1. 知识表示

知识是智能的基础，知识应用的难点在于知识推理，知识推理的难点在于知识表示。要让计算机，就得让它具有知识。然而计算机并不能很好的理解人类的知识，这就需要将人类的知识用适当的模式来表示，才能让计算机具备智能，因此知识表示是基于知识的人工智能应用中的核心部分。

本章将从以下方面介绍知识表示：首先是经典知识的概念，特性以及知识表示的概念。接着逐个介绍常见的知识表示方法，为后面的学习奠定基础。

**3.1知识与知识表示的概念**

**3.1.1 知识的概念**

知识是人类在长期的生活实践中以及科学研究实验中累积起来的，是对客观世界的认识及经验。把有关信息关联在一起所形成的信息结构称为知识。信息之间有多种关联形式，用得最多的一种是用“如果......，则......”表示的关联形式。在人工智能中，这种知识被称为“规则”，它反映了信息间的某种因果关系。例如人类知识中“燕子低飞蛇过道，大雨不久就来到”，转化为能让计算机理解的模式，可得到以下知识：如果燕子低飞蛇过道，则快要下大雨。

还有一种知识被称为“事实”，例如“老虎是动物”，这个知识表达了“老虎”与“动物”之间的关系。

**3.1.2知识的特性**

1. 相对正确性

知识是人类对客观世界认识的结果，并且受到长期实践的检验。因此我们可以认为，在一定的条件及环境下，知识是正确的。注意，这里的“一定的条件及环境”必不可少，这也是知识正确性的前提。例如“三角形内角和为180°”，这个知识在平面上就是正确的，但如果三角形放到一个曲面上，就是不正确的。

在人工智能中，知识的相对正确性尤为重要。在建造专家系统时，为了减少知识库的规模，通常将知识限制在所求解问题的范围内。通俗来说，就是知识对所求解问题时正确的就行。比如在一个动物识别系统中，仅仅识别老虎、猫、猫头鹰这三种动物时，仅靠着知识“如果该动物会飞，则该动物是猫头鹰”就是正确的。

1. 不确定性

在现实世界中，信息不一定是准确的，关联也不一定是确定的。这就使得知识并不总是

只有“真”与“假”两种状态，而是在“真：与”假“之间还存在许多中间状态。知识的这一特性称为不确定性。造成知识具有不确定性的原因是多方面的，主要有：

①由随机性引起的不确定性。

由随机事件形成的知识不能直接用“真”与“假”来刻画，因为它是不确定的。例如，“如果出现头疼和流鼻涕的症状，则有可能患了感冒”这条知识，其中的“有可能”反映的便是“头疼和流鼻涕”与“患了感冒”之间的一种不确定的因果关系

②由模糊性引起的不确定性。

由于有些事物本身的概念不清楚，或者事物之间存在着模糊关系，我们无法判断它们是否为“真”还是“假”，像这样由模糊概念、模糊关系所形成的知识显然就是不确定的。例如健康的人和不健康的人并没有明确的划分，健康是一个模糊的定义，我们要判断一个人是否健康，便是不确定的。

③由经验引起的不确定性。

知识大多数是专家学者在长期的实践及研究中累积起来的。就如同临床医学一样，在面对相同的病例多数情况下医生可以根据自己的临床经验来让患者治愈，但并不是每次都用相同的知识就能解决。经验本身就蕴含着不确定性及模糊性，这也形成了知识的不确定性。

④由不完全性引起的不确定性。

知识实际上有一个逐步完善的过程，当客观事物表露得不充分，人们对它的认识就不够全面。这样的出来的知识便是不确定的。例如，盲人摸象，象是确定的，但每个人遮住眼睛，无法直观地观察象，显然得到的知识便是不完全的，这也导致了盲人对象的不确定性。不完全性是使知识具有不确定的一个重要原因

