第 3 章　知识表示

3.1　知识和知识表示的概念

3.1.1　知识的定义与分类

在人工智能领域，“知识就是力量”这句话蕴含着极为深刻的内涵。智能系统之所以能够模拟人类解决复杂问题，核心就在于它能够获取、存储并灵活运用知识。那么，究竟什么是知识？它与我们日常接触的数据、信息又有怎样的区别？

简单来说，知识是人们在长期实践中通过观察、体验、思考积累和总结的，是对客观世界的本质规律、事物属性以及事物之间关联的认识和理解。它不是孤立存在的零散信息，而是经过加工、整理、归纳后形成的具有逻辑关联性和实用价值的认知成果。我们可以通过一个生活化的例子，清晰地区分数据、信息和知识。

数据（Data）：纯粹的、未经解释的符号或数值，不具备任何实际意义。例如，25、10:30、多云、80%。这些孤立的符号本身无法传递有效信息，只是构成信息的原始素材。

信息（Information）：对数据进行解释和赋予含义后形成的描述。例如，“今日最高气温 25 摄氏度”“开往上海的 G123 次列车 10:30 发车”“明天天气多云，相对湿度 80%”。信息让数据有了明确的指向性和用途，但仍然停留在“描述事实”的层面。

知识（Knowledge）：在信息的基础上，通过逻辑推理、归纳总结、经验提炼形成的规律、准则或决策依据。例如，“如果今日最高气温超过 25 摄氏度且湿度大于 70%，那么外出需做好防晒和降温措施”“如果列车发车前 30 分钟仍未到达车站，大概率会错过乘车”。这种“如果……那么……”的表述，或是经过验证的规律总结，才是真正能够指导行动、解决问题的知识。

根据知识的性质、用途和表现形式，人工智能领域通常将其分为以下 3 类，各类知识在智能系统中扮演着不同的角色。

（1）事实性知识（Factual Knowledge）

事实性知识又称陈述性知识，是对客观事物的属性、状态或关系的直接描述，回答“是什么”的问题。这类知识是构建知识库的基础，具有明确的客观性和稳定性。

示例：“地球围绕太阳公转”“水在标准大气压下 100 摄氏度时沸腾”“李白是唐代诗人”“北京大学创办于 1898 年”。

特点：易于用文字、符号等形式明确表述，无须复杂推理即可验证其真实性，是智能系统中最基础的知识类型。

（2）规则性知识（Rule-based Knowledge）

规则性知识又称过程性知识，是描述解决问题的方法、步骤、规则或策略的知识，回答“如何做”的问题。这类知识通常具有明确的条件和结论，是智能系统进行推理决策的核心依据。

示例：“如果患者出现发热、咳嗽、乏力症状，且有疫情中高风险地区旅居史，那么需优先进行核酸检测”“如果想通过搜索引擎找到精准信息，那么应使用具体关键词而非模糊描述”“如果要计算矩形面积，那么用长乘以宽即可得到结果”。

特点：以“条件-动作”或“前提-结论”的形式存在，具有很强的实用性和可操作性，专家系统、智能决策系统等大多以这类知识为核心构建。

（3）元知识（Meta-knowledge）

元知识是关于知识的知识，即指导人们如何运用、选择、管理其他知识的规则和策略。它回答“如何有效使用知识”的问题，是智能系统具备灵活性和高效性的关键。

示例：“在解决数学难题时，优先使用数形结合的方法”“对于紧急医疗问题，优先调用急救相关的专家规则”“如果两条规则存在冲突，选择最新更新的规则进行推理”“当常规方法无法解决问题时，尝试调用启发式知识”。

特点：不直接针对具体问题，而是面向知识本身的管理和运用，能够帮助智能系统在复杂场景中选择最优的推理路径，避免无效计算。

3.1.2　知识表示的含义与重要性

如果把知识比作人工智能的“燃料”，那么知识表示就是将“燃料”转化为机器能够识别、存储和利用的“容器”与“传输管道”。没有合适的知识表示方法，再丰富的知识也无法被智能系统有效运用，就像再多的石油如果没有合适的存储和传输设备，也无法为汽车提供动力。

知识表示（Knowledge Representation）是指将人类的知识通过某种形式化的语言或数据结构转换为计算机能够存储、处理和推理的形式。这个过程本质上是在人类认知与机器运算之间搭建一座桥梁，解决了人工智能的核心难题：如何让机器“理解”人类的知识，并基于这些知识进行思考和决策。

知识表示的重要性体现在人工智能研究和应用的各个层面，具体可分为以下 3 点。

（1）实现机器推理

智能系统的核心能力之一是逻辑推理和决策，而推理的基础是机器能够“读懂”知识。只有将知识以形式化、结构化的方式表示出来，算法才能对这些知识进行比较、匹配、运算，从而得出新的结论。例如，将“所有狗都会叫”和“小黑是一条狗”表示成机器能理解的格式，机器才能自动推理出“小黑会叫”。

（2）构建智能系统

无论是早期的专家系统、如今的智能问答助手，还是自动驾驶汽车、智能推荐系统，其核心都是一个庞大的知识库。这个知识库中的知识，都需要用一种有效的方式进行表示和组织，才能让系统高效地运行。

（3）促进人机协作

良好的知识表示方法，不仅要让机器理解，也要方便人类的理解和维护。一个清晰、直观的知识表示方法，可以让开发者更容易地添加、修改和调试知识，从而不断提升系统的智能水平。

3.1.3 知识表示的基本原则和要求

一个优秀的知识表示方法，需要遵循一些基本原则，以确保其有效性和实用性。

①有效性（Representational Adequacy）：这是最基本的要求。一个知识表示方法必须能够充分表示所需的所有知识。如果一个方法连最简单的“猫有四条腿”这样的事实都无法表示，那么它就是无效的。

②可操作性（Inferential Adequacy）：知识被表示出来后，必须能够支持有效的推理。也就是说，系统应该能够利用这些表示的知识，通过推理机制（如逻辑推理、规则匹配等），来得出新的结论。一个好的表示方法应该让推理过程尽可能简单和高效。

③可维护性（Maintainability）：知识库是动态变化的，需要不断地更新和维护。一个好的知识表示方法应该易于修改、添加和删除知识，而不会对整个系统造成破坏。例如，如果知识是模块化的，那么我们修改其中一条规则时，不会影响到其他部分。

④清晰性（Clarity）：知识表示不仅是给机器看的，也是给人类看的。清晰、直观的表示方法能够帮助人类理解和调试知识库，从而更好地掌控系统的行为。虽然有时候为了提高效率，我们可能会牺牲一些可读性，但在设计时仍应尽可能保持清晰。

3.2 表示知识的方法

要让计算机像人一样思考，第一步就是教会它如何“看懂”和“存储”知识。人类的知识具有多元性和复杂性，既有描述客观事实的静态知识，也有指导行动步骤的动态知识；既有确定无疑的绝对真理，也有模糊不确定的经验判断。因此，人工智能领域发展出了多种知识表示方法，不同方法适用于不同类型的知识和应用场景。

3.2.1 陈述性表示与过程性表示

根据知识的表示重点不同，可将知识表示方法分为陈述性表示和过程性表示，前者关注“是什么”，后者关注“怎么做”，二者相辅相成，共同构成了智能系统的知识基础。

（1）陈述性表示

陈述性表示是一种静态的知识表示方法，主要描述事物的属性、状态、关系等客观事实，重点在于“声明”知识的内容，而不关心这些知识是如何被使用的。它就像一本百科全书或字典，只记录事实和概念，不包含使用这些事实的具体步骤。

