以修改。例如，只要将目的地指定为目标，就可以很容易地更改基于目标的智能体的行为，以到达不同的目的地。反射型智能体关于何时转弯和何时直行的规则只适用于单一目的地，这些规则必须全部更换才能去新的目的地。

2.4.5 基于效用的智能体

在大多数环境中，仅靠目标并不足以产生高质量的行为。例如，许多动作序列都能使出租车到达目的地（从而实现目标），但有些动作序列比其他动作序列更快、更安全、更可靠或更便宜。目标只是在“快乐”和“不快乐”状态之间提供了一个粗略的二元区别。更一般的性能度量应该允许根据不同世界状态的“快乐”程度对智能体进行比较。经济学家和计算机科学家通常用**效用**（utility）这个词来代替“快乐”，因为“快乐”听起来不是很科学。^①

我们已经看到，性能度量会给任何给定的环境状态序列打分，因此它可以很容易地区分到达出租车目的地所采取的更可取和更不可取的方式。智能体的**效用函数**（utility function）本质上是性能度量的内部化。如果内部效用函数和外部性能度量一致，那么根据外部性能度量选择动作，以使其效用最大化的智能体是理性的。

再次强调，这不是理性的唯一实现方式，我们已经看到了一个适用于真空吸尘器世界的理性智能体程序（图 2-8），但并不知道它的效用函数是什么。与基于目标的智能体一样，基于效用的智能体在灵活性和学习方面有很多优势。此外，在两种情况下，仅靠目标是不充分的，但基于效用的智能体仍然可以做出理性的决策。首先，当存在相互冲突的目标时，只能实现其中的一部分（例如速度和安全），效用函数会进行适当的权衡。其次，当智能体有多个目标实现，但没有一个目标可以确定地实现时，效用提供了一种方法，可以权衡目标的重要性和成功的可能性。

^① 这里的“utility”一词指的是“实用的品质”，而不是电力公司或自来水厂等公共设施。

部分可观测性和非确定性在真实世界中普遍存在，因此，不确定性下的决策也普遍存在。从技术上讲，基于效用的理性智能体会选择能够最大化其动作结果**期望效用**（expected utility）的动作，也就是在给定每个结果的概率和效用的情况下，智能体期望得到的平均效用（附录 A 更精确地定义了期望）。在第 16 章中，我们证明，任何理性智能体的行为都必须表现得好像拥有一个效用函数，并试图最大化其期望值。具有显式效用函数的智能体可以使用通用算法做出理性决策，该算法不依赖于特定效用函数的最大化。通过这种方式，理性的“全局”定义（将那些具有最高性能的智能体函数指定为理性）变成了对理性智能体设计的“局部”约束，并可以通过一个简单的程序来表示。

基于效用的智能体结构如图 2-14 所示。基于效用的智能体程序见第 16 章和第 17 章，其中设计了决策型智能体，必须处理非确定性或部分可观测环境中固有的不确定性。如第 18 章所述，多智能体环境中的决策也在效用理论的框架下进行了研究。