1. 可表示性与可利用性

知识的可表示性是指知识可以通过适当形式表示，如我们可以用语言文字等来表示知识，这样知识才能被存储和传播。

知识的可利用性则更简单理解了，我们每个人每天都在利用知识解决生活中的各种问题，这便体现了知识的可利用性。

**3.1.3知识表示的概念**

知识表示(knowledge representation)就是将人类知识形式化或者模型化。知识表示是对知识的一种描述，或者说是一种规定，一种计算机可以接受的用于描述知识的数据结构。通过知识表示，计算机可以存储并运用人类知识，进而形成人工智能。

目前用得比较多的知识表示方法有一阶谓词逻辑表示法、产生式表示法、框架表示法、语义网络表示法、脚本表示法、状态空间表示法等。

同一个知识可以有多种表示方法，但不同表示方法效果却不一样。并不是每种方法对任何智能问题都合适，面对具体的问题用哪种知识表示方法更好，要因问题而异。

选择知识表示方法的原则有：(1)充分表示领域知识；(2)有利于对知识的利用；(3)便于对知识的组织、维护与管理；(4)便于理解与实现。

**3.2知识表示方法**

**3.2.1 逻辑表示法**

逻辑本身根据复杂性从简单到复杂分为：命题逻辑、一阶谓词逻辑、高阶逻辑。

首先先了解一下命题，命题即非真即假的陈述句，它定义了具有真假值的原子命题。若命题的意义为真，称它的真值为真，记为*T*。若命题的意义为假，称它的真值为假，记为*F*。除此之外，还可以通过与（∧）、或（∨）、非（┐）、蕴含（⇒）、当且仅当（）等逻辑连接符将多个原子命题组合成复合命题，这些逻辑连接符可以将一些原子命题组合成现实中的复杂知识。为了避免运算的歧义，命题逻辑还定义了不同的连接词和操作符的优先级关系，例如非（┐）具有最高优先级。逻辑连接符根据真值表运算组合命题的真假值,真值表如表3-1所示。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | Y | ┐X | X∧Y | X∨Y | X⇒Y | XY |
| T | T | F | T | T | T | T |
| F | T | T | F | T | T | F |
| T | F | F | F | T | F | F |
| F | F | T | F | F | T | T |

接下来介绍谓词逻辑，谓词逻辑可分为一阶谓词逻辑和高阶谓词逻辑。由于高阶谓词逻辑过于复杂，实践中应用很少，本书不做详细介绍。

一阶谓词逻辑在命题逻辑的基础上增加了量词的概念，即全称量词()和存在量词()。一阶逻辑的基本语法元素是表示对象、关系和函数的符号。其中对象对应常量符号，通常指一些事物的个体或类别，如老虎、苹果、叶子等。关系对应谓词符号，指的是一种映射，例如“朋友”便是一个谓词，对于对象“小明”，朋友（小明，x）是谓词对对象的操作，其中x可以是小明的一个或多个朋友，也可以为空。与谓词不同，函词是代表全函数的一种特殊的谓词形式，它要求每一个定义域的对象具有一个映射值，例如“国籍”就是一个函词，因为一般情况下每个人的国籍有且仅有一个。

相比于命题逻辑，谓词逻辑的优势可以表达对象集合的属性，而不用逐一列举所有对象，通过量词能够实现对对象集合的描述。

全称量词()表示集合的全部，通常读作“对于所有的......”，存在量词()表示集合中至少存在一个对象，通常读作“存在有......”，这一对象并不需要显式表示出来。下面用几个例子来进行说明。

例如所有一班的同学都是小明的朋友，如果仅用逻辑命题来表示，朋友（小明，Y），Y表示一班的所有同学的集合，这样表示过于复杂。那用全称谓词便可以很方便地表示出来：

x一班（x）⇒x是小明的朋友

这里的x称为变量，一般用小写字母表示。形式化地，对于任意的含有x的逻辑表达式P，语句xP表示对于每一个对象x，P为真。

同理，若我们用全称谓词来表示二班有一些同学是小明的朋友，表示如下：

x二班（x）⇒x是小明的朋友

形式化地，对于任意的含有x的逻辑表达式P，语句xP表示至少存在一个对象x，使P为真。

这两个量词之间也存在着相互转变的关系，主要依靠否定词来实现，它们遵循如下的德摩根定律：

 

 

 

 