它具有如下核心特点。

描述静态事实：仅关注“是什么”，不涉及“如何做”。例如，“地球是太阳系的行星”“三角形的内角和为 180 度”“北京是中国的首都”，这些表述都是对客观事实的直接声明，不包含任何操作步骤。

知识与使用分离：陈述性知识的表示形式与推理机制相互独立，同一条知识可以被不同的推理过程使用，具有很强的复用性。例如，“三角形的内角和为 180 度”这条知识，既可以用于计算锐角三角形的一个未知角，也可以用于验证一个图形是否为三角形。

易于维护和扩展：由于知识是独立的、模块化的，添加新知识时无须修改已有的知识和推理机制，修改或删除知识时也不会影响其他知识的有效性。例如，在地理知识系统中添加“乌鲁木齐是新疆维吾尔自治区的首府”这条知识，无须修改其他城市的相关知识。

推理过程相对复杂：由于陈述性知识不包含使用方法，推理时需要额外的推理机制来搜索和匹配知识，因此推理效率相对较低。

（2）过程性表示

过程性表示是一种动态的知识表示方法，将知识表示为一系列具体的操作步骤和过程，重点在于“描述如何做”，即通过明确的步骤来实现特定的目标。它就像一本菜谱、一份操作手册或一段计算机程序，详细说明了完成某件事情的先后顺序和具体方法。

它具有如下核心特点。

描述动态过程：仅关注“怎么做”，不单独声明事实，知识蕴含在操作步骤中。例如，计算“1+2×3”的过程性知识可以表示为“第一步，先计算乘法 2×3=6；第二步，再计算加法 1+6=7”，这里没有单独声明“乘法优先于加法”的事实，而是通过步骤体现了这一规则。

知识与使用结合：过程性知识的表示形式直接包含了推理或操作的步骤，知识的使用方法已经编码在表示中，无须额外的推理机制即可执行。例如，一段计算平方根的程序，本身就是过程性知识，运行程序的过程就是使用知识的过程。

推理效率高：由于知识已经包含了操作步骤，推理时无须进行复杂的知识搜索和匹配，直接执行步骤即可得到结果，因此推理效率远高于陈述性表示。

可维护性和扩展性差：过程性知识的步骤之间通常存在依赖关系，修改一个步骤可能会影响整个过程的正确性，添加新知识时可能需要重新设计整个操作流程。例如，修改计算“1+2×3”的步骤为“先加法后乘法”，会导致结果错误；添加“计算 1+2×3+4”的步骤时，需要重新调整整个计算流程。

（3）两者的关系

陈述性表示和过程性表示并非相互排斥，而是互补的关系。在实际的智能系统中，通常需要将两种表示方法结合使用，才能既保证知识的灵活性和可维护性，又保证推理的高效性。

这就像我们学习数学的过程：既要记住公式（陈述性知识），如“圆的面积公式为 $S=\pi r^2$”，也要掌握解题步骤（过程性知识），如“如何根据半径计算圆的面积，如何将复杂图形分解为基本图形后计算总面积”。如果只记住公式而不知道如何应用，公式就没有实际意义；如果只知道解题步骤而不理解公式的原理，遇到新的题型就无法应对。

3.2.2　确定性表示与不确定性表示

世界的复杂性在于，许多知识并非百分之百确定。因此，我们需要能够表示不确定性的方法。

（1）确定性表示

确定性表示假定知识是完全准确和可靠的。确定性表示方法处理那些结果明确、边界清晰的知识，通常基于经典逻辑。

它具有如下特点。

绝对性：知识的真值是确定的，要么为真（True），要么为假（False），没有“可能为真”“大概率为真”等中间状态。例如，“2 是偶数”为真，“三角形有四条边”为假，不存在其他

可能性。

精确性：知识的描述是精确无歧义的，不存在模糊不清的表述。例如，“水在标准大气压下的沸点为 100 摄氏度”，这里的“标准大气压”“100 摄氏度”都是精确的概念，没有歧义。

推理简单：基于确定性知识的推理过程遵循经典逻辑的推理规则，推理结果是唯一的、确定的，无须处理概率或模糊性。例如，由“所有鸟都会飞”和“麻雀是鸟”，可以确定地推出“麻雀会飞”。

适用范围有限：仅适用于知识明确、无不确定性的领域，如数学、物理、计算机科学等基础学科，无法处理现实世界中大量存在的模糊和不确定知识。

（2）不确定性表示

不确定性表示承认知识可能是不完整或不精确的。现实世界中很多知识都具有不确定性，不确定性表示方法就是用来处理这类知识的。

它具有如下特点。

概率性或模糊性：知识的真值是不确定的，通常用概率、置信度、隶属度等数值来量化不确定性。例如，“明天有 60%的概率下雨”“患者有 80%的置信度患有流感”“年轻人的隶属度为 0.7（针对 25 岁的人）”。

容错性：能够处理模糊、矛盾或缺失的信息，即使知识不完整或存在轻微矛盾，也能通过不确定性推理得出合理的结论。例如，医生在诊断疾病时，患者可能只表现出部分症状，医生仍能根据已有症状和经验，给出大概率的诊断结果。

推理复杂：基于不确定性知识的推理需要采用专门的不确定性推理算法，如概率推理、模糊推理等，推理结果也是不确定的，通常以概率或置信度的形式呈现。

实用性强：更符合现实世界的实际情况，大多数现实问题都存在不确定性，因此不确定性表示具有更广泛的应用场景。

3.2.3 知识表示方法的评价标准

选择合适的知识表示方法是构建智能系统的关键步骤，不同的表示方法在不同的应用场景中表现各异。要判断一种知识表示方法是否适合某个特定问题，需要从以下 4 个核心维度进行综合评价。

1. 有效性

有效性是评价知识表示方法的首要标准，指该方法能否准确、完整地表达领域内所有必要的知识，包括事实性知识、规则性知识、元知识等，不遗漏关键信息，不扭曲知识的本质。例如，如果一个逻辑系统无法表示时间关系，那么它就无法用于描述“先发生 A，然后发生 B”的知识。

2. 可操作性

知识表示出来后，必须能够支持有效的推理机制，从而得出新的结论。一个好的表示方法应该能够让推理过程尽可能简单和高效，避免不必要的复杂性。

3. 可维护性

知识库是动态变化的，需要不断地更新和维护。一个好的表示方法应该易于修改、添加

和删除知识，而不会对整个系统造成破坏。例如，如果知识是模块化的，那么修改一条规则时，不会影响到其他部分。

4. 清晰性

清晰性指知识的表示形式是否直观、易懂，便于人类开发者和领域专家理解、调试和维护知识库。

3.3　产生式表示法

产生式表示法是人工智能中最常用的一种知识表示方法，它来源于人类心理学的产生式系统理论。这种表示方法模仿了人类解决问题的思维方式："如果出现某种情况，那么就采取某种行动。"

