# 第一部分 绪论

## 第二章 Agent的概念

理性的定义：Agent能做好的事情。2.2节

**PEAS分析**：P（性能度量）、E（环境）、A（执行者）、S（传感器）

Agent环境的维度：**完全（部分、不）可观察、单（多）Agent、确定（随机）、片段（延续）、静态（动态）、离散（连续）、已知（未知）。2.3节**

Agent按功能分类：

**简单反射Agent**：传感器获取环境的感知序列，Agent查询条件-动作规则库、效应器执行具体行动序列。**2.4.2节**

**基于模型的反射Agent**：（应对部分可观察环境）在简单反射的基础上增加了一个用来描述世界状态的模型，模型通过观察到的感知序列更新内部状态。**2.4.3节**

**基于目标的Agent**：（Agent只知道环境状态信息是不够用的）基于模型Agent，在执行动作前判断动作将产生的新状态，使用目标集来指导Agent执行指向目标的行动序列。**2.4.4节**

**基于效用的Agent**：（基于目标的Agent收益不一定好）基于模型Agent，不仅判断下一个动作产生的状态，也要评估效用，使用效用函数指导Agent执行最优效益（使期望效益最大化）的行动序列。**2.4.5节**

**学习型Agent**：（能自主学习的Agent）Agent内部有**性能元件、评判元件、学习元件、问题生成器**等；性能元件是一个完整的Agent（接收感知信息并决策）；学习元件负责改进提高；评判元件反馈评价Agent的行动；问题生成器负责得到新的和有信息的经验的行动提议（建议Agent去探索，改进自身）。**2.4.6节**

# 第二部分 问题求解

## 第三章 搜索

本章前提：世界是**确定的、可观察的、静态的和完全可知的。**

**问题的结构：初始状态、行动集、转移模型**（RESULT(s, a)）**、目标测试、路径耗散、状态空间，问题的解。3.2节实例**

树搜索：遍历访问

图搜索：引入探索集（已经探索过的结点），化图为树，避免冗余路径，可以是广度优先搜索。

无信息搜索（指除了问题条件本身外没有其他信息，无启发式）**3.4节**

**BFS、一致代价搜索**（引入了代价函数g(n)，每次拓展的是路径消耗最小的节点）**、DFS、深度受限搜索**（避免DFS的无穷路径，设置搜索深度L，存在无解情况）**、迭代加深搜索**（搜索深度L每次加一的DFS）**、双向搜索**（如果逆向有更小的分支）