命题逻辑和一阶谓词逻辑是人工智能领域使用最早的和最广泛的知识表示方式。在命题逻辑的基础上，引入量词和变量使其能够描述更抽象的知识，便于知识推理，而且使用谓词逻辑表示的知识容易转化成计算机的内部形式。逻辑表示能够保证知识表示的一致性，也能够确保推理结果的正确性。但是谓词表示无法表示不确定性的知识，而且当知识中的属性、谓词和命题数量增大时，以及盲目使用推理规则，可能形成组合爆炸问题，计算复杂性呈指数级增长态势，工作效率低下。

**3.2.2产生式表示法**

产生式通常用于表示事实、规则以及它们的不确定性度量，适合于表示事实性知识和规则性知识

1. 确定性规则的产生式表示

确定性规则的产生式表示的基本形式为：

IF *P* THEN *Q* 或者 *P*→*Q*

其中，*P*是前提，用于指出该产生式是否可用的条件；Q是一组结论或操作，用于指出当前提P所指出的条件被满足时，应该得出的结论或应该执行的操作。整个产生式的含义是：如果前提P被满足，则结论Q成立或执行Q所规定的操作。例如：

*r1* :IF 动物会飞 AND 会下蛋 THEN 该动物是鸟

其中，*r1*是该产生式的编号；“动物会飞AND会下蛋”是前提P；“该动物是鸟”是结论Q。

1. 不确定性规则的产生式表示

不确定性规则的产生式表示的基本形式为：

IF *P* THEN *Q* （置信度）或者 *P*→*Q*（置信度）

例如：

IF 发烧 THEN 感冒 （0.6），它表示当人发烧的时候，结论是人就会感冒，可以相信的程度为0.6。

1. 确定性事实的产生式表示

确定性事实一般用三元组表示：

（对象，属性，值）或者（关系，对象1，对象2）

例如，“小明年龄为14岁”表示为(Xiaoming, Age, 14)，“小明和小刚是好朋友”表

示为(Friend, Xiaoming, Xiaogang)。

1. 不确定性事实的产生式表示

不确定性事实一般用四元组表示

（对象，属性，值，置信度）或者（关系，对象1，对象2，置信度）

例如，“小明年龄很可能为14岁”表示为(Xiaoming, Age, 14, 0.9)，这里的0.9表示可能性很高。

产生式不仅可以表示精确知识，而且可以表示不精确知识。这是因为在产生式表示知识

的系统中，事实与一条规则的前提条件的匹配可以是不精确的。

把一组产生式放在一起，让它们互相配合，协同作用，一个产生式生成的结论可以供另一个产生式作为已知事实使用，以求得问题得解决，这样的系统称为产生式系统。一般来说，一个产生式系统由以下三部分组成：规则库、综合数据库、推理机。它们之间的关系如图3-2所示。



3-2 产生式系统的基本结构

规则库：用于描述相应领域内知识的产生式集合。规则库是产生式系统求解问题的基础，所以需要对规则库的知识进行合理的组织之和管理，保持知识的一致性。

综合数据库（事实库、上下文、黑板等）：一个用于存放问题求解过程中各种当前信息的数据结构。

推理机：由一组程序组成，除了推理算法，还负责整个产生式系统的运行，实现对问题的求解，具体工作如下：

(1)从规则库中选择与综合数据库中的已知事实进行匹配。

(2)匹配成功的规则可能不止一条，进行冲突消解；

(3)执行某一规则时，如果其右部是一个或多个结论，则把这些结论加入到综合数据库中；如果其右部是一个或多个操作，则执行这些操作；

(4)对于不确定性知识，在执行每一条规则时还要按一定的算法计算结论的不确定性；

(5)检查综合数据库中是否包含了最终结论，决定是否停止系统的运行。

产生式适合于表达具有因果关系的过程性知识，具有表达直观，便于推理，能够表达确定性和不确定性知识等优点。但是产生式也有着效率不高、不能表达具有结构性知识的缺点。

综上所述，产生式表示法适合表示

1. 由许多相对独立的知识元组成的领域知识，彼此间关系不密切，不存在结构关系的知识；
2. 具有经验性及不确定性的知识，而且相关领域中对这些知识没有严格、同一的理论；
3. 领域问题的求解过程可被表示为一系列相对独立的操作，而且每个操作可被表示为一条或多条产生式规则。