产生式表示法是人工智能领域中应用最广泛、最成熟的知识表示方法之一，它起源于 20 世纪 40 年代美国心理学家桑代克（Edward Thorndike）提出的"试误说"学习理论，后来经认知科学家西蒙（Herbert Simon）和纽厄尔（Allen Newell）进一步发展，形成了用于人工智能系统的产生式系统理论。这种表示方法模仿了人类解决问题时的思维模式："如果遇到某种情况，就采取某种行动，"具有自然直观、模块化强、易于维护等优点，被广泛应用于专家系统、智能决策、故障诊断等领域。

3.3.1　产生式系统的基本结构

产生式系统是基于产生式规则构建的智能系统，其核心思想是将知识表示为一系列"条件-动作"形式的规则，通过推理机对这些规则进行匹配、选择和执行，从而实现问题求解。一个完整的产生式系统由知识库（规则库）、综合数据库和推理机 3 个核心部分组成，三者协同工作，构成一个"感知-思考-行动"的闭环系统。

1. 知识库（规则库）

知识库又称规则库，是产生式系统的"大脑"，用于存储和管理所有由产生式规则表示的领域知识。每条规则都是一个独立的知识单元，形如"如果（IF）…，那么（THEN）…"，因此也被称为"IF-THEN"规则。例如，"IF 湿度很高 AND 温度很高 THEN 可能会下雨"。

2. 综合数据库

综合数据库又称事实库、工作内存（Working Memory），是产生式系统的"临时记忆"，用于存储当前问题的初始事实、推理过程中产生的中间结论，以及最终的推理结果。综合数据库中的数据是动态变化的，随着推理过程的进行，新的事实会不断被添加，过时的事实可能会被删除或标记为无效。

3. 推理机

这是系统的“心脏”，负责驱动整个系统运行。它不断地在综合数据库中寻找与知识库中规则的条件相匹配的事实，一旦匹配成功，就执行规则的结论部分，并可能将新生成的事实或结论存入综合数据库。这个过程会一直重复，直到达成最终目标或无法再进行新的推理为止。

推理机具有如下的主要功能。

规则匹配：遍历知识库中的所有规则，检查每条规则的条件部分是否与综合数据库中的事实相匹配。

冲突消解：当有多个规则的条件都与综合数据库中的事实相匹配时，推理机需要根据一定的策略选择一条规则优先执行，这个过程称为冲突消解。

规则执行：执行被选中规则的结论部分，将结论添加到综合数据库中，或执行结论中指定的动作（如生成诊断报告、发出预警等）。

推理控制：控制推理过程的启动、暂停和终止。当综合数据库中出现目标事实（如最终诊断结果）时，推理过程终止；当没有可匹配的规则且未达到目标时，也会终止推理并提示无法求解。

产生式系统的工作流程如图 3.1 所示。推理机首先从综合数据库中读取初始事实，然后在知识库中寻找匹配的规则，通过冲突消解选择一条规则执行，将新的结论添加到综合数据库中，之后重复上述过程，直到得出最终结论或无法继续推理为止。

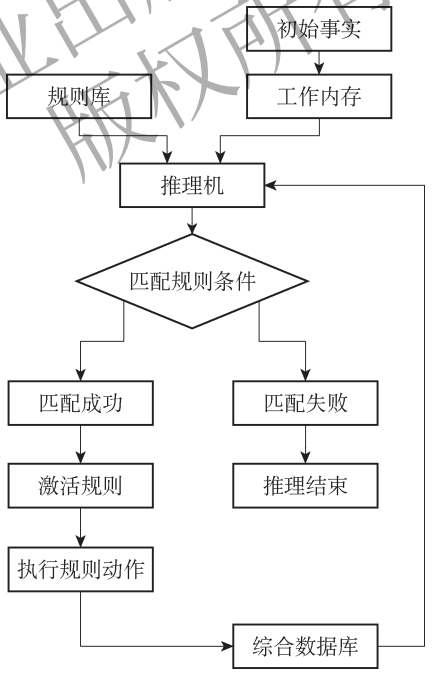

图 3.1　产生式系统的工作流程

3.3.2　产生式规则的表示形式

产生式规则是产生式系统的基本单位，通常采用一种直观的“IF-THEN”结构来表示。

IF 部分：被称为条件或前提。它通常由一个或多个事实、状态或模式组成，这些事实之间可以通过逻辑运算符（如 AND、OR、NOT）连接。

THEN 部分：被称为结论或动作。它定义了在条件满足时系统应该采取的行动或得出的新结论。

【示例】

IF 天空有乌云 THEN 可能会下雨，其中，条件：天空有乌云；结论：可能会下雨。

复杂规则：

IF 股价上涨 AND 交易量很大 THEN 股票处于牛市，其中，条件：股价上涨 和 交易量很大；结论：股票处于牛市。

这种形式的优点在于其直观性和模块化。每一条规则都是独立的知识模块，可以方便地进行添加、删除或修改，而无须改变其他规则。

3.3.3　产生式系统的推理过程

产生式系统的推理过程是推理机按照一定的策略，利用知识库中的规则对综合数据库中的事实进行匹配、执行，最终得出结论的过程。根据推理的方向不同，产生式系统的推理过程主要分为正向推理和反向推理两种，此外还有将二者结合的混合推理。

（1）正向推理（数据驱动）

正向推理又称数据驱动推理，是一种从“初始事实”到“目标结论”的推理方式。它以综合数据库中的初始事实为起点，不断地在知识库中寻找匹配的规则，执行规则并添加新的事实，直到得出目标结论或无法继续推理为止。步骤如下：

①初始化综合数据库，将用户提供的初始事实存入其中。

②遍历知识库中的所有规则，检查每条规则的条件部分是否与综合数据库中的事实完全匹配。

③若找到匹配的规则，则将其加入“冲突集”（所有可匹配规则的集合）。

④若冲突集非空，则根据冲突消解策略选择一条规则执行；若冲突集为空，则推理终止，说明无法得出目标结论。

⑤执行选中规则的结论部分，将新的事实或结论添加到综合数据库中。

⑥检查综合数据库中是否包含目标事实，若包含，则推理成功，输出目标结论；若不包含，则返回步骤②继续进行下一轮匹配。

【示例】

知识库规则：

R1：IF 体温超过 38.5℃ AND 有发热症状 THEN 建议服用退烧药。

R2：IF 咳嗽 AND 咳痰为黄色 THEN 怀疑有细菌感染。

R3：IF 怀疑有细菌感染 AND 无青霉素过敏史 THEN 建议使用青霉素治疗。

初始事实：{"体温：39℃","症状：发热","症状：咳嗽","咳痰：黄色","青霉素过敏史：无"}。

目标结论：给出治疗建议。

推理过程：

①综合数据库初始状态：{"体温：39℃", "症状：发热", "症状：咳嗽", "咳痰：黄色", "青霉素过敏史：无"}。

②第一轮匹配。

R1 的条件“体温超过 38.5℃ AND 有发热症状”与事实匹配。

R2 的条件“咳嗽 AND 咳痰为黄色”与事实匹配。

冲突集：{R1, R2}。

冲突消解：选择优先级相同的规则，按顺序执行 R1。

执行 R1：添加“建议服用退烧药”到综合数据库。

③第二轮匹配。

将综合数据库更新为：{"体温：39℃", "症状：发热", "症状：咳嗽", "咳痰：黄色", "青霉素过敏史：无", "建议服用退烧药"}。

再次遍历规则，R2 仍匹配，R3 的条件“怀疑有细菌感染”尚未满足。

冲突集：{R2}。

执行 R2：添加“怀疑有细菌感染”到综合数据库。

④第三轮匹配。

将综合数据库更新为：{"体温：39℃", "症状：发热", "症状：咳嗽", "咳痰：黄色", "青霉素过敏史：无", "建议服用退烧药", "怀疑有细菌感染"}。