有信息搜索（引入启发式函数h(n)）**3.5节**

一般最佳优先搜索（概念）：根据评估函数来拓展搜索结点。

**贪婪最佳优先搜索：拓展子结点中h(n)值最小的结点。3.5.1节**

**A\*算法：拓展子结点中f(n)=h(n)+g(n)值最小的结点。3.5.2节**

改进（优化了内存，但增加耗时）：递归最佳优先搜索（RBFS）、简单内存受限的A\*算法（SMA\*）

启发式函数：**3.6节**

有效分支因子b\*：在一部分问题集合上测出b\*的值，有利于探索启发式的总体实用性。好的b\*值接近1。

**松弛问题：减少原问题的行动限制，得到更容易求解的松弛问题，松弛问题的最优解的代价（比如步数）可以作为原问题的启发式。3.6.2节**

**模式数据库：把原问题划分成多个子问题（区别于松弛问题），保存子问题的解代价，该解代价作为原问题的启发式。**

## 第四章 搜索+

本章前提：世界是**连续的、不确定的、部分可观察的或者未知的**，在这种情况下搜索。（这里分情况讨论）

**局部搜索：4.1节**

**爬山法**：不断向值增加的方向移动，直到邻接状态中没有更好的解。（只关心邻接的状态，不需要在完整的树结构上搜索）。不足：容易陷入局部极大（小）值。

随机爬山法：按概率随机移动，直到找到比当前更好的子节点，增加找到全局最优解的可能。

随机重启爬山法：换多个随机的起点重试爬山法，增加找到全局最优解的可能。

模拟退火算法：大体上还是爬山法，区别在于慢慢降低移动的步调。

局部束搜索：多进程随机从不同起点状态开始，并行跑爬山法，同时进程间通信，交流找到的解。

**遗传算法**：状态集合内使用杂交、变异等手段，得到适合（需要设置**适应度函数**）的子状态，直到找到解。

**连续空间**的搜索：引入梯度计算，连续曲线（面）的梯度为0处有解。**4.2节**

**不确定环境**的搜索：引入**与或树**。在**或结点**选择行动，在**与结点**列出必须处理的所有可能后继状态。**4.3.2节**

**部分可观察环境**的搜索：引入信念状态

在信念状态空间中搜索解。**4.4.1节**

解是一个条件规划。**4.4.3节**

实例：迷宫定位（已知迷宫地图、Agent可观察到四个方向是否有障碍物，求解Agent当前位置）。**4.4.4节**

**未知环境**的搜索：引入联机搜索的概念（脱机是指计算出最后的解再行动；联机搜索则是实时地、计算与行动交替进行的搜索）

联机DFS、联机迭代加深搜索：Agent按照DFS或者迭代加深算法亲自实际地走一遍，把访问过的状态记录到表里。**4.5.2节**

联机局部搜索：引入随机行走算法（基本思路：Agent在探索时偏向于未尝试过的行动）；引入**实时学习A\*算法**（LRTA\*）**4.5.3节**

## 第五章 对抗搜索

前提：多Agent竞争

博弈的概念：有完整信息、确定的、轮流行动的零和游戏。**5.1节**

零和的概念：多个Agent之间的效用之和为零；

效用函数（作为启发式）：比如评定胜负、或者计算得分；

博弈树（上三角：MAX节点；下三角：MIN节点）；

**极大极小算法**：在博弈树上搜索（本质是深度优先）（**“挑选返回的最小值里最大的结点”**）。**5.2.1节**

**α-β剪枝算法**：提前发现不符合的子结点，不予以访问，起到剪枝的效果。**5.3节**

不完美的实时决策，极大极小算法的近似替代：**5.4节**

为了避免无限的搜索，应该今早截断搜索，所以引入**评估函数（EVAL）**作为启发式，有两种方法：1 用估计棋局效用值的评估函数代替效用函数；2 用**截断测试**取代目标测试，在截断测试时调用评估函数。

评估函数启发式的设计：1 EVAL对结果的排序（赢平输）与效用函数一致；2 EVAL的运算消耗低；3 对于非终止状态，评估函数应该和取胜几率（非终止状态下胜负还未确定，但是可以用剩下的棋子（兵马车）来估算剩余棋局赢面）密切相关。问题在于尽管当前评估效果很好，但也有可能最终导向失败。**5.4.1节**

截断搜索（简单版本）：设置搜索深度，达到制定深度即截断。**5.4.2节**

前向剪枝（α-β搜索的向前剪枝版本）：需要先验经验支持，用先验经验来评估是否剪枝。**5.4.3节**

用查表代替搜索：引入换位表概念（存储棋局的哈希表）**5.4.4节**

**随机环境**的博弈：抽卡、扔色子带来不确定性，引入**随机博弈**；博弈树中引入**机会结点、**计算极大极小值的时候引入**期望极大极小值。5.5节**

**部分可观察环境**的博弈：在信念状态中搜索博弈树。**5.6节**

## 第六章 约束满足问题

CSP基本概念：

状态：一组变量/值对；变量满足一些约束条件；

解：所有变量都赋予了满足约束条件的值的状态。

问题解法思路：识别违反约束的变量/值对的组合，消除空间搜索。

地图着色问题实例（引入了**约束图**）。**6.1.1节**

CSP的形式化：线性约束、多元约束、约束超图（密码算术问题）。**6.1.3节**

约束传播：推理的一种（区别于搜索）**6.2节**

结点相容：变量的值域的所有取值满足一元约束。

**弧相容**（Arc Compatible）：两个变量（弧）满足二元约束，也就是两个值域能完全对应。

**AC-3算法**：遍历所有弧，检查相容性，若改变了其他变量的值域，需要对相关的弧再入队列检查，当出现值域为空时无解，弧队列为空时有解**。6.2.2节**

路径相容：第三个变量总是有适合的取值来满足与前两个变量的每一种取值的相容。

k-相容：第k个变量总能与前k-1个变量都相容，则CSP就是k-相容的。

全局约束：（字面意思）用来全局地检测不相容的赋值。

**回溯搜索**（本质是DFS）：推理解决不了的CSP问题就用搜索解决。**6.3节**

状态：已经被赋值的变量集合

流程：为剩余变量赋值，不断尝试，检测到不相容就回溯，再换其他值尝试，直到找到解或者证明无解。

回溯搜索的启发式：**6.3.1节，最后一段可以帮助理解**

**度启发式**（**选第一个搜索结点**时用）：选择与其他未赋值的变量约束最多的变量；目的：降低未来搜索的分支因子。

**最少剩余值启发式（MRV）**（**选下一个搜索结点**时用）：选择可选取值最少（值域最小）的变量。（又叫最受约束变量算法、失败优先算法；算法目的：提前尝试最冒险的方式，减少回溯次数，起到剪枝效果）。

**最少约束值**（**为变量赋值**时用）：选择能给之后的变量留下更多选择（更少约束）的值；目的：选一个更无关的值，给剩下的变量更大的取值空间。

推理与搜索结合（交错进行）：推理的作用：通过前向检测，检查结点相容性，尽早发现并删除不相容的值，缩小值域，减小搜索的工作量。**6.3.2节**

前向检验：一个变量被赋值，相关的未赋值变量的值域要删去与其不相容的值。

维护弧相容（MAC）：（前向检验不够远）利用AC-3算法检查弧相容性。由AC-3算法反馈是否回溯。

**智能回溯**：**6.3.3节**

冲突集：导致某一个变量赋值失败的前置变量的赋值的集合。（Z赋值失败时，前面的X=x，Y=y都算到冲突集）

搜索中的智能回溯：直接回跳到**冲突集最后一次被赋值的变量**。（而不是回跳到上一个被赋值的变量）

冲突集使用：变量X取值X=x时，如果前向检验删除了变量Y值域中的值，那么X=x被记录到Y的冲突集中；当Y的值域为空（无解了），需要回溯到X，X吸收Y的冲突集（Y的冲突集可能来自多个路径，Y的值域可能被别的变量影响过，不仅仅是X）作为X自己的冲突集，X选其他值。若X值域为空，继续回溯。**6.3.3节**