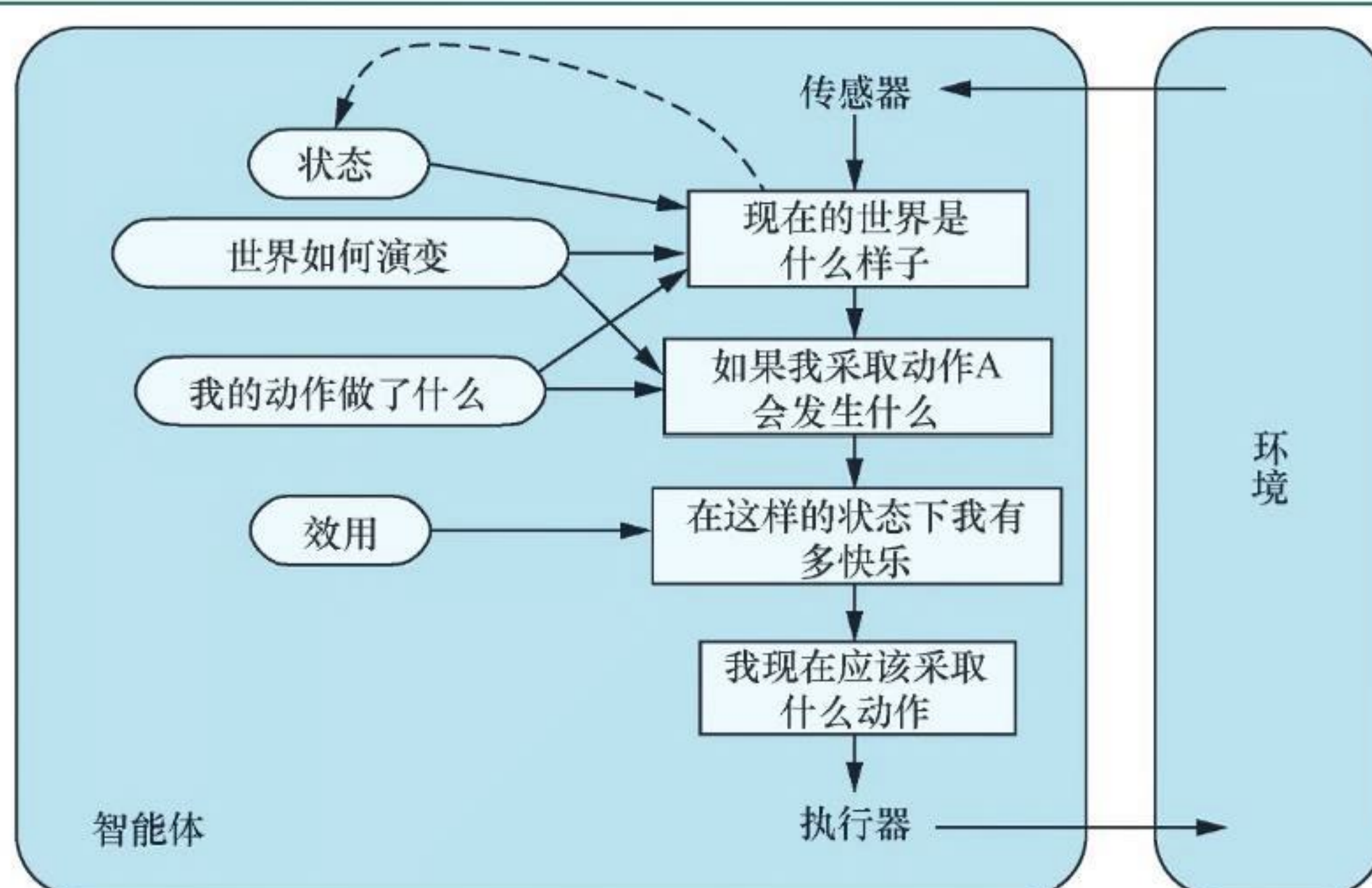


图 2-14 基于模型、基于效用的智能体。它使用了一个世界模型以及一个效用函数来衡量它在各状态之间的偏好，然后选择产生最佳期望效用的动作，其中期望效用是通过对所有可能的结果状态和对应概率加权所得

说到这里，读者可能会想，“这么简单吗？只需要构建能够最大化期望效用的智能体，我们就完成了？”这类智能体确实是智能的，但这并不简单。基于效用的智能体必须对其环境进行建模和跟踪，这些任务涉及大量关于感知、表示、推理和学习的研究。这些研究结果填满了本书的许多章节。选择效用最大化的行动方案也是一项艰巨的任务，需要更多的章节描述精巧的算法。即使使用这些算法，由于计算复杂性，完美理性在实践中通常是无法实现的（正如我们在第 1 章中所指出的）。我们还应该注意到，并非所有基于效用的智能体都是基于模型的。我们将在第 22 章和第 26 章中看到，**无模型的智能体**（model-free agent）可以学习在特定情况下什么样的动作是最好的，而不必确切地了解该动作如何改变环境。