R3 的条件“怀疑有细菌感染 AND 无青霉素过敏史”与事实匹配。

冲突集：{R3}。

执行 R3：添加“建议使用青霉素治疗”到综合数据库。

⑤检查目标：综合数据库中已包含治疗建议，推理终止。

⑥最终结论：{"建议服用退烧药", "建议使用青霉素治疗"}。

（2）反向推理

反向推理又称目标驱动推理，是一种从“目标结论”到“初始事实”的推理方式。它以用户提出的目标为起点，不断地在知识库中寻找能够推导出该目标的规则，将规则的条件作为新的子目标，递归地验证子目标是否存在于综合数据库中，直到所有子目标都被验证为真，或无法找到匹配的规则为止。步骤如下：

①设定一个目标，例如，“明天会下雨吗？”

②在知识库中寻找其结论与目标相匹配的规则，例如，“IF 天空有乌云 THEN 会下雨”。

③将规则的条件（天空有乌云）作为新的子目标。

④重复以上步骤，直到所有的子目标都可以在综合数据库的事实中找到。

换一种表述为：

①设定目标事实（即用户想要证明或求解的问题）。

②检查综合数据库中是否已存在该目标事实，若存在，目标得证，推理终止。

③如果综合数据库中不存在该目标事实，则需要遍历知识库中的所有规则，寻找结论部分与目标事实相匹配的规则。

④如果找到匹配的规则，则将该规则的条件部分作为新的子目标；如果未找到匹配的规则，则目标无法得证，推理终止。

⑤递归地对每个子目标执行步骤②～④，直到所有子目标都被验证为真（目标得证），或某个子目标无法得证（目标无法得证）。

【示例】

目标：会下雨吗？

规则：IF 天空有乌云 THEN 会下雨。

推理：系统反向追溯，发现要证明“会下雨”，就需要证明“天空有乌云”这个条件。于是，它会去事实库中寻找“天空有乌云”这个事实。如果找到，则目标得证；如果没找到，则继续寻找能证明“天空有乌云”的新规则。

知识库规则：同上。

初始事实：同上。

目标事实："建议使用青霉素治疗"。

推理过程：

①目标：“建议使用青霉素治疗”。

②检查综合数据库：无该目标事实。

③寻找结论为该目标的规则：R3（IF 怀疑有细菌感染 AND 无青霉素过敏史 THEN 建议使用青霉素治疗）。

④设定子目标："怀疑有细菌感染" AND "无青霉素过敏史"。

⑤验证子目标 1："无青霉素过敏史"。

检查综合数据库：存在该事实，子目标 1 得证。

⑥验证子目标 2：“怀疑有细菌感染”。

检查综合数据库：无该事实。

寻找结论为该子目标的规则：R2（IF 咳嗽 AND 咳痰为黄色 THEN 怀疑有细菌感染）。

设定子目标："咳嗽" AND "咳痰为黄色"。

⑦验证子目标 2.1：“咳嗽”。

检查综合数据库：存在该事实，子目标 2.1 得证。

⑧验证子目标 2.2：“咳痰为黄色”。

检查综合数据库：存在该事实，子目标 2.2 得证。

⑨子目标 2 得证，因此 R2 的结论成立，“怀疑有细菌感染”被添加到综合数据库中。

⑩所有子目标都得证，因此 R3 的结论成立，目标“建议使用青霉素治疗”得证。

⑪终止推理，输出目标结论。

（3）混合推理

混合推理是将正向推理和反向推理结合起来的推理方式，它综合了二者的优点，既可以利用初始事实驱动推理，又可以通过目标引导推理方向，提高推理效率和准确性。

3.3.4　产生式表示法的特点与应用

产生式表示法具有诸多显著优点。首先是自然性，其“IF-THEN”的形式与人类日常的思维和表达习惯相似，人们在描述知识和规则时能够轻松上手，比如“如果下雨，那么出门要带伞”，这种表述方式通俗易懂。其次是模块性，每条产生式规则都是独立的知识单元，规则之间相互独立，修改、添加或删除某条规则，通常不会对其他规则产生较大影响，便于知识的管理和维护。再次是有效性，产生式表示法能够很好地表示确定性知识和不确定性知识，无论是像数学定理这样明确的知识，还是医疗诊断中带有概率性的知识，都能准确表达。最后清晰性也是其一大特点，规则的条件和结论一目了然，整个知识体系结构清晰，易于理解和解读。

然而，产生式表示法也存在一些不足。效率不高是较为突出的问题，在推理过程中，需要

对规则库中的大量规则进行逐一匹配，当规则数量众多时，匹配过程会消耗大量时间和资源，影响系统运行速度。此外，它不太擅长表达结构性知识，对于具有复杂层次结构和内在联系的知识，用产生式表示可能会显得烦琐且难以体现其结构关系。

在实际应用中，产生式表示法在多个领域发挥着重要作用。在专家系统中，它被广泛用于表示专家的经验和知识。例如，医疗专家系统，将医生诊断疾病的经验总结成产生式规则，如“IF 患者出现胸痛、呼吸困难症状 AND 心电图显示 ST 段抬高 THEN 患者可能患有心肌梗死”，系统根据患者的症状和检查结果，运用这些规则进行推理，辅助医生做出诊断。在工业生产的自动化控制中，产生式表示法可用于制定生产流程规则。比如“IF 产品质量检测不合格 THEN 进行返工处理”，通过这些规则实现生产过程的自动化决策和控制。在智能教学系统中，也能利用产生式规则根据学生的学习情况和答题表现，提供个性化的学习建议和指导。

产生式表示法作为人工智能领域最经典的知识表示方法之一，具有显著的优点和一定的局限性，其应用场景也主要集中在适合“规则推理”的领域。

1. 产生式表示法的优点

（1）自然性和直观性

产生式规则的“IF-THEN”结构与人类日常的思维方式和表达习惯高度一致，例如，“如果下雨，那么带伞”“如果体温升高，那么多喝水”，这种表述方式通俗易懂，领域专家可以直接参与规则的编写和维护，无须深入掌握复杂的计算机技术。

（2）模块化和独立性

每条产生式规则都是一个独立的知识单元，规则之间没有直接的依赖关系，添加、修改或删除一条规则都不会影响其他规则的有效性。例如，在医疗专家系统中，新增一条“糖尿病的诊断规则”，无须修改已有的“高血压诊断规则”，大大降低了知识库的维护成本。

（3）有效性和灵活性

产生式表示法能够灵活地表示各种类型的知识，包括事实性知识、规则性知识、不确定性知识和元知识。

（4）易于实现和扩展

产生式系统的结构简单清晰，知识库、综合数据库和推理机相互独立，便于程序实现。随着领域知识的更新和扩展，只需不断向知识库中添加新的规则，无须改变系统的整体架构，系统就具有良好的可扩展性。

2. 产生式表示法的局限性

（1）推理效率较低

在大规模知识库中，推理机需要遍历所有规则才能找到匹配的规则，当规则数量达到数万条甚至数百万条时，匹配过程会耗费大量的时间和计算资源，导致推理效率低下。此外，当存在大量匹配规则时，冲突消解过程也会增加推理的复杂度。