**CSP局部搜索**：初始状态随机赋值，搜索过程是每次改变一个变量的值，早期必然存在一些违反约束的取值的情况，局部搜索的目的是解决这些冲突。**6.4节**

**最少冲突启发式**：为变量赋新值时，选择与其他变量冲突最少的值（可能一次赋值之后仍存在冲突，要继续搜索下去）。

CSP问题的结构：用结构化方式来表示问题**6.5节**

树状约束图（树结构CSP）：树的求解比带有环的约束图的求解效率高。可以直接沿着变量列表选择任意剩余值，无需回溯。

简化一般CSP为树结构CSP：

**基于环割集的算法**：将CSP的变量集分成一个环割集S（一组变量集合）与剩余的变量树，对S尝试所有变量的满足约束的所有赋值，同时删除剩余变量树的值域中不相容的值，若最后剩余的树还有解，则该解与S的赋值为原CSP的解。

**树分解**：把约束图分解为独立求解的子问题，子问题串成树CSP，新的树CSP的变量是每个子问题，变量的值域是子问题的所有取值集合。（如果一个子问题无解，就能提前知道原问题无解）

# 第三部分 知识、推理与规划

逻辑推理部分是7、8、9章，规划部分是10，11章，知识工程部分是12章

## 第七章 逻辑Agent

基于知识的Agent：在传感器效应器的基础上，维护一个知识库。**7.1节**

知识库（KB）：语句集合。

语句：表示了关于世界的断言，由知识表示语言来表示。

公理：初始时直接给出的语句，可以理解为Agent的背景知识，区别于后续推理的到的语句。

TELL任务：将新语句更新到知识库。

ASK任务：查询目前所知内容。

推理：从原有语句中推导出新语句。

基于知识的Agent的行动：

1 告知（TELL）知识库感知序列。

2 查询（ASK）相应的动作，伴随着知识的推理。

3 告知（TELL）知识库Agent所执行的行动，同时效应器执行行动。

Wumpus问题实例。**7.2节**

**逻辑。7.3节**

语法：为所有合法语句给出规范。

语义：语句的含义，定义了语句在可能世界的真值。

模型：所谓的“可能世界”，M(α)指α的全部模型。（一个语句包含多个符合条件的模型。**图7.5**）

**蕴涵关系：某语句逻辑上跟随另一个语句。**（α|=β，α蕴含β；α是比β更强的断言，α的模型更小更准确，排除了更多可能的世界。**前提为假的蕴含式都为真。**）

**模型检验**：通过枚举所有模型来检验KB为真时α都为真，证明M(KB)包含于M(α)。

导出：如果推理算法i可以根据KB到处α，记为KB |-*i* α

可靠（真值保持）：只导出蕴含句的推理算法。

完备：如果推理算法可以生成任一蕴涵句，则它是完备的。

（可靠和完备指的是推理算法的属性。）

推理过程在前提为真的任何世界中可以保证结论为真。

**命题逻辑： 7.4节**

**命题词（原子语句）：**每个命题词代表一个或真或假的命题。

**逻辑连接词：**与、或、非、蕴含、当且仅当。

知识库实例。**7.4.3节**

通过**模型检验**来判断蕴含：本质是枚举真值表，在KB为真的所有模型中，验证α是否全部为真。**7.4.4节**

**图7.10 TT-Entail（真值表推理）算法**代码解读：

输入KB、α，提取二者所有的命题，每个命题都有真、假两个值，与CSP回溯算法类似，每次选一个命题赋值，在全部赋值过程结束后（此时得到一个模型，总共有2n个模型），先后判断该模型中KB的真值和α的真值，KB为真且α为假时，原蕴含式不成立。KB与α的其余情况则成立。