**3.2.3框架表示法**

1975年，美国著名的人工智能学者Minsky提出了框架理论：人们对现实世界中各种事物的认识都是以一种类似于框架的结构存储在记忆中。框架表示法便是一种结构化的知识表示方法，已在多种系统中得到应用。

框架(frame)是一种描述所述对象（一个事物、事件或概念）属性的数据结构。一个框架由若干个被称为“槽”（slot）的结构组成，用于描述所论对象某一方面的属性，而每个槽又可以根据实际情况划分为若干个“侧面”，用于描述相应属性的一个方面。槽和侧面所具有的属性值分别被称为槽值和侧面值。

在一个用框架表示知识的系统中一般都含有多个框架，一个框架一般都含有多个不同槽、不同侧面，分别用不同的框架名、槽名及侧面名表示。对于框架、槽或侧面，都可以为其附加上一些说明性的信息，一般是一些约束条件，用于指出什么值才能填入到槽和侧面中去。

框架的一般结构如图3-3所示：



图3-3 框架表示法的一般形式

下面举一个例子，要表示一个教室的组成要素，用框架表示法如下：

<教室>

墙数：

窗数：

门数：

座位数：

墙：<墙框架>

门：<门框架>

窗：<窗框架>

黑板：<黑板框架>

天花板：<天花板框架>

讲台：<讲台框架>

框架表示法最突出的特点是易于表达结构性知识，能够将知识的内部结构关系及知识间的联系表示出来。在框架网络中，下层框架可以继承上层框架的槽值，也可以进行补充和修改，避免了重复描述，节约了时间和空间的开销。框架表示法与人在观察事物时的思维活动是一致的，表示更加自然。

**3.2.4语义网络表示法**

语义网络最早是1968年Quillian在他的博士论文中作为人类联想记忆的一个显式心理学模型提出的。语义网络是一种采用网络形式表示知识的方法。一个语义网络是一个带标识的有向图。其中，带有标识的结点表示问题领域中的概念、物体、事件、动作或者态势，有向边表示概念和概念之间的联系，边上附着的文字称为语义指针。最简单的语义网络是如下的一个三元组：

（节点1，弧，节点2）

也可以用如下图3-4的一个图来表示



图3-4 语义网络表示

其中A,B是节点，RAB是A和B之间的某种语义关系，而且弧线的方向是有意义的。

语义网络中的关系有很多类型，其中包括以下几种：

1. 实例关系（ISA）：体现的是“具体与抽象”的概念，含义是“是一个”，表示一个事物是另一个事物的一个实例，如“小明是一个人”，用语义网络表示便是图3-5：



图3-5 实例关系语义网络表示

1. 分类关系（AKO，a kind of）：也称为泛化关系，体现的是“子类与超类”的概念，含义为”是一种”，表示一个事物是另一个事物的一种类型，如“老虎是一种哺乳动物”。
2. 成员关系（a-member-of）：体现的是“个体与集体”的关系，含义为“是一员”，表示一个事物是另一个事物的一个成员，如“小明是一名教师”。
3. 属性关系：体现的是事物、属性及其取值之间的关系，例如，“小明今年20岁，身高180cm”，用语义网络表示便是图3-6：



图3-6 属性关系语义网络表示

1. 聚合关系：也称为包含关系，指具有组织或者结构特征的“部分与整体”之间的关系，例如“凳子是桌子的一部分”。
2. 时间关系：指不同事件在其发生时间方面的先后次序关系，常用的时间关系有“在前（表示一个事件在另一事件之前发生）”和“在后（表示一个事件在另一个事件之后发生）”，例如“2008年北京夏季奥运会在2022北京冬奥会之前”。
3. 位置关系：指不同事物在位置方面的关系。常用的有“在”，“在上”，“在下”等等。例如“手机在书桌上”。
4. 相近关系：表示事物之间的相似和接近关系。