（2）不擅长表示结构性知识

产生式表示法适合表示“条件-结论”形式的规则性知识，但不擅长表示具有复杂层次结构和内在联系的结构性知识。例如，“汽车的结构由发动机、底盘、车身、电气设备组成，发动机又由气缸、活塞、曲轴等部件组成”，这种具有层次关系的知识用产生式规则表示会非常烦琐，且无法清晰地体现其结构关系。

（3）知识获取难度大

对于复杂的领域，构建一个完整的产生式规则库需要领域专家投入大量的时间和精力，手动提取和编写规则。尤其是在知识模糊、边界不清晰的领域，规则的提取和验证难度极大，容易出现规则遗漏、冲突或冗余的情况。

（4）缺乏对知识的整体把握

产生式规则是零散的知识单元，没有对知识的整体结构进行描述，因此人类开发者和领域专家难以从全局上把握知识库的内容，不利于知识库的优化和调试。例如，在包含数千条规则的医疗专家系统中，很难快速判断哪些规则是相关的，哪些规则存在冲突。

3. 产生式表示法的典型应用

尽管存在一定的局限性，但产生式表示法仍然在多个领域得到了广泛的应用，尤其是在需要规则推理、知识模块化的场景中。

（1）专家系统

专家系统是产生式表示法最经典的应用场景。专家系统通过将领域专家的知识表示为产生式规则，模拟专家的推理过程，解决特定领域的复杂问题。

（2）智能决策系统

在企业管理、金融投资、物流调度等领域，智能决策系统利用产生式规则表示决策逻辑，根据实时数据和环境信息，自动做出决策。

（3）智能控制与自动化系统

在工业生产、智能家居等领域，产生式表示法用于表示控制规则，实现自动化控制。

（4）智能教学系统

智能教学系统利用产生式规则表示教学策略和知识点关联，根据学生的学习情况，提供个性化的学习指导。

3.4　框架表示法

框架表示法是一种用于表示结构化知识的人工智能技术，由 Marvin Minsky（明斯基）于 1974 年提出。它模仿人类在理解新情境时使用先前经验的方式，通过“框架”这种数据结构来组织和表示现实世界中的各类概念和情境。

框架表示法特别适合表示具有固定结构、多属性、层次关系明确的知识，如物体的分类、事件的描述、场景的建模等，在自然语言理解、智能检索、专家系统等领域有着广泛的应用。

3.4.1　框架理论的基本思想

人类在认识和理解新事物时，并不是从零开始的，而是会从记忆中调用一个与新事物相关的“经验框架”，然后根据新事物的具体情况，对这个框架进行修改和补充，从而快速形成对新事物的认识。例如，当听到“苹果”这个词时，人们大脑中会立刻激活一个“苹果”的框架，这个框架包含了苹果的典型属性：颜色（红色、绿色）、形状（圆形）、味道（甜）、用途（食用）、类别（水果）等。即使看到一个从未见过的绿色苹果，也能通过这个框架快速理解它

是苹果的一种，具备苹果的核心属性。

当有人说“我昨天去看电影了”，大脑会激活“看电影”的框架，这个框架包含了看电影的典型要素：参与者、电影名、时间、地点、情节、感受等。即使对方没有详细说明，也能通过这个框架推测出可能包含的信息，如“他可能和朋友一起去的”“他可能觉得电影很好看”。

框架理论正是将人类的这种认知模式形式化，其基本思想可以概括为以下 4 点。

（1）结构化存储

知识不是孤立存在的，而是以结构化的方式组织在一起。每个框架对应一个事物或概念，框架内部通过“槽”（Slot）和“值”（Value）的形式，存储该事物的属性、特征和关系，形成一个完整的知识单元。

（2）默认值机制

框架中为一些常见属性预设了默认值（Default Value）。当缺乏关于该属性的具体信息时，系统会自动使用默认值进行推理，从而处理不完整的信息。例如，在“鸟”的框架中，“会飞”可以作为默认值，当提到“麻雀”时，即使没有明确说明“麻雀会飞”，系统也会默认它具备这个属性；只有当提到“鸵鸟”“企鹅”等不会飞的鸟时，才会修改这个默认值。

（3）继承特性

框架可以通过继承关系（Inheritance）组织成层次结构。子框架（子类）可以自动继承父框架（父类）的所有属性和特征，无须重复定义。例如，“汽车”是“交通工具”的子框架，“交通工具”的框架包含“轮子数量”“动力来源”等槽，“汽车”框架无须再次定义这些槽，直接继承即可，只需补充“品牌”“发动机类型”等汽车特有的属性。

（4）预期驱动

框架包含了对事物典型特征的预期，能够指导系统主动收集相关信息，填补框架中的空白槽。例如，“学生”框架包含“姓名”“年龄”“专业”“院校”等槽，当系统激活“学生”框架后，会主动询问用户这些信息，以完善对该学生的描述。

框架表示法的核心价值在于，它能够将分散的属性和特征组织成一个结构化的整体，清晰地体现事物的层次关系和内在联系，同时通过默认值和继承机制，高效地处理不完整信息，减少知识的冗余存储。

3.4.2 框架的结构与组织方式

1. 框架的基本结构

一个框架（Frame）是一个用于描述某类事物或概念的结构化数据结构，其核心组成部分包括框架名、槽、侧面和值。基本结构如图 3.2 所示。

各组成部分的说明如下。

（1）框架名

框架的唯一标志，通常是一个能够准确描述事物或概念的名称，如“手机”“大学生”“生日聚会”等。

（2）槽

框架的核心组成部分，用于描述事物的一个属性、特征或关系。每个槽对应事物的一个方面，如“手机”框架的“品牌”“屏幕尺寸”“操作系统”等槽，“生日聚会”框架的“参与者”“时间”“地点”等槽。

```
框架名：<框架标识>  // 框架的唯一名称，用于区分不同的框架
  槽1：<槽名1>  // 描述事物的一个属性或特征
    侧面1.1：<侧面名1.1>  // 对槽的进一步说明或约束（如数据类型、取值范围、提问方式等）
      值1.1.1：<具体值1>  // 槽的具体内容，可以是常量、变量、函数、甚至另一个框架
      值1.1.2：<具体值2>
    侧面1.2：<侧面名1.2>
      值1.2.1：<具体值>
  槽2：<槽名2>
    侧面2.1：<侧面名2.1>
      值2.1.1：<具体值>
  ...
  槽n：<槽名n>
    侧面n.1：<侧面名n.1>
      值n.1.1：<具体值>
```

图 3.2　框架的基本结构

（3）侧面

对槽的补充说明和约束，用于细化槽的属性，提高知识表示的准确性。常见的侧面类型包括取值范围、数据类型等。

取值范围：限定槽的取值集合，如“性别”槽的取值范围为{男，女}。

数据类型：限定槽的值的类型，如“年龄”槽的数据类型为整数，“价格”槽的数据类型为浮点数。

默认值：槽的默认取值，当没有具体信息时使用。

提问方式：系统获取槽值的提问语句，如“请输入该手机的品牌？”