**命题逻辑定理证明：7.5节**

通过**定理证明**来判断蕴含：在KB的所有语句上直接应用推理规则以构建目标语句的证明。无需关注模型。

逻辑等价：如果语句α、β在同样的模型集合中为真，则它们逻辑等价。

有效性：如果一个语句在所有模型中都为真，那么它是有效的。

可满足性：如果一个语句在某些模型中为真，那么它是可满足的。（枚举模型集，找到一个为真，则语句满足可满足性）

（有效性和可满足性指的是语句的属性，跟前面的可靠性、完备性不同）

**有效性与可满足性的关联：α有效当且仅当┐α不可满足。同理“α|=β有效”当且仅当“(α∧┐β)不可满足”**

SAT问题：命题逻辑语句的可满足性判定。

推导与证明。**7.5.1节**

假言推理规则：给定语句(α⇒β, α)，推出语句β。

消去合取词：给定语句α∧β，推导出语句α、β。

单调性：逻辑蕴含语句集会随着添加到知识库的信息的增长而增长。

归结证明：推理规则的一种。**7.5.2节**

单元归结：给定(L1∨L2∨…∨Lk, m)，其中m与Li是互补文字，推出语句 L1∨…∨Li-1∨Li+1…∨Lk（从原来的析取式中去掉了Li）

全归结：给定(L1∨…∨Lk, M1∨…∨Mn)，推出二者的析取式，其中互补的文字被去掉。

注：归结理解为把给定的两个析取式合并，再去掉其中互补的文字。结果子句中，每个文字只能出现一次，消去文字多余副本的过程叫归并。

合取范式（CNF）：以子句的合取式表达的语句（语句是子句的合取，最外层是合取，内层是析取）。

**归结算法：通过推理来证明蕴含关系**

**思路：为了证明α╞β成立，用反证法，需要证明α∧¬β是不可满足的。**

**1 将(α∧¬β)展开转化成CNF。**

**2 对结果子句运用归结规则，对含有互补文字的子句进行归结产生新的子句，如果该新的子句没有出现过就将其加入子句集中。**

**3 循环1，2直到：**

**没有可以添加的新语句，则α不蕴含β**

**存在两个子句能归结出空子句（因为产生了值为假的CNF子句，则原式值为假），则α蕴含β**

Horn子句和限定子句。**7.5.3节**

限定子句：析取式（CNF子句）只含一个正文字，其他文字都是负文字。

目标子句：析取式没有正文字。

Horn子句：析取式至多只有一个正文字。（封闭性：两个Horn子句归结，结果还是Horn子句）。

Horn子句句型中，前提称为体，而结论称为头。只包含一个正文字的语句称为事实。

使用只包含限定子句的知识库的理由：

1 限定子句可以转换成蕴涵式α∧β⇒γ——前提为正文字合取式，结论为单个正文字，便于理解。

2 方便使用后面提到的前向链接、后向链接算法。

3 Horn子句判定蕴涵的运行时间与知识库规模呈线性。

**前向和反向链接**（在Horn知识库的基础上）**。7.5.4节**

**前向链接**：从知识库中的已知事实(正文字)开始。如果蕴涵的所有前提已知，那么把它的结论加到已知事实集。继续这一过程，直到查询q被添加或者直到无法进行更进一步的推理。前向链接的**运行时间是线性的**。前向链接是**数据驱动推理**的实例——即推理是从已知数据开始的。

**与或图**。**图7.16**

对于Horn KB，前向链接算法满足：

可靠的：每个推理本质上是假言推理规则的一个应用。

完备的：每个被蕴涵的原子语句都可以推导得出。

**反向链接**：从查询开始进行推理。如果查询q已知为真,那么无需进行任何操作。否则,寻找知识库中那些以q为结论的蕴涵式。如果其中某个蕴含式的所有前提都能证明为真（通过反向链接），则q为真。反向链接的时间复杂度也是线性的。反向链接是一种**目标制导的推理**形式。

有效的命题逻辑模型检验（详细）。**7.6节**

一个逻辑语句包含多个符合条件的模型，模型是被部分赋值的逻辑语句。

基于模型校验的命题推理的两类有效算法：1 回溯算法，2 基于爬山法（联想到CSP的两种解法）

**回溯算法（DPLL, Davis-Putnam）**：类似CSP回溯搜索算法和模型校验算法，输入CNF，返回语句的真值。（本质是对可能模型的递归优先深度枚举算法，回忆CSP回溯搜索的解法，把语句的文字当作变量挨个赋真值）

局部搜索：证明输入语句是可满足的，只要返回一个模型即可，这里用类似CSP局部搜索算法求解。在为子句中选择一个命题符号进行真值翻转时，有两种方式：1 最小化新状态下未满足的语句的数量；2 随机行走。

基于命题逻辑的Agent。**7.7节**

1 Agent在给定感知历史信息的情况下尽可能**对世界状态进行推理**；2 Agent如何在不回头检查每一个推理的感知历史的情况下有效**跟踪世界的变化**；3 Agent如何使用逻辑推理来构造确保**完成目标的规划**。

引入时间概念（流、非时序变量）

后继状态公理：t时刻的行动使F在t+1时刻为真；或F在t时刻已经为真且t时刻的行动没有使F为假。公式：Ft+1 ⇿ ActionCausesFt ∨ (Ft ∧ **┐**ActionCausesNotFt)

