计算机科学188：人工智能导论
2024年春季 笔记9

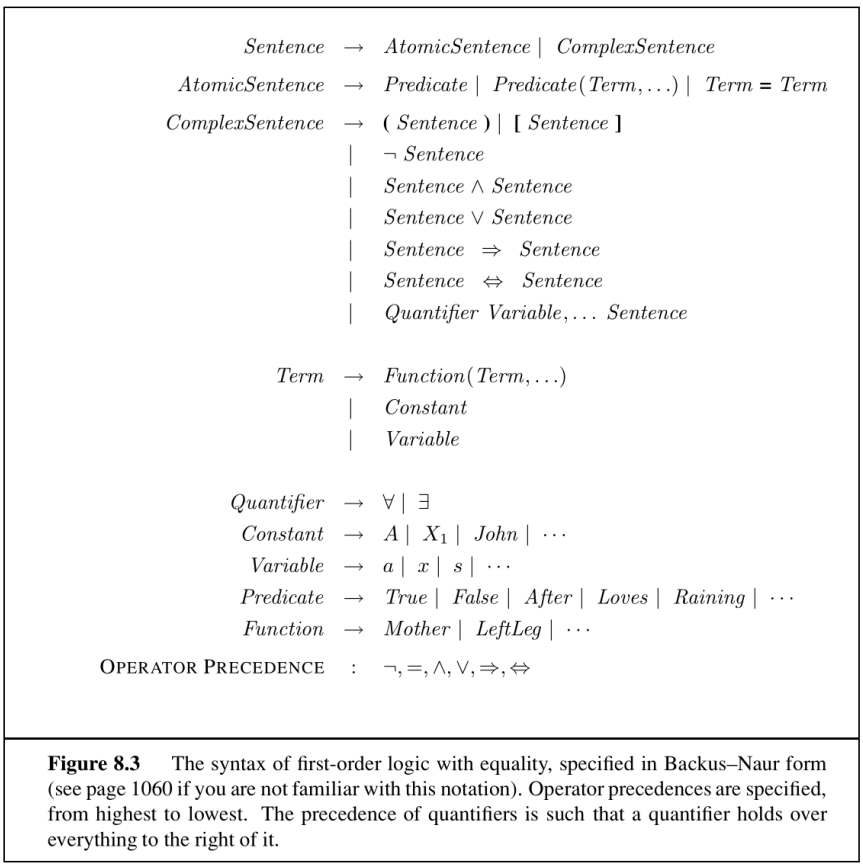
作者（其他所有笔记）：尼基尔·夏尔马
作者（贝叶斯网络笔记）：乔希·胡格和杰基·梁，由王蕾吉娜编辑
作者（逻辑笔记）：亨利·朱，由考佩林编辑
致谢（机器学习与逻辑笔记）：部分章节改编自教材《人工智能：一种现代方法》。

最后更新时间：2023年8月26日

一阶逻辑

逻辑的第二种形式，即一阶逻辑（FOL），比命题逻辑更具表现力，并将对象作为其基本组成部分。使用一阶逻辑，我们可以描述对象之间的关系，并对它们应用函数。每个对象由一个常量符号表示，每个关系由一个谓词符号表示，每个函数由一个函数符号表示。

下表总结了一阶逻辑语法。



$$
\begin{array}{rcl}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} & \rightarrow & (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
& \mid & \neg\ \textit{Sentence} \\
& \mid & \textit{Sentence} \wedge \textit{Sentence} \\
& \mid & \textit{Sentence} \vee \textit{Sentence} \\
& \mid & \textit{Sentence} \Rightarrow \textit{Sentence} \\
& \mid & \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
& \mid & \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& \mid & \textit{Constant} \\
& \mid & \textit{Variable} \\
\\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Figure 8.3 The syntax of first-order logic with equality, specified in Backus–Naur form (see page 1060 if you are not familiar with this notation). Operator precedences are specified, from highest to lowest. The precedence of quantifiers is such that a quantifier holds over everything to the right of it.