过程依附：依附于槽的函数或程序，用于动态计算槽值或执行特定动作，如“手机”框架的“续航时间”槽可以依附一个程序，根据“电池容量”和“屏幕功耗”计算出续航时间。

（4）值

槽的具体内容，是框架存储的知识核心。值的类型可以是多样的。

常量值：如“品牌”槽的值为“苹果”“华为”，“年龄”槽的值为 20。

变量值：如“姓名”槽的值为“？X”（表示未知，需要用户输入）。

函数或表达式：如“面积”槽的值为“长×宽”。

其他框架：如“学生”框架的“所属院校”槽的值可以是“清华大学”框架，体现框架之间的关联。

2. 框架的组织方式

单个框架只能描述某一类事物的属性，但现实世界中的事物往往存在复杂的层次关系和关联关系。因此，框架通常通过以下两种方式组织成一个完整的知识体系。

（1）层次结构（继承关系）

框架之间通过“AKO”（A Kind Of，“是一种”）槽建立继承关系，形成树状的层次结构。上层框架被称为父框架（父类），下层框架被称为子框架（子类），子框架自动继承父框架的所有槽和侧面，同时可以定义自己特有的槽和侧面，或重写父框架的默认值。

层次结构的框架例子如图 3.3 所示。

在这个示例中，“汽车”是“交通工具”的子框架，继承了“轮子数量”“动力来源”槽，新增了“品牌”“发动机类型”槽；“电动车”是“汽车”的子框架，继承了“交通工具”和“汽车”的所有槽，重写了“动力来源”的默认值，新增了“电池容量”槽。

```
框架名：交通工具  // 父框架
  槽1：轮子数量
    侧面：默认值 → 4
    侧面：取值范围 → {0, 2, 3, 4, 6, ...}
  槽2：动力来源
    侧面：取值范围 → {燃油，电力，人力，混合动力}
    侧面：提问方式 → “该交通工具的动力来源是什么？”

框架名：汽车  // 子框架，AKO → 交通工具
  槽3：品牌  // 汽车特有的槽
    侧面：提问方式 → “该汽车的品牌是什么？”
  槽4：发动机类型
    侧面：取值范围 → {汽油发动机，柴油发动机，电动发动机}
  槽1：轮子数量  // 重写父框架的默认值
    侧面：默认值 → 4（与父框架一致，可省略）

框架名：电动车  // 子框架，AKO → 汽车
  槽2：动力来源  // 重写父框架的取值
    侧面：默认值 → 电力
  槽5：电池容量
    侧面：数据类型 → 浮点数（单位：kWh）
    侧面：提问方式 → “该电动车的电池容量是多少kWh？”
```

图 3.3　层次结构的框架例子

（2）关联结构（横向关系）

除了纵向的继承关系，框架之间还可以通过“ISA”（Is A，“是一个”）、“PART-OF”（“是…的一部分”）、“RELATED-TO”（“与…相关”）等槽建立横向的关联关系，描述不同类别事物之间的联系。

关联结构的框架例子如图 3.4 所示。

```
框架名：手机  // 主体框架
  槽1：品牌 → 苹果
  槽2：操作系统 → iOS
  槽3：生产厂商  // 关联到“苹果公司”框架
    侧面：类型 → 框架
    值 → 苹果公司框架
  槽4：屏幕供应商  // 关联到“三星”框架
    侧面：类型 → 框架
    值 → 三星框架

框架名：苹果公司  // 关联框架
  槽1：成立时间 → 1976年
  槽2：总部地点 → 美国加利福尼亚州库比蒂诺
  槽3：主要产品 → [手机，电脑，平板]（关联到对应的框架）

框架名：三星  // 关联框架
  槽1：成立时间 → 1938年
  槽2：总部地点 → 韩国首尔
  槽3：主要业务 → [电子设备，屏幕制造，半导体]
```

图 3.4　关联结构的框架例子

在这个示例中，“手机”框架通过“生产厂商”槽与“苹果公司”框架关联，通过“屏幕供应商”槽与“三星”框架关联。

3.4.3　框架系统例子——动物识别框架系统

为直观理解框架系统的工作原理，我们构建一个简易的动物识别框架系统，通过层次化

框架和交互式推理，实现对动物类别的判断。

系统的知识库包含以下框架，形成一个简单的层次结构。

```
框架名：动物
槽：皮肤覆盖
侧面：提问语句->“它的身体表面覆盖着什么？(毛发/羽毛/鳞片)”
侧面：取值范围→{毛发、羽毛、鳞片、皮肤…}
槽：移动方式
侧面：提问语句→“它的移动方式是什么？（飞行/游泳/行走）”
侧面：取值范围→{飞行、游泳、行走、爬行…}

框架名：鸟
AKO：鸟-继承自动物，所以也有移动方式和皮肤覆盖槽
槽：皮肤覆盖 (重写默认值)
值：羽毛 (默认值)
槽：移动方式 (重写默认值)
值：飞行 (默认值)
槽：典型特征
侧面：提问语句 -> "它会下蛋吗？"

框架名：鱼
AKO：动物
槽：皮肤覆盖
值：鳞片 (默认值)
槽：移动方式
值：游泳 (默认值)
槽：典型特征
侧面：提问语句 -> "它用鳃呼吸吗？"

框架名：知更鸟
AKO：鸟-是一种鸟，继承鸟和动物的所有属性
槽：颜色
值：红色胸脯 (默认值)

框架名：金枪鱼
AKO：鱼
槽：颜色
值：银蓝色 (默认值)
```

系统的推理过程介绍如下。

假设用户观察到一个未知动物，系统通过激活框架、填充槽值、匹配子框架，逐步推理类别。

步骤 1：激活顶层框架。用户输入“我看到一个动物”，系统激活顶层的“动物”框架，准备收集核心属性信息。

步骤 2：填充“皮肤覆盖”槽。系统查看“动物”框架的槽，优先询问关键属性“皮肤覆盖”。

系统提问：“它的身体表面覆盖着什么？（毛发 / 羽毛 / 鳞片 / 皮肤）”

用户回答：“羽毛。”

系统匹配：“羽毛”与“鸟”框架的“皮肤覆盖”默认值完全一致，将当前焦点切换到“鸟”框架。

步骤 3：验证继承属性与新增特征系统继承“鸟”框架的属性，进一步确认细节：

系统提问（移动方式）：“它的移动方式是什么？（飞行/游泳/行走）”（继承自“动物”框架，使用“鸟”的默认值验证）

用户回答：“飞行。”

系统提问（典型特征）：“它会下蛋吗？（是/否）”（“鸟”框架新增槽）

用户回答：“会。”

匹配结果：所有属性均符合“鸟”框架的预期，继续尝试更具体的子框架。

步骤 4：匹配细分子类（知更鸟）。

系统激活“鸟”的子框架“知更鸟”，验证特有属性：

系统提问：“它的胸脯是红色的吗？（是/否）”

用户回答：“不是。”

匹配结果：与“知更鸟”的默认值冲突，排除该子框架，保持“鸟”框架为当前焦点。

步骤 5：得出最终结论。

系统遍历所有子框架后，无更匹配的细分类别，基于已确认信息输出结果。

系统输出：“该动物是一只鸟。核心特征：身体覆盖羽毛，移动方式为飞行，会下蛋。”