我们可以按照论元个数把关系分为一元关系、二元关系和多元关系。在表达多元关系时，语义网络可以把多元关系转化为多个二元关系的组合，例子可参考图3-6。

语义网络表示能把事物的属性及事物间的各种语义联系显式地表示出来，便于以联想的方式实现对系统的检索，且便于理解。缺点是语义网络仅用节点及其关系描述知识，推理过程不像谓词逻辑表示方法那样明了，而且目前并没有公认的形式表示体系，所表达的含义依赖于处理程序如何对它进行解释，推理方法不完善。

**3.2.5状态空间表示法**

状态空间（state space）是利用状态变量和操作符号表示系统或问题的有关知识的符号体系。状态空间可以用一个四元组表示：

(S, O , S0, G)

其中，S是状态集合，S中每一元素表示一个状态，状态是某一类事物在不同时刻所处于的信息状况。O是操作的集合，利用操作可以将一个状态转换为另一个状态。S0是问题的初始状态的集合，是S的非空子集。G是问题的目的状态的集合，是S的非空子集。G可以是若干具体状态，也可以是满足某些性质的路径的信息描述。

从S0结点到G结点的路径称为求解路径。求解路径上的操作序列为状态空间的一个解。例如，操作序列O1，...，Ok使初始状态转换为目标状态：



图3-7 状态空间的解

则O1，...，Ok即为状态空间的一个解。当然，解往往不是唯一的。

状态可以用任何数据类型的数据结构来表示，例如符号、字符串、向量、多维数组、树和表格等，所选用的数据结构形式要与状态所蕴含的某些特征具有相似性。

状态空间也可用有向图来描述，图的节点表示问题的状态，图的弧表示状态之间的关系。初始状态对应于实际问题的已知信息，是图的根节点。求解就是找到初始状态转换为目标状态的路径。下面给出状态空间有向图3-8：



图3-8 状态空间有向图

若S9属于G的一个子集，那么通过O1,O5,O9的操作便可让状态S0转换为S9，O1,O5,O9便是一个解。

以上的例子状态比较少，还可以显式展示出来。现实生活中如旅行商问题，节点一多，不可能将所有的解枚举出来，这就需要计算机掌握高级的搜索算法，节省计算时间。

**3.2.6脚本表示法**

脚本是一中与框架类似的知识表示方法，是由Schank等人于二十世纪七十年代提出的。脚本通过一系列的原子动作来表示事物的基本行为，按照事件顺序描述事物的发生。

脚本表示的知识有确定的事件或因果顺序，必须前一个动作完成后才能触发下一个动作的开始。脚本表示法是可以用来描述一个动态过程的知识表示方法。

一个完整的脚本因该包括以下几个重要的组成部分：

1. 进入条件：即脚本描述的事件可能发生的前提条件；
2. 角色：描述事件中可能出现的人物；
3. 道具：描述事件中可能出现的相关物体；
4. 舞台：描述事件发生的空间；
5. 场景：时间发生的序列，指在一定的时间、空间内发生的一定的行动。比如去餐馆吃饭，可能经过等位、点餐、吃饭、付款等场景的切换。然而场景并不是单一的动作序列，可能存在很多分支，对所有可能发生动作序列的枚举是一个很大的工程，这也是脚本表示法的一个缺陷
6. 结局：给出脚本描述事件发生之后产生的结果，对应着进入后续脚本的先决条件。

脚本表示法能比较细致地刻画时序关系的信息，在一些领域有很大的应用，比如智能对话系统。但是这种表示法不具备对元素基本属性的描述能力，也很难去描述多变的事件发展的可能所有方向。

**3.3本章小结**

1. 知识的概念：知识是人类在长期的生活实践中以及科学研究实验中累积起来的，是对客观世界的认识及经验。
2. 知识表示：知识表示是对知识的一种描述，或者说是一种规定，一种计算机可以接受的用于描述知识的数据结构。
3. 逻辑表示法
4. 产生式表示法
5. 框架表示法
6. 语义网络表示法
7. 状态空间表示法
8. 脚本表示法