一阶逻辑中的项是指称对象的逻辑表达式。最简单的项形式是常量符号。然而，我们不想为每个可能的对象定义不同的常量符号。例如，如果我们想提及约翰的左腿和理查德的左腿，我们可以通过使用函数符号，如Leftleg(约翰)和Leftleg(理查德)来做到这一点。函数符号只是命名对象的另一种方式，而不是实际的函数。

一阶逻辑中的原子语句是对对象之间关系的描述，如果关系成立则为真。原子语句的一个例子是Brother(约翰,理查德)，它由一个谓词符号后跟括号内的一列项组成。一阶逻辑的复杂语句类似于命题逻辑中的语句，是由逻辑连接词连接的原子语句。

自然地，我们希望有描述整个对象集合的方法。为此我们使用量词。全称量词 $\forall$ 的意思是“对于所有”，存在量词 $\exists$ 的意思是“存在”。

例如，如果我们世界中的对象集是所有辩论的集合，句子 $\forall a$, TwoSides (a) 可以翻译为“每场辩论都有两个方面”。如果我们世界中的对象集是人，句子 $\forall x, \exists y$, SoulMate(x,y) 将意味着“对于所有人来说，都有某个人是他们的灵魂伴侣”。匿名变量 a, x, y 是对象的替身，可以用实际对象替换，例如，将劳拉代入 x 到我们的第二个例子中，将得到一个陈述“有某个人是劳拉的灵魂伴侣”。
全称量词和存在量词分别是对所有对象表达合取或析取的便捷方式。因此，它们也遵循德摩根定律（注意类似的关系

between conjunctions and disjunctions):

$$\begin{array}{lll} \forall x \ \neg P \equiv \neg\exists x \ P & & \neg(P \vee Q) \equiv \neg P \wedge \neg Q \\ \neg\forall x \ P \equiv \exists x \ \neg P & & \neg(P \wedge Q) \equiv \neg P \vee \neg Q \\ \forall x \ P \equiv \neg\exists x \ \neg P & & P \wedge Q \equiv \neg(\neg P \vee \neg Q) \\ \exists x \ P \equiv \neg\forall x \ \neg P & & P \vee Q \equiv \neg(\neg P \wedge \neg Q) \end{array}$$

最后，我们使用等号来表示两个符号指代同一个对象。例如，令人难以置信的句子（妻子（爱因斯坦） = 第一代堂表亲（爱因斯坦） ∧ 妻子（爱因斯坦） = 第二代堂表亲（爱因斯坦））是真的！

与命题逻辑不同，在命题逻辑中，一个模型是对所有命题符号赋予真或假的赋值，而在一阶逻辑中，一个模型是将所有常量符号映射到对象、谓词符号映射到对象之间的关系以及函数符号映射到对象的函数。如果句子所描述的关系在该映射下为真，那么这个句子在一个模型下就是真的。虽然命题逻辑系统的模型数量总是有限的，但如果对象的数量不受限制，一阶逻辑系统可能有无限多个模型。

这两种逻辑方言使我们能够以不同方式描述和思考世界。在命题逻辑中，我们将世界建模为一组取值为真或假的符号。在这种假设下，我们可以将一个可能世界表示为一个向量，每个符号对应一个1或0。这种世界的二元视图就是所谓的因式分解表示。在一阶逻辑中，我们的世界由相互关联的对象组成。这种世界的第二种面向对象视图被称为结构化表示，在许多方面更具表现力，并且与我们自然用来谈论世界的语言更紧密地对齐。

一阶逻辑推理

使用一阶逻辑，我们以完全相同的方式制定推理。我们想弄清楚是否 $KB \models q$ ，也就是说，在所有使 KB 为真的模型中 q 是否为真。找到解决方案的一种方法是命题化，即将问题转化为命题逻辑，以便可以使用我们已经阐述的技术来解决。每个全称（存在）量词句子都可以转换为句子的合取（析取），其中对于每个可以代入变量的可能对象都有一个子句。然后，我们可以使用SAT求解器，如DPLL或Walk-SAT，来确定 $(KB \wedge \neg q)$ 的（不）可满足性。