3.4.4 框架表示法的优缺点

1. 优点

框架表示法在知识表示领域具有诸多显著优势。

（1）结构化

能够很好地表示复杂事物的结构、属性和层次关系，使知识组织更加清晰。

（2）易于理解

其表示方式非常直观，类似于面向对象编程中的类和对象，方便人类理解和维护。

（3）支持默认推理

能够处理不完整的信息，并通过默认值来快速进行推断。

（4）节省空间

通过继承机制，避免了知识的重复存储，提高了存储效率。

2. 缺点

然而，框架表示法也并非十全十美，存在一些不足之处。

（1）不擅长表示规则性知识

虽然可以通过过程依附来部分解决，但它不适合表示像“如果…那么…”这样纯粹的推理规则。

（2）推理的灵活性有限

其推理机制主要依赖于结构本身，对于一些需要复杂逻辑推理的问题，处理能力相对较弱。

（3）知识获取困难

构建大规模框架库需耗费大量人力，需逐一梳理事物的属性、约束和层次关系，尤其在知识边界模糊的领域更困难。

（4）可维护性较差

框架间的层次和关联关系一旦确定，修改成本较高，若变更了父框架属性，可能需同步调整所有子框架。

3.5　知识图谱

随着互联网的爆炸式增长，我们面临着一个挑战：如何从海量的非结构化数据（如网页、新闻、社交媒体）中提取知识，并让机器像人一样理解世界？传统的知识表示方法，如规则和框架，很难处理如此庞大且动态变化的信息。知识图谱（Knowledge Graph）应运而生，它提供了一种全新的、面向大规模互联知识的表示方法。

3.5.1　知识图谱的基本概念与发展历程

随着互联网海量非结构化数据（网页、新闻、社交媒体内容）的爆发，传统知识表示方法（如规则、框架）难以处理大规模、动态关联的知识。知识图谱以图结构为核心，将知识表示为“实体-关系-实体”的网络，实现对万物关联的高效建模，成为人工智能领域处理大规模知识的核心技术之一。

1. 基本概念

知识图谱是一种用图结构表示知识、建模万物关联的技术方法，本质是大规模语义网络，核心组成包括 4 类元素。

实体：现实世界中的具体对象，是图中的“节点”，如“爱因斯坦”“北京大学”“华为手机”“北京市”。

概念：具有相同特征的实体集合（类别），如“科学家”“大学”“手机品牌”“城市”，可作为实体的“类型标签”。

关系：实体或概念之间的语义关联，是图中的“边”，如“爱因斯坦-出生于-德国”“北京大学-位于-北京市”。

属性：实体的特征描述，通常表示为“实体-属性-值”的三元组，如“爱因斯坦-出生日期-1879 年 3 月 14 日”“华为手机-价格-5999 元”。

知识图谱的核心表示单元是三元组（Triple），其形式为（头实体，关系，尾实体）或（实体，属性，值）。例如：

（姚明，身高，2.26 米）→属性三元组。

（姚明，效力过，休斯顿火箭队）→关系三元组。

（休斯顿火箭队，位于，美国）→关系三元组。

这些三元组相互连接，形成一张庞大的知识网络，直观呈现实体间的复杂关联。

2. 发展历程

知识图谱的思想源于早期语义网络，但随着技术和数据的发展逐步成熟，可分为 3 个阶段。

（1）萌芽期（20 世纪 80 年代—21 世纪初）

核心是“语义网络”和早期专家系统，研究者尝试用“节点-边”表示知识关联，但受限于数据量小、技术不成熟，仅应用于小规模领域（如医疗诊断专家系统）。

（2）发展期（2007—2011 年）

随着互联网结构化数据的积累，维基百科等项目推动了知识库的规模化构建。2007 年，DBpedia 项目将维基百科的信息转化为结构化三元组，Freebase、YAGO 等知识库相继出现，为知识图谱奠定数据基础。

（3）爆发期（2012 年至今）

2012 年，谷歌正式提出“知识图谱”概念，并将其应用于搜索引擎，用户搜索“爱因斯坦”时，不仅返回网页，还直接展示其生平、成就、关联人物等结构化信息，极大提升了搜索体验。此后，知识图谱成为人工智能、大数据领域的研究热点，被广泛应用于智能问答、推荐系统、金融风控等行业。

3.5.2 知识图谱的表示与存储方式

知识图谱主要采用图数据结构来表示和存储知识。

1. 表示方式

知识图谱的核心是三元组（Triple），其形式为 (头实体, 关系, 尾实体)。例如，(爱因斯坦, 出生于, 德国)、(相对论, 提出了, 爱因斯坦)、(普林斯顿大学, 位于, 新泽西州)。这些三元组共同构成了一张巨大的知识网络。实体还通常包含其属性，例如，爱因斯坦的属性可能是国籍、出生日期等。

为了让机器更好地处理知识图谱，还会采用“图嵌入（Graph Embedding）”技术，将实体和关系映射为低维向量（数值数组）。例如，将“爱因斯坦”映射为[0.2, 0.5, -0.3]，“德国”映射为[0.4, -0.1, 0.6]，通过向量运算可计算实体间的关联强度，支持更复杂的推理。

2. 存储方式

知识图谱的存储需要高效支持实体关联查询（如“查找与爱因斯坦相关的所有科学家”），传统关系型数据库（如 MySQL）难以应对复杂图结构的高效查询，因此主要采用图数据库存储。

图数据库的核心优势是“以关系为核心”，直接存储实体和关系的关联，查询时无须多表连接，效率远高于关系型数据库。常见的图数据库有 Neo4j、ArangoDB 等。

3.5.3 知识图谱的构建方法与技术

构建大规模知识图谱是复杂的系统工程，核心流程包括知识抽取、知识融合、知识推理三个步骤，每个步骤都依赖自然语言处理、机器学习等技术。

1. 知识抽取：从数据中提取知识

知识抽取的目标是从非结构化（如新闻文本）、半结构化（如表格、网页列表）数据中，自动提取三元组（实体、关系、属性），核心任务包括三项。

实体识别：从文本中识别出实体，标注类型（如人名、地名、组织名）。例如，从“爱因

斯坦 1879 年出生于德国乌尔姆市”中，识别出实体“爱因斯坦”（人名）、“德国”（地名）、“乌尔姆市”（地名）、“1879 年”（时间）。

关系抽取：识别实体间的语义关系。例如，从“姚明效力于休斯顿火箭队”中，提取关系“效力于”，形成三元组（姚明，效力于，休斯顿火箭队）。

属性抽取：提取实体的属性值。例如，从“苹果 iPhone15 的屏幕尺寸为 6.1 英寸，售价 5999 元起”中，提取属性“屏幕尺寸”（值：6.1 英寸）、“售价”（值：5999 元起）。

2. 知识融合：整合多源知识

由于知识可能来自不同数据源（如维基百科、新闻网站、企业内部数据），存在冗余、冲突、格式不一致等问题，知识融合的目标是“去重、消歧、统一”，构建统一的知识库，核心任务包括实体对齐、关系融合、冲突消解。

实体对齐：判断不同数据源中的实体是否指向同一个真实对象。例如，“Barack Obama”和“奥巴马”是同一人，“苹果公司”和“Apple Inc.”是同一组织。

关系融合：统一不同数据源中含义相同的关系。例如，“就职于”和“效力于”都表示“工作关联”，需统一为同一关系类型。

冲突消解：处理知识矛盾。例如，数据源 A 称“爱因斯坦出生于 1879 年”，数据源 B 称“出生于 1880 年”，需通过可信度分析（如数据源 A 为权威百科）选择正确信息。