逻辑状态评估：逻辑状态评估涉及的是维护与观察历史一致的描述可能状态集的逻辑语句。每一更新步骤要求使用环境的转移模型，转移模型可以从后继状态公理构建。

逻辑Agent的决策可以通过SAT求解进行：找出模型规范到达目标的未来可能的行动序列。这种方法只适用于完全可观察或无传感信息的环境。

命题逻辑无法扩展到无限的环境，原因是它缺之足够的表达能力来准确地处理时间、空间以及对象间关系的模式。

## 第八章 一阶逻辑（概念介绍）

一阶逻辑比命题逻辑有更强的表达能力，引入对象、关系、本体论概念。

一阶逻辑：世界由对象构成，对象之间的某种关系或成立或不成立。

语法语义。**8.2节**

模型：对象、对象之间的关系（元组集合）、解释（这个详见下一节）。**8.2.1节**

符号和解释：**8.2.2节**

常量符号：表示对象；

谓词符号：表示关系，输出的是真值；

函词：表示函数，输出的是对象。

解释：解释将常量映射到对象、谓词映射到函数间的关系、函词映射到对象上的函数。

项：代指对象的逻辑表达式。（常量符号、函词+常量符号）

原子语句：由谓词以及随后括起来的列表项组成。

复合语句：在原子语句的基础上用上了逻辑连接词。

量词：全称量词、存在量词。

等词：逻辑“=”号。

一阶逻辑的运用。**8.3节**

断言：被TELL到知识库的语句。

查询、目标：用ASK提出的问题，被知识库逻辑蕴含的任何查询都肯定可以得到回答。

量化查询（量词引入了变量）：查询的到的结果是一个变量的赋值（置换、绑定表）。

公理（最基本的信息）、定理（由公理推导得到的）。

## 第九章 一阶逻辑的推理

量词的推理规则，把一阶逻辑退化成不含量词的命题逻辑。**9.1节**

全称量词实例化（UI）：置换（substitution），SUBST(θ, α)表示把置换θ（形式为{x/X, y/Y}）应用到语句α。

存在量词实例化：Skolem化过程，用一个从未出现过的常量符号代替变元。存在量词实例化只能应用一次，通过常量生成实例后抛弃**。**

全称量化语句也可以被所有可能的实例化集代替：把知识库词汇表中所有可能的基项全部代入全称量化语句，得到语句集合，然后抛弃全称量化语句。

**合一和算法提升**（把一阶逻辑退化成命题逻辑再推理效率太低，所以下面直接进行一阶逻辑的推理）**。9.2节**

**一般化假言推理规则：**置换θ能应用于蕴含式所有的前提pi，推出置换θ能应用于蕴含式结论q**。**

**合一（unify）：输入两个语句，输出它们共用的置换。**（若Subst(θ,p)= Subst(θ,q)，则unify(p,q)=θ）。**9.2.2节**

标准化分离：原始语句用了共同的变量命名，需要解决变量命名冲突。

最一般合一置换（MGU）：从合一的结果来判断，存在多个具体的置换结果时，MGU指的是对变量取值限制最少的置换（更通用，更抽象，甚至还保留变量）。

求解MGU的算法：并排遍历两个表达式，同时建立置换，当结构中两个对应点不匹配，则失败。

存储（Store）与检索（Fetch）：是TELL与ASK更底层的函数。**9.2.3节**

Store(s)：将语句s存到知识库；

Fetch(q)：返回所有合一置换，这些合一置换能使查询q与知识库的语句合一。

求解过程：将知识库中所有事实存到一个列表中，把查询q和表中每一个元素进行合一。

提升Fetch的优化：建立索引，避免无意义的合一。比如谓词索引——同一个谓词算作一个存储桶。

**前向链接：9.3节**

简单前向链接算法：从已知事实出发，触发所有前提得到满足的规则，然后把这些规则的结论加入到已知的事实中。每次迭代都会进行合一。重复，直到查询到结果（有解）或是没有新事实加入（无解）。**9.3.2节**

高效的前向链接：9.3.3节

存在的问题：

1 算法寻找所有可能的合一置换，把规则的前提与知识库中合适事实集进行合一，过程代价昂贵；

2 算法每次迭代都要对所有规则重新进行检查，确定其前提是否得到满足，效率低下；

3 算法有可能生成许多与目标无关的事实。

对于已知事实的规则匹配：对前提的合取项进行排序，尽早缩小检索规模，减少遍历开销。启发式：类似CSP的MRV启发式，优先选最少置换数的合取项进行合一。

增量前向链接：算法在第t-1次迭代得到新事实，就能直接用于第t次迭代中语句合取项的合一。但是，剩下的合取项还是要跟先前的事实匹配的。这样是为了减少重复检索。对于一次迭代中合一出来的不完全匹配的语句，应该保留并逐步完成。

拓展：RETE算法，用空间换时间提高运行效率。

避免无关的推理：**采用反向链接**；把前向链接限制在一个选定的规则子集内；引入演绎数据库，利用目标信息重写规则集，从而在前向推理过程中只考虑相关变量的绑定。

**反向链接：**从目标开始反向推导链接规则，直到找到支持证明的已知事实。**9.4节**

数据结构：目标列表（就是个栈）。

返回：满足查询的所有置换的集合。

流程：

在目标列表选取目标；

在KB中遍历，寻找正文字能与目标合一的所有子句；

每个子句的前提被加入目标列表，创建一个新的递归反向链接调用；

（**已知事实**：只有头（正文字），没有体（前提）的子句）

当目标和某个**已知事实**合一时，没有新的子目标加入目标列表，该目标得到解决。

如果当前所有目标都能满足，则当前的证明分支是正确的。

逻辑程序知识，介绍**Prolog**。**9.4.2-9.4.6节**

语句A∧B⇨C，表达为C:=A, B.

[E|L]表示列表，E表示表头，L表示剩余部分。

append(A, B, C)拼接列表A、B得到C，参数都是列表类型。

一阶逻辑归结 **9.5节**

**归结的前提是语句为CNF；**

一阶逻辑语句转换成CNF过程：**消除蕴含词**；**将量词前面的取反词内移**；**变量标准化**（避免变量冲突）；**Skolem化**（消除存在量词，使用Skolem函数代替直接常量赋值）；**直接删除全称量词**；**分配合取符号，转成CNF**。

归结推理：反证KB∧┐α是不可满足的，特别注意，与命题逻辑归结推理有区别，一阶逻辑需要在归结的同时进行合一置换。**9.5.2-9.5.3节**

关于**等词断言**的推理，当语句中出现等词语句的时候的推理算法，可以应用到数学推理 **9.5.5节**

Sub(x, y, m)表示用单元子句y置换语句m中所有单元子句x的位置。

**1 解调** 有x=y，α，用y置换α的x可以得到新的子句。y只需能与α的x合一即可，不必完全等于x。

解调公式：给定(x=y, M1∨…∨Mn)，其中x与文字Mi中的z可以合一得到置换θ（Unify(x, z)=θ），推出Sub(Subst(θ,x), Subst(θ,y), M1∨…∨Mn)。