这种方法存在一个问题，即我们可以进行无穷多次替换，因为我们对一个符号应用函数的次数没有限制。例如，我们可以根据需要多次嵌套函数 Classmate(...Classmate(Classmate(Austen))...)，直到涉及整个学校。幸运的是，雅克·埃尔布朗（1930年）证明的一个定理告诉我们，如果一个句子被一个知识库所蕴含，那么存在一个仅涉及命题化知识库有限子集的证明。因此，我们可以尝试遍历有限子集，具体来说是通过对嵌套函数应用进行迭代加深搜索，即首先搜索常量符号的替换，然后是Classmate(Austen)的替换，接着是Classmate(Classmate(Austen))的替换，等等。

另一种方法是直接使用一阶逻辑进行推理，也称为提升推理。例如，我们已知 $(\forall x HasAbsolutePower(x) \wedge \text{Person}(x) \Rightarrow \text{Corrupt}(x)) \wedge \text{Person}(John) \wedge \text{HasAbsoluteP}$ （“绝对权力绝对腐败”）。我们可以通过用 x 替换 John 来推断出 $\text{Corrupt}(\text{John})$ 。此规则称为广义假言推理。一阶逻辑的前向链接算法反复应用广义假言推理和替换来推断出 q 或者表明无法推断出它。

逻辑智能体

既然我们已经了解了如何阐述我们所知道的内容以及如何据此进行推理，接下来我们将讨论如何把演绎能力融入到我们的智能体中。智能体应该具备的一项明显能力是，根据观察历史以及它对世界的了解（状态估计），弄清楚自己处于什么状态。例如，如果我们告诉智能体，在熔岩池附近空气开始闪烁，并且它观察到自己正前方的空气在闪烁，那么它就可以推断附近有危险。

为了将其过去的观察结果纳入对当前位置的估计中，智能体需要有时间概念以及状态之间的转换概念。我们将随时间变化的状态属性称为流，并为时间写一个带索引的流，例如，在时间 t 空气是热的 $Hot^t =$ 。如果在那个时候有某种东西导致空气变热，或者在前一个时间空气是热的且没有发生改变它的动作，那么在时间 t 空气应该是热的。为了表示这个事实，我们可以使用下面后继状态公理的一般形式：

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$

在我们的世界中，这种转变可以表述为 $\text{Hot}^{t+1} \Leftrightarrow$ 靠近熔岩一步 $^t \vee \left(\text{Hot}^t \wedge \neg \text{StepAwayFromLava}^t\right)$.

在逻辑上写出世界的规则后，我们现在实际上可以通过检查某些逻辑命题的可满足性来进行规划！为此，我们构建一个句子，其中包含有关初始状态、转变（后继状态公理）和目标的信息。（例如，InOasis $^T \wedge$ Alive ^{T} 编码了在时间T之前在绿洲中生存并最终到达绿洲的目标）。如果世界的规则已经正确制定，那么找到所有变量的一个满足赋值将使我们能够提取出一系列行动，这些行动将把智能体带到目标。

总结

在本笔记中，我们介绍了基于知识的智能体可用于对世界进行推理和决策的逻辑概念。我们介绍了逻辑语言、其语法以及标准逻辑等价式。命题逻辑是一种基于命题符号和逻辑连接词的简单语言。一阶逻辑是一种比命题逻辑更强大的表示语言。一阶逻辑的语法建立在命题逻辑的基础之上，使用项来表示对象，并使用全称量词和存在量词进行断言。

我们进一步描述了用于检查命题逻辑中可满足性（SAT问题）的DPLL算法。它是对可能模型的深度优先枚举，使用提前终止、纯符号启发式和单元子句启发式来提高性能。当我们的知识库仅由命题逻辑中的文字和蕴含式组成时，前向链接算法可用于推理。

一阶逻辑中的推理可以直接通过使用诸如广义假言推理之类的规则来完成，或者通过命题化来完成，命题化将问题转化为命题逻辑并使用SAT求解器得出结论。