3. 知识推理：生成新知识

知识推理是在已有知识图谱的基础上，通过逻辑规则或机器学习，自动发现隐藏的知识，补充知识图谱的完整性，核心方法包括规则推理、机器学习推理、多跳推理。

规则推理：基于人工或自动生成的逻辑规则，推导新关系。例如，规则“如果 A 是 B 的父亲，B 是 C 的父亲，则 A 是 C 的祖父”，可从（刘备，父亲，刘禅）、（刘禅，父亲，刘谌）推导出（刘备，祖父，刘谌）。

机器学习推理：利用图嵌入技术，通过实体向量的运算预测潜在关系。例如，模型通过学习大量“人物-出生于-地点”的三元组，发现“爱因斯坦”和“德国”的向量关联，预测出未明确标注的“出生于”关系。

多跳推理：通过多步关系链推导间接关联。例如，从（苹果公司，总部位于，库比蒂诺）、（库比蒂诺，属于，加利福尼亚州）、（加利福尼亚州，属于，美国），推导出（苹果公司，位于，美国）。

3.5.4　知识图谱系统的例子——“世界名人”迷你知识图谱

第一步：知识表示（构建图谱）。

知识图谱的核心是用三元组来表示知识，格式为：(头实体，关系，尾实体)。

我们为这个系统构建以下一系列三元组，它们共同构成了一个微型的知识网络，如表 3.1 所示。

表 3.1　三元组示例

实体	关系	实体（属性）	说明
史蒂夫·乔布斯	出生于	旧金山	关系：连接两个实体

续表

实体	关系	实体（属性）	说明
史蒂夫·乔布斯	就职于	苹果公司	关系：连接两个实体
史蒂夫·乔布斯	创建了	皮克斯	关系：连接两个实体
苹果公司	位于	库比蒂诺	关系：连接两个实体
库比蒂诺	位于	加利福尼亚州	关系：连接两个实体
加利福尼亚州	位于	美国	关系：连接两个实体
史蒂夫·乔布斯	姓名	"Steve Jobs"	描述实体的特征
史蒂夫·乔布斯	出生日期	"1955-02-24"	描述实体的特征
苹果公司	名称	"Apple Inc."	描述实体的特征
旧金山	类型	城市	描述实体的特征
美国	类型	国家	描述实体的特征

图 3.5 展示了这些三元组构成的微型知识图谱，清晰地揭示了实体间的复杂关系。

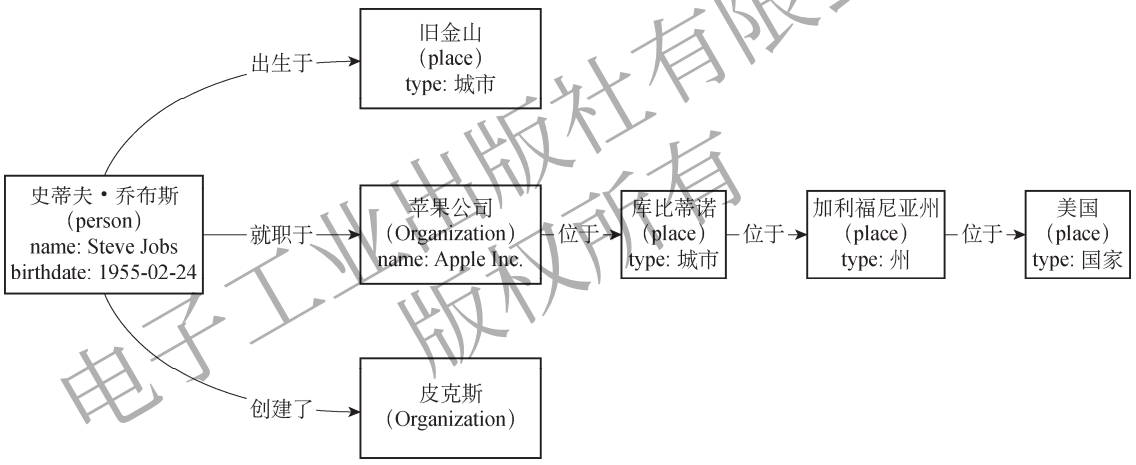

图 3.5　微型知识图谱

第二步：知识存储与查询（与系统对话）。

这个微型知识图谱被存储在一个支持图查询的数据库（如 Neo4j）中。我们可以用一种专门的查询语言（如 Cypher）来“问”它问题。

简单查询：“史蒂夫·乔布斯在哪里出生？”

查找：(史蒂夫·乔布斯) - [出生于] -> (? 地点)

答案：旧金山。

多跳查询（推理）：“史蒂夫·乔布斯在哪个国家工作？”

系统需要执行多步推理：

①找到乔布斯工作的公司 -> 苹果公司。

②找到苹果公司的位置 -> 库比蒂诺。

③找到库比蒂诺所在的州 -> 加利福尼亚州。

④找到加利福尼亚州所属的国家 -> 美国。

答案：美国（这个答案并没有直接存储在知识库中，而是通过关系链推理出来的！）。

路径查询：“找出从史蒂夫·乔布斯到美国的所有路径。”

系统会遍历所有可能的关系路径，并返回结果，例如，

乔布斯->就职于->苹果 ->位于->库比蒂诺->位于->加州->位于->美国

乔布斯->出生于->旧金山->位于->加州->位于->美国

【总结】这个简单例子体现了知识图谱的核心价值。

结构化：知识被清晰地组织成（实体-关系-实体）和（实体-属性-值）的形式。

关联性：不同的知识点通过关系连接成一个巨大的网络，而不是孤立的信息孤岛。

语义化：关系和实体的类型都有明确的含义（如“出生于”和“位于”含义完全不同），机器可以理解。

可推理：机器可以通过遍历关系网络，发现未被直接告知的隐藏知识（如乔布斯在美国工作），这是知识图谱最核心的价值。

这个微型知识图谱系统虽然简单，但放大亿万倍后，其基本原理就变成了Google、百度用来增强搜索的知识图谱，也成为了一切智能问答和推荐系统的基石。

练习题

第3章　练习

1. 请举例说明数据、信息和知识之间的区别与联系。

2. 评价一个知识表示方法的“可操作性”和“清晰性”原则。对于一个医疗诊断专家系统，这两个原则哪个更重要？为什么？

3. 用产生式表示法和框架表示法分别描述“智能手机”这个概念，并比较两种表示法的优劣。

4. 假设有以下规则库：

```
R1: IF 发烧 AND 咳嗽 THEN 可能患感冒
R2: IF 可能患感冒 AND 流鼻涕 THEN 很可能患流感
```

综合数据库初始事实为：{发烧， 咳嗽， 流鼻涕}。请模拟正向推理过程，写出每一步的推理结果。

5. 请为“学术会议”设计一个框架表示，要求包含至少4个槽（Slot），并为其中一个槽说明“过程依附”可能是什么。

6. 以“红楼梦”为主题，构建一个包含至少4个实体和3种关系的微型知识图谱，并用三元组形式列出。

7. 基于你为“红楼梦”构建的知识图谱，提出一个需要经过至少两步推理（多跳查询）才能回答的问题，并写出推理路径。

8. 在表示“如何烹饪一道菜”这类知识时，陈述性表示和过程性表示哪个更合适？为什么？

9. 知识图谱与传统的数据库相比，在支持智能应用（如智能搜索、推荐系统）方面具有哪些独特优势？

10. 产生式表示法在表示复杂、结构化的领域知识（如一台精密仪器的内部结构）时可能遇到什么困难？