最后，所有这些都假设设计者能够正确地指定效用函数，第 17 章、第 18 章和第 22 章将更深入讨论未知效用函数的问题。

2.4.6 学习型智能体

我们已经描述了一些智能体程序和选择动作的方法。到目前为止，我们还没有解释智能体程序是如何产生的。在图灵（Turing, 1950）早期的著名论文中，他考虑手动编程实现智能机器的想法。他估计了这可能需要多少工作量，并得出结论，“似乎需要一些更快捷的方法”。他提出的方法是构造学习型机器，然后教它们。在人工智能的许多领域，这是目前创建最先进系统的首选方法。任

何类型的智能体（基于模型、基于目标、基于效用等）都可以构建（或不构建）成学习型智能体。

正如我们之前提到的，学习还有另一个优势：它让智能体能够在最初未知的环境中运作，并变得比其最初的知识可能允许的能力更强。在本节中，我们简要介绍学习型智能体的主要思想。在整本书中，我们对特定类型智能体中的学习因素和方法的评论贯穿全书。第19～22章将更加深入地介绍学习算法本身。

学习型智能体可分为4个概念组件，如图2-15所示。最重要的区别在于负责提升的**学习元素**（learning element）和负责选择外部行动的**性能元素**（performance element）。性能元素是我们之前认为的整个智能体：它接受感知并决定动作。学习元素使用来自**评估者**（critic）对智能体表现的反馈，并以此确定应该如何修改性能元素以在未来做得更好。