**2调解**公式：给定(L1∨…∨Ln∨x=y, M1∨…∨Mn)，推出Sub(Subst(θ,x), Subst(θ,y), Subst(θ, L1∨…∨Ln∨x=y, M1∨…∨Mn))。

例子：P(F(x, B), x)∨Q(x), F(A, y)=y∨R(y), θ=UNIFY(F(A, y), F(x, B))={x/B, y/A}，解：P(B, A)∨Q(A)∨R(B)

3 等词合一（用交换率结合律等特殊属性求解）。

归结算法优化：**9.5.6节**

单元优先：优先对那些包含单文字的语句进行归结，试图尽快归结出空子句；单元归结：限制归结过程每次都必须包含单元子句。

引入支撑集（一个特殊子句集）；

输入归结：每次归结都是输入语句（来自KB或查询）与某个语句的结合，这样树的结构是单一的骨干型的，空间更小；

包容法：如果存相似语义的语句（如P(A)，P(x)），清除更特例的，保留更通用的。

## 第十章 经典规划

前提：确定的、完全可观察的、静态的环境。

经典规划概念定义：**10.1节**

要素化表示：用一组变量表示世界的一个状态的表示方法。

PDDL（Planning Domain Definition Language）。

状态是**流**（等同于状态变量）**的合取**；流是基元的、不含函数。

At(a, B)，Has(c, D) 等等。

动作模式：动作名；动作的变量列表；前提；效果；适用的实例对象；

Action(***A(x1, x2, …, xn),***

Precond**:** ***X****∧****Y****∧****…****∧****Z***

Effect**: *I****∧****J****∧****…****∧****K***

)

状态迭代过程：在状态s执行动作a后得到新状态s’，s‘由s删除a效果中的负文字，添加a效果中的正文字得到。

规划问题定义成为搜索问题：从**初始状态s0**出发，沿着**Actions**、**Result(s, a)**函数，到达**目标测试**。

**PDDL问题命题化&实例：图10.1、10.1.1-10.1.3节**

初始状态（Init(X∧Y∧…∧Z)）、目标状态（Goal(I∧J∧…∧K)）、动作模式、解（动作的序列）；

状态空间搜索规划。**10.2节**

规划问题的描述定义了一个搜索问题：

**前向搜索**：从初始状态出发，搜索状态空间，找到目标，缺点在于：1 容易搜索到无关的动作，2 状态空间大，搜索计算量大；

**后向搜索**：PDDL允许算法从目标状态后向搜索初始状态；

**启发式**：松弛问题的解代价 **3.6.3节**

**松弛问题思路**：把搜索问题看作图，节点为状态，边为动作，解为初始状态到目标状态的路径。把问题松弛化：1 增加更多的边，使求解更容易；或者，2 把一些点集中到一组，减少状态数量。

（增加边 1）忽略前提启发式：忽略前提，每个状态都可以使用全部动作，意味着松弛问题的步数与剩余未满足目标数相近；

（增加边 2）忽略删除列表：假设所有目标和前提只含有正文字（在松弛问题中不考虑某个动作会抵消另一个动作的效果）；

（减状态）状态抽象：分解问题，仅关注与子问题相关的几个流，减少目标状态流的数量。例子：飞机拉货

模式数据库；

规划图。**10.3节**

适用：规划图只能用于没有变量的规划问题（比如命题规划）。

规划图结构：

**层次**：状态层和动作层交替排列（S0, A0, S1, A1,…）；

**单个文字**：表示状态（Have(A)，At(A, B) 等等）；

**方格**：方格内的文字表示具体动作，空的方格表示空动作，即保持状态不变；

**连线**：连接状态与动作，表示前提和效果；

**弧线**：表示互斥关系。

互斥关系（存在于状态和动作中），互斥意味着不可能同时发生（提供一种约束）。互斥关系满足条件：

状态之间的互斥：互为负文字，或者得到文字的动作是互斥的。

**动作之间的互斥**（满足其一）：

1 不一致效果（两个动作效果互相否定）：一个动作否定另一个动作的效果；

2 冲突（两个动作前提与效果否定）：一个动作的效果之一是另一个动作的前提的否定；

3 竞争（两个动作前提互相否定）：一个动作的前提之一与其他动作的一个前提互斥。

规划图建立过程：从S0开始，构造满足前提的动作层，生成新的状态层，直至连续两层一模一样（即规划图达到稳定）。如果目标状态的文字没有在图的最后一层出现，那么问题无解。求解规划（动作序列）的过程就是从S0开始的搜索过程。

带启发式的规划图：**10.3.1节**

层次代价：初始状态S0到目标文字Gi首次出现层Si的层数。

序列化规划图：（规划图允许每层有多个动作，启发式只关心层数而非动作数，层次代价不总是准确）序列化通过互斥约束了一次只有一个动作，层次代价更准确。

最大层启发式 取目标的最大层次代价（可采纳，但不准确）；

层次和启发式 取所有目标的层次代价之和；

集合层次启发式 找到合取目标中所有文字出现的层，在这一层没有任何一对目标文字是互斥的。

**Graphplan算法**：从规划图中抽取出一个规划。**10.3.2节**

**过程**：输入problem（包含目标goal），重复进行规划图拓展（每次迭代Expand-Graph过程会为规划图添加新的状态层），当所有目标文字在同一状态层出现且没有互斥（意味着出现一个可能的解），则调用Extract-Solution搜索规划图中问题的解。若无解，则继续迭代拓展新的层。当没有理由继续时，以失败终止过程。