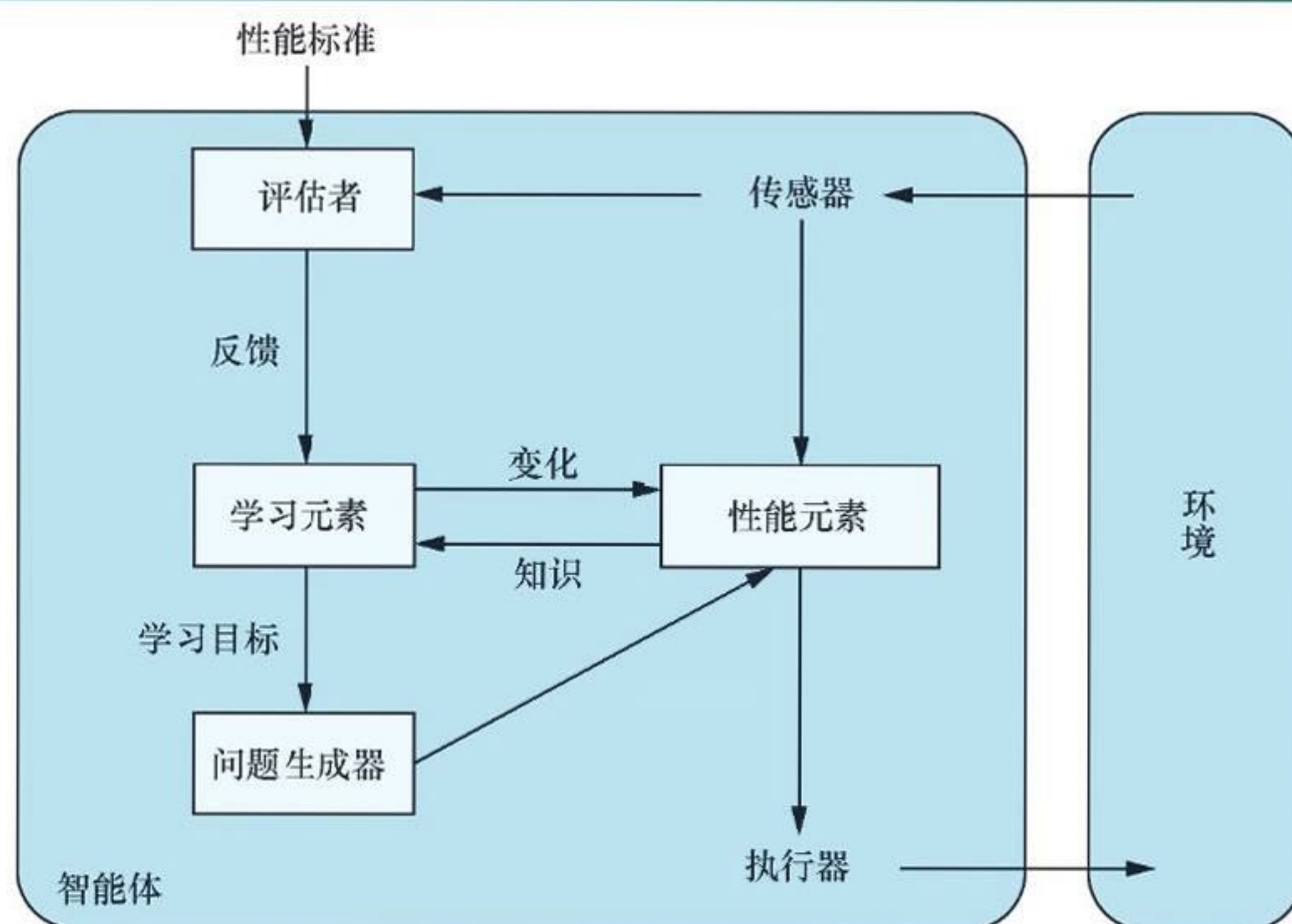


图 2-15 通用学习型智能体。“性能元素”框表示我们之前认为的整个智能体程序，现在“学习元素”框可以修改该程序以提升其性能

学习元素的设计在很大程度上取决于性能元素的设计。当设计者试图设计一个学习某种能力的智能体时，第一个问题不是“我要如何让它学习这个？”而是“一旦智能体学会了如何做，它将使用什么样的性能元素？”给定性能元素的设计，可以构造学习机制来改进智能体的每个部分。

评估者告诉学习元素：智能体在固定性能标准方面的表现如何。评估者是必要的，因为感知本身并不会指示智能体是否成功。例如，国际象棋程序可能会收到一个感知，提示它已将死对手，但它需要一个性能标准来知道这是一件好事；感知本身并没有这么说。确定性能标准很重要。从概念上讲，应该把它看作完全在智能体之外，因为智能体不能修改性能标准以适应自己的行为。

学习型智能体的最后一个组件是**问题生成器**（problem generator）。它负责建议动作，这些动作将获得全新和信息丰富的经验。如果性能元素完全根据自己的方式，它会继续选择已知最好的动作。但如果智能体愿意进行一些探索，并在短期内做一些可能不太理想的动作，那么从长远来看，它可能会发现更好的动作。问题生成器的工作是建议这些探索性行动。这就是科学家在进行实验时所做的。伽利略并不认为从比萨斜塔顶端扔石头本身有价值。他并不是想要打碎石头或改造不幸行人的大脑。他的目的是通过确定更好的物体运动理论来改造自己的大脑。

学习元素可以对智能体图（图2-9、图2-11、图2-13和图2-14）中显示的任何“知识”组件进行更改。最简单的情况是直接从感知序列学习。观察成对相继的环境状态可以让智能体了解“我的动作做了什么”以及“世界如何演变”以响应其动作。例如，如果自动驾驶出租车在

湿滑路面上行驶时进行一定程度的刹车，那么它很快就会发现实际减速多少，以及它是否滑出路面。问题生成器可能会识别出模型中需要改进的某些部分，并建议进行实验，例如在不同条件下的不同路面上尝试刹车。

无论外部性能标准如何，改进基于模型的智能体的组件，使其更好地符合现实几乎总是一个好主意。（从计算的角度来看，在某些情况下简单但稍微不准确的模型比完美但极其复杂的模型更好。）当智能体试图学习反射组件或效用函数时，需要外部标准的信息。

例如，假设出租车司机因为乘客在旅途中感到非常不适，没有收到小费。外部性能标准必须告知智能体，小费的损失对其整体性能有负面影响；然后，该智能体可能会了解到暴力操作有损其自身的效用。从某种意义上说，性能标准将传入感知的一部分区分为**奖励**（reward）或**惩罚**（penalty），以提供对智能体行为质量的直接反馈。动物的疼痛和饥饿等固有的性能标准可以通过这种方式理解。

更一般地说，人类的选择可以提供有关人类偏好的信息。例如，假设出租车不知道人们通常不喜欢噪声，于是决定不停地按喇叭以确保行人知道它即将到来。随之而来的人类行为，如盖住耳朵、说脏话甚至可能剪断喇叭上的电线，将为智能体提供更新其效用函数的证据。这个问题将在第 22 章进一步讨论。

总之，智能体有各种各样的组件，这些组件可以在智能体程序中以多种方式表示，因此学习方法之间似乎存在很大差异。然而，主题仍然是统一的：智能体中的学习可以概括为对智能体的各个组件进行修改的过程，使各组件与可用的反馈信息更接近，从而提升智能体的整体性能。

2.4.7 智能体程序的组件如何工作

我们已经将智能体程序（用非常高级的术语）描述为由各种组件组成，其功能是回答诸如“现在的世界是什么样的？”“我现在应该采取什么动作？”“我的动作将导致什么？”等问题。人工智能学生的下一个问题是，“这些组件究竟是如何工作的？”要正确回答这个问题大约需要一千页的篇幅，但在这里我们希望读者能够注意一些基本区别，即组件表示智能体所处环境的各种方式之间的区别。

粗略地说，我们可以通过一个复杂性和表达能力不断增加的横轴来描述表示，即原子表示、因子化表示和结构化表示。为了辅助说明这些观点，我们可以考虑特定的智能体组件，例如处理“我的动作会导致什么”。这个组件描述了作为采取动作的结果可能在环境中引起的变化，图 2-16 展示了如何表示这些转移的示意图。

在**原子表示**（atomic representation）中，世界的每一个状态都是不可分割的，它没有内部结构。考虑这样一个任务：通过城市序列找到一条从某个国家的一端到另一端的行车路线（我们在图 3-1 中会解决这个问题）。为了解决这个问题，将世界状态简化为所处城市的名称就足够了，这就是单一的知识原子，也是一个“黑盒”，它唯一可分辨的属性是与另一个黑盒相同或不同。搜索和博弈中的标准算法（第 3 ~ 5 章）、隐马尔可夫模型（第 14 章）以及马尔可夫决策过程（第 17 章）都基于原子表示。

因子化表示（factored representation）将每个状态拆分为一组固定的**变量或属性**，每个变量或属性都可以有一个**值**。考虑同一驾驶问题更真实的描述，即我们需要关注的不仅仅是一个城市或另一个城市的原子位置，可能还需要关注油箱中的汽油量、当前的 GPS 坐标、油量警示灯是否工作、通行费、收音机上的电台等。两个不同的原子状态没有任何共同点（只是不同的黑盒），但两个不同的因子化状态可以共享某些属性（如位于某个特定的 GPS 位置），而其他属性不同（如有大量汽油或没有汽油），这使得研究如何将一种状态转换为另一种状态变得更加容易。人工智能的许多重要领域都基于因子化表示，包括约束满足算法（第 6 章）、命题逻辑