案例细节：

初始层S0标出初始状态中所有正的流（文字）和所有状态的负流。

拓展新层时需要标注互斥。

关于Extract-Solution算法：

1 **布尔CSP问题解法**，其变量是每层的动作，变量值域为动作在规划内还是规划外，约束是互斥关系以及满足全部目标和前提；

2 **后向搜索**，从最后一层Sn的目标状态集合开始，每个Si层选择Ai-1层中其中一个无冲突的、效果覆盖目标状态的动作集，新的目标集合是这些动作的前提，位于Si-1层。最终的目的是达到S0且所有目标状态得到满足。（后向搜索的启发式：基于文字层次代价的贪婪算法，优先选择最高层次代价的文字，优先选择前提的层次代价和最小的动作）

**Graphplan**的终止：当问题无解，则算法终止并返回失败（前面只介绍了成功的情况）

规划图的稳定：只意味着暂时两层状态一样，并不意味着当前就有问题的解；

no-good标识：当在给定层的目标集中求解失败时，将层数和仍未被满足的目标集合{level, goals}记录为no-good。

终止的条件：在建立规划图时，一旦图和no-good都达到稳定，且没有解，则以失败终止。

其他经典规划：转换为布尔满足性问题（SATPlan，用命题逻辑的解法求解）；转换为一阶逻辑推理规划（情景演算）；转换为约束满足问题；转换为偏序规划。**10.4节**

偏序规划：由一个动作集合与一个约束集合组成，约束表达了动作的发生顺序。**10.4.4节**

偏序规划图建立过程：1 从仅包含初始状态和目标状态的空规划开始，中间没有动作；2 搜索规划中的缺陷（检查有没有满足目标状态或者某个动作的前提）；3 加入动作，纠正当前缺陷（如果不能纠正，则回溯，尝试其他动作）；4 循环2，3直到找到解。

## 第十一章 现实世界的规划与行动

资源和调度 **11.1节**

现实中需要考虑资源约束和时序，把问题分解成**规划阶段**和**调度阶段**。规划阶段选择动作，考虑次序约束，满足问题的目标；调度阶段把时间信息加入到规划中以满足资源和期限约束。调度问题为一组工作，工作为一组给定时序、耗时以及资源消耗的动作。

调度问题求解实例。**11.1.2节**

**关键路径方法**：（先不考虑资源约束，仅考虑最小化完成时间）将次序约束视为与动作相关的有向图。构造偏序规划图，图中时间最长的规划路径的就是关键路径。

调度问题需要求解所有动作的ES和LS构成。动作的松弛=LS-ES。关键路径上没有松弛。

引入资源约束后调度的解法：首先需要偏序规划图。在时间轴的基础上列出资源约束，把所有路径都编排上去。

最小松弛启发式：每次迭代，考虑前置动作已被调度而自身未被调度的动作，并且调度具有最小松弛的那个动作作为最早可能开始的。迭代的同时更新那些收到影响的动作的ES、LS。如此反复。（有效率，但未必是最优解）