(第7章)、规划(第11章)、贝叶斯网络(第12~16章)以及各种机器学习算法。

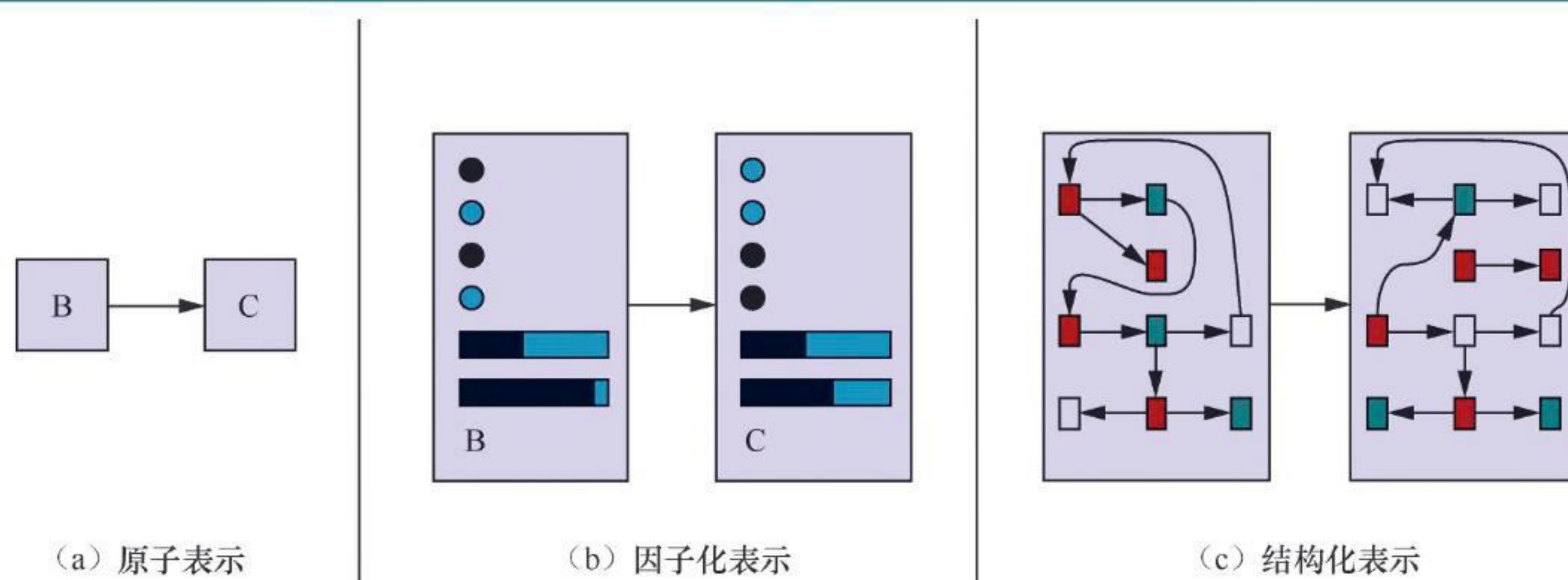


图 2-16 表示状态及其之间转移的 3 种方法：(a) 原子表示一个状态（如 B 或 C）是没有内部结构的黑盒；(b) 因子化表示状态由属性值向量组成，值可以是布尔值、实值或一组固定符号中的一个；(c) 结构化表示状态包括对象，每个对象可能有自己的属性以及与其他对象的关系

出于许多目的，我们需要将世界理解为存在着相互关联的事物，而不仅仅是具有值的变量。例如，我们可能会注意到前面有一辆卡车正在倒车进入一个奶牛场的车道，但一头散养的奶牛挡住了卡车的路。因子化表示不太可能为属性 *TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow* 预先配备 true 或 false 的值。这就需要有一个**结构化表示**（structured representation），在这种表示中可以明确地描述诸如奶牛和卡车之类的对象及其各种不同的关系（见图 2-16c）。结构化表示是关系数据库和一阶逻辑（第 8~10 章）、一阶概率模型（第 15 章）和大部分自然语言理解（第 23 章和第 24 章）的基础。事实上，人类用自然语言表达的大部分内容都与对象及其关系有关。

如前所述，原子表示、因子化表示和结构化表示所在的轴是**表达性**（expressiveness）增强的轴。粗略地说，可以通过简洁的描述捕捉到更具表达性的表示，表达性差的表示也可以捕捉到一切，但需要更多描述。通常，表达性更强的语言更简洁；例如，国际象棋规则可以用一两页结构化表示语言（如一阶逻辑）来描述，但需要数千页因子化表示语言（如命题逻辑）来描述，而需要 10^{38} 页的原子语言（如有限状态自动机）来描述。但是，随着表示能力的增强，推理和学习变得更加复杂。为了在避免缺点的同时获得表达性表示的好处，真实世界中的智能系统可能需要轴上的所有点同时运行。

另一个表示轴涉及从概念到物理记忆中位置的映射，包括计算机的内存和大脑的记忆。如果概念和记忆位置之间存在一对一的映射，我们称之为**局部表示**（localist representation）。但是，如果一个概念表示分布在多个记忆位置，并且每个记忆位置被用作多个不同概念表示的一部分，我们称之为**分布式表示**（distributed representation）。分布式表示对噪声和信息丢失更健壮。使用局部表示，从概念到记忆位置的映射是随机的，如果传输错误而导致几位乱码，我们可能会将卡车（Truck）与无关的概念停战（Truce）混淆。但在分布式表示中，可以把每个概念想象成多维空间中的一个点，即使有一些乱码，也会移动到该空间中附近的点，其具有相似的含义。