**分层规划 11.2节**

高层动作

层次任务网络（HTN）

基元动作：具有标准的前提-效果模式，是完全可观察、确定的一组可用动作（不能再细化）。

高层动作（HLA）：有一个或多个细化动作序列，细化动作可以是一个HLA（可嵌套）或者基元动作。

HLA的实现过程：HLA细化到只有基元动作。

高层规划的实现：每个HLA的实现的拼接。HLA能否到达目标需要从基元动作的前提-效果中判断。

给定一个状态，当高层规划的至少一个实现能从状态达到目标，则规划有解。求解方法：

1 在高层规划的多个实现中**搜索**一个有效的实现；

or 2 直接对HLA**推理**。

搜索基元解：**11.2.2节**

分层搜索算法（本质是对规划的BFS）：反复在当前规划中选择一个HLA，用它的细化代替它，直到规划达到目标。

规划库：存有复杂高层次动作的已知实现。通过学习来构建规划库。

搜索抽象解：**11.2.3节**

可到达集：给定状态s，一个HLA的到达集h，HLA任一实现可到达的状态集合（Reach(s,h)）。HLA序列的可达集是它们各自可达集的并集。

如果一个高层规划（HLA序列）的可达集与目标状态集有交集，那么规划可能达到目标，算法将致力于那个规划，对它进一步细化。

乐观集和悲观集。

非确定性领域中的规划与行动（前提：部分可观察、非确定的、未知的环境）。**11.3节**

Agent在规划过程中需要对它将获得的感知进行推理，需要引入感知模式（Percept）来拓展PDDL。例子：感知物体和颜料罐的颜色。

无传感器规划（无观察环境）：搜索信念状态、物理转移模型、条件效果、近似算法、启发式算法；**11.3.1节**

应急规划（部分可观察、非确定性环境）：基于感知的条件分支的规划。**11.3.2节**

应急规划Agent保持信念状态为一个逻辑公式，存在一个条件公式时，通过确定这个信念状态是否蕴涵这个条件公式或否定来评估每个分支条件。

在线重规划（未知环境）：当Agent遇到应急事件时，需要确定是否需要一个新的规划。**11.3.3节**

需要重规划的条件：缺失前提、缺失效果、缺失状态变量；

按照对环境监控的程度分：

1 动作监控：Agent验证所有前提是否仍然成立；

2 规划监控：Agent验证剩下的规划是否还会成功；

3 目标规划：Agent检查是否有它可以尝试达到的更好的目标集。

多Agent规划（多效应器规划、多体规划、分散规划）：**11.4节**

多同步动作的规划：

转移模型：引入联合动作（a1…an）代替单一动作a，那么转移模型分支因子从b变成b的n次方。

松耦合问题解思路：假装Agent之间是完全分离的，然后再进行交互。降低复杂度。

并发动作列表：在PDDL动作模式中拓展Concurrent，说明哪些动作必须并发地执行或不能并发执行，这是一种约束。

合作与协调：

协定：一个协定是对联合规划进行选择的任何约束。（共识，例如驾驶车辆靠右行驶）

没有协定的时候就用即时通讯来完成联合规划。

## 第十二章 知识表示

**抽象的类与具体的物体的逻辑推理**

本体论工程 **12.1节**

**类别与对象12.2节**

类别物化得到对象；类的包含和继承关系、子类关系、分类系统；类的不相交关系；

PartOf(X, Y)函数：参数是两个具体的物体对象，表示一个物体X是另一个物体Y的部分；该关系满足可传递性、自反性；

BunchOf({X, Y, Z})函数：参数是一个对象集合，得到复合对象；

度量，单位函数。例子：Length(X) = **Inches(1.5)** = **Centimeters(3.81)**。可以使用等号，大于小于号表示逻辑；

物体（thing）和物质（stuff）、（非）固有属性；

事件：**12.3节**

事件演算（相对于情景演算，它的动作是离散的，瞬间的，一次发生一个动作，例如装一盆水，状态只有空和满）：基于时间点而不是基于情景，动作是连续的。事件演算对流和对象进行物化。

流对象：表明状态事实的对象，对标状态变量。谓词T(F, t)用于表示流F在t时间点为真；

事件对象：事件类别的实例，例如**E∈Flying(John, A, B)** ，跟动作有区别。用Happens(E, i)表示事件E在时间区间i(t1, t2)内发生；

**启动**：Initiates(E, F, t)函数表示事件E使流F从时间点t开始成立；

**终止**：Terminates(E, F, t)函数表示E使F从t开始不再成立；

**剪切**：Clipped(F, i)函数表示F在i内的时间点开始不再为真；

**恢复**：Restored(F, i)函数表示F在i内的时间点开始恢复为真；

时间谓词：前后（before，after）、相接（meet）、涵盖（during）、交错（overlap）、共同开始（start）、共同结束（finish）、相等（equal）

流和对象，例子：美国总统**President(USA)**需要有时间说明；

精神事件和精神对象（人的信念）：引入命题态度（相信、知道、想要、打算、通知等）。**12.4节**

可能世界、可达性关系

**类别推理**系统：基于类别的组织和推理特别设计的系统。**12.5节**

语义网络图：为知识库可视化提供**图形**帮助，并在类别隶属关系基础上，为推断对象的属性提供有效算法；

一阶逻辑易于描述关于对象的状态（量词、谓词、函数），**描述逻辑**是易于描述类别的定义和属性的符号表示，**Classic语言**。

**缺省信息推理**：**12.6节**

限定：指定被假设为“尽可能错”的特殊谓词——除了已知为真的对象之外的都为假。描述鸟能飞的例子：Bird(x)∧┐Abnormal(x) ⇒Flies(x)，这里引入的谓词Abnormal()被假设尽可能错，它是被限定的。

模型偏好逻辑：如果一个语句在知识库中所有偏好模型都为真，那么它是被蕴含的。对于限定，如果一个模型有更少的反常对象（或真值原子），那么它对于另一个而言是被偏好的。少的那一部分是偏好的。

缺省逻辑：一种形式化方法，比如鸟会飞的例子：Bird(x): Flies(x)/Flies(x)。缺省规则：P: J1, …, Jn / C（P先决条件，C结论，Ji准则，Ji中存在一个为假，就不能得到结论）**12.6.1节**

**真值维护系统（TMS）**：**12.6.2节**

信念修正：（知识表示系统得到的推论只有缺省情况，而不是绝对的确定）某些推论而来的事实最后发现可能是错误的，将不得不在新的信息面前撤销。如果发现过去TELL到知识库的语句P被否定，那么需要从KB中撤销语句P，以及从P推断出的所有语句（P⇒Q）。若存在语句Q，同时有P⇒Q、R⇒Q，Q是不必须被撤销的。TMS用于处理这种情况。

**真值维护的方法：**

最简单的方法：对语句P编号（P1…Pn）。记录语句被TELL给KB的顺序，当要撤销Pi，系统恢复到Pi被添加之前的状态，删除Pi及从Pi得到的推论。之后重新添加Pi+1…Pn。但是代价大。

更有效的方法：基于准则的真值维护系统（JTMS）。知识库的每一条语句用一个由推理出他的语句集组成的准则来注释，在撤销时删除对应的准则。

基于假设的真值维护系统（ATMS）：被设计用来使假设世界之间的这类上下文切换变得高效。ATMS在同一时刻表示已经被考虑的所有状态（假设）。每个语句有一个用一组假设集组成的标记。语句只有在一个假设集中的全部假设都成立时才成立。