小结

本章是人工智能的旋风之旅，在这个过程中我们认为人工智能是智能体设计的科学。本章要回顾的要点如下。

- **智能体**是在环境中感知和行动的事物。智能体的**智能体函数**指定智能体在响应任意感知序列时所采取的动作。
- **性能度量**评估智能体在环境中的行为。给定到目前为止所看到的感知序列，**理性智能体**的动作是为了最大化性能度量的期望值。
- **任务环境规范**包括性能度量、外部环境、执行器和传感器。在设计智能体时，第一步必须始终是尽可能完整地指定任务环境。
- 任务环境在几个重要维度上有所不同。它们可以是完全可观测的或部分可观测的、单智能体的或多智能体的、确定性的或非确定性的、回合式的或序贯的、静态的或动态的、离散的或连续的、已知的或未知的。
- 在性能度量未知或难以正确指定的情况下，智能体优化错误目标的风险很大。在这种情况下，智能体设计应该反映真实目标的不确定性。
- **智能体程序**实现智能体函数。存在各种基本的智能体编程，反映了决策过程中明确使用的信息类型。这些设计在效率、紧凑性和灵活性方面各不相同。智能体程序的适当设计取决于环境的性质。
- **简单反射型智能体**直接响应感知，而**基于模型的反射型智能体**保持内部状态以跟踪当前感知中不明晰的世界状态。**基于目标的智能体**采取行动来实现目标，而**基于效用的智能体**试图最大化自己期望的“快乐”。
- 所有智能体都可以通过**学习**提升性能。

参考文献与历史注释

动作在智能中的核心作用（实践推理的概念）至少可以追溯到亚里士多德的《尼各马可伦理学》（*Nicomachean Ethics*）。实践推理也是麦卡锡颇具影响力的论文“Programs with Common Sense”（McCarthy, 1958）的主题。机器人和控制理论领域本质上主要与物理主体有关。控制理论中的**控制器**概念与人工智能中的**智能体**概念相同。也许令人惊讶的是，人工智能在其历史上的大部分时间都集中在问答系统、定理证明器、视觉系统等孤立组件上，而不是完整智能体。吉内塞雷斯和尼尔森写的教科书（Genesereth and Nilsson, 1987）对智能体的讨论是一个有影响的例外。完整智能体的观点现在被广泛接受，并且是最近教科书（Padgham and Winikoff, 2004; Jones, 2007; Poole and Mackworth, 2017）的中心主题。

第1章追溯了理性概念在哲学和经济学中的根源。直到20世纪80年代中期，这一概念才在人工智能领域引起人们的兴趣，那时它开始充斥在许多关于该领域正确技术基础的讨论中。乔恩·多伊尔（Jon Doyle）的一篇论文（Doyle, 1983）预测称，理性智能体的设计将被视为人工智能的核心任务，而其他热门主题将衍生形成新的学科。

传统控制理论对环境特性及其对理性智能体设计的影响关注仔细且明显，例如，经典控制系统（Dorf and Bishop, 2004; Kirk, 2004）处理完全可观测的、确定性环境，随机最优控制（Kumar and Varaiya, 1986; Bertsekas and Shreve, 2007）处理部分可观测的随机环境，混合控制（Henzinger and Sastry, 1998; Cassandras and Lygeros, 2006）处理包含离散和连续元素的环境。在运筹学领域（Puterman, 1994）发展起来的**动态规划**（dynamic programming）文献中，完全可观测环境和部分可观测环境之间的区别也是核心问题，我们将在第17章中对此进行讨论。

虽然简单的反射型智能体是行为主义心理学的核心（见第1章），但大多数人工智能研究人

员认为它们过于简单而无法提供太多影响。罗森舍因 (Rosenschein, 1985) 和布鲁克斯 (Brooks, 1986) 质疑了这个假设 (见第 26 章)。人们已经在寻找追踪复杂环境的有效算法方面做了大量工作 (Bar-Shalom *et al.*, 2001; Choset *et al.*, 2005; Simon, 2006), 其中大部分是在概率环境下所做的。

从亚里士多德的实践推理观点到麦卡锡关于逻辑人工智能的早期论文, 都以基于目标的智能体为前提。机器人 Shakey (Fikes and Nilsson, 1971; Nilsson, 1984) 是第一个基于目标的逻辑智能体的化身。吉内塞雷斯和尼尔森 (Genesereth and Nilsson, 1987) 对基于目标的智能体进行了全面的逻辑分析, 肖厄姆 (Shoham, 1993) 开发了一种称为面向智能体编程的基于目标的编程方法。基于智能体的方法现在在软件工程中非常流行 (Ciancarini and Wooldridge, 2001)。它还渗透到操作系统领域, 其中自主计算 (autonomic computing) 指的就是通过感知-行为的循环和机器学习方法来监控自身的计算机系统和网络 (Kephart and Chess, 2003)。注意, 一组旨在在真正的多智能体环境中协同工作的智能体程序必然表现出模块化, 即这些程序不共享内部状态, 只通过环境相互通信。在多智能体系统领域, 通常将单智能体的智能体编程为一个模块化的自主子智能体的集合。在某些情况下, 人们甚至可以证明, 由此产生的系统提供了与整体设计相同的最佳解决方案。

以目标为基础的智能体观点也主导了问题求解领域的认知心理学传统, 从影响巨大的 *Human Problem Solving* (Newell and Simon, 1972) 开始, 并贯穿于纽厄尔后期所有的工作 (Newell, 1990)。目标, 进一步分析为欲望 (广义) 和意图 (当前追求), 是迈克尔·布拉特曼 (Michael Bratman) 开发的具有影响力的智能体理论的核心 (Bratman, 1987)。

如第 1 章所述, 效用理论作为理性行为基础的发展可以追溯到数百年前。在人工智能的早期研究中避开效用而倾向目标, 但也有一些例外 (Feldman and Sproull, 1977)。20 世纪 80 年代对概率方法兴趣的复苏, 导致人们接受期望效用最大化作为决策的最一般框架 (Horvitz *et al.*, 1988)。珀尔的教科书 (Pearl, 1988) 是人工智能领域第一本深入介绍概率和效用理论的教科书, 它对不确定性下推理和决策的实用方法的阐述可能是 20 世纪 90 年代迅速转向基于效用的智能体的最大因素 (见第 16 章)。强化学习在决策理论框架内的形式化也促成了这一转变 (Sutton, 1988)。值得注意的是, 直到最近, 几乎所有的人工智能研究都假设性能度量可以通过效用函数或奖励函数的形式精确且正确地指定 (Hadfield-Menell *et al.*, 2017a; Russell, 2019)。

图 2-15 中所示的学习型智能体的一般设计是机器学习文献中的经典 (Buchanan *et al.*, 1978; Mitchell, 1997)。程序中体现的设计示例至少可以追溯到亚瑟·塞缪尔的 (Samuel, 1959, 1967) 中的学习型西洋跳棋程序。学习型智能体将在第 19 ~ 22 章中深入讨论。

(Huhns and Singh, 1998) 和 (Wooldridge and Rao, 1999) 中收集了一些关于基于智能体方法的早期论文。关于多智能体系统的教科书为智能体设计的诸多方面提供了很好的指引 (Weiss, 2000a; Wooldridge, 2009)。20 世纪 90 年代, 多个专门讨论智能体的系列会议开始举办, 包括智能体理论、架构和语言国际研讨会 (International Workshop on Agent Theories, Architectures, and Languages, ATAL), 国际自主智能体会议 (International Conference on Autonomous Agents, AGENTS), 以及国际多智能体系统联合会议 (International Joint Conference on Multi-Agent Systems, ICMAS)。2002 年, 这 3 个会议合并成了国际自主智能体及多智能体系统联合会议 (International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS)。从 2000 年到 2012 年, 每年都会举办面向智能体的软件工程 (Agent-Oriented Software Engineering, AOSE) 研讨会。期刊 *Autonomous Agents and Multi-Agent System* 创立于 1998 年。*Dung Beetle Ecology* (Hanski and Cambefort, 1991) 提供了大量有关粪甲虫行为的有趣信息。YouTube 上有关于它们活动的视频记录, 非常鼓舞人心。