15. 句子含义的逻辑表示

ISHMAEL: *Surely all this is not without meaning.*

Herman Melville, *Moby Dick*

在这一章中，我们介绍了语言表达的含义可以通过被称为含义表示(meaning representations)的形式结构来表达。考虑一些需要某种形式的语义处理的任务，比如通过阅读手册学习使用新软件，通过阅读菜单决定在餐馆点什么菜，或者遵循菜谱。完成这些任务需要将语言元素与必需的非语言知识联系起来。阅读菜单，决定点什么菜，给去哪里吃饭的建议，根据食谱，生成新的食谱，所有这些都需要关于食物和它的准备，人们喜欢吃什么，以及餐馆是什么样的知识。通过阅读使用手册来学习使用软件，或提供使用软件的建议，需要了解该软件以及类似的应用程序、电脑和用户。

在本章中，我们假设语言表达具有含义表示，这些含义表示与用来表示世界上日常常识的东西是相同的。创建这种表示并将其分配给语言输入的过程称为**语义解析(semantic parsing)**或语义分析，而设计含义表示和相关语义解析器的整个过程称为**计算语义(computational semantics)**。

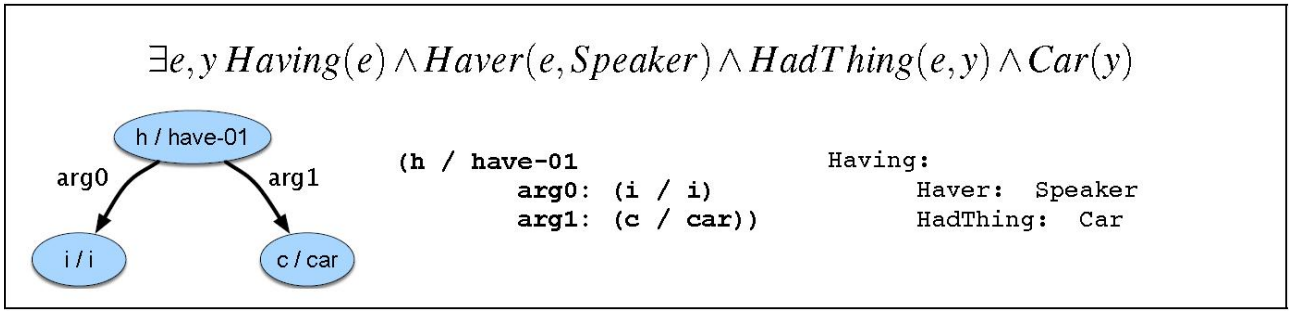

图 15-1：含义表示的采样器

图注：一个符号列表，两个有向图，和一个记录结构:I have a car 的含义表示的采样器。

以图 15.1 为例，它用四种常用的含义表示语言展示了 I have a car 这句话的示例含义表示。

第一行用**一阶逻辑(First-Order Logic)**说明了一个句子，在第 15.3 节中将详细介绍。

有向图及其对应的文本形式是**抽象含义表示(Abstract Meaning Representation，AMR)**形式的一个例子(Banarescu 等人，2013)。

右边是基于框架或**槽填充(slot-filler)**的表示，将在第 15.5 节和第 17 章中再次讨论。

虽然这些方法之间有很大的差异，但它们都有一个共同的概念，即含义表示由一组符号或表示词汇表组成的结构组成。当适当地安排时，这些符号结构被认为是对应于对象，对象的属性，以及对象之间的关系在某些事件的状态被表示或推理。在这种情况下，所有四种表示法都使用了与说话者、一辆汽车和一种关系相对应的符号，这种关系表示两者之间的占有关系。

重要的是，在所有这些方法中，这些表示至少可以从两个不同的角度来看待：作为特定语言输入 I have a car 的含义表示，以及作为某些世界事务状态表示。正是这种双重视角使这些表示能够将语言输入与世界以及我们对世界的认识联系起来。

在接下来的部分中，我们提供一些背景知识：我们对一种含义表示语言的渴望，并保证这些表示将实际执行我们需要它们执行的操作-提供与所表示的事务状态的对应关系。在 15.3 节中，我们介绍了一阶逻辑，这是历史上研究自然语言语义的主要技术，并在 15.4 节中介绍了如何将其用于捕获英语中事件和状态的语义。然后，第 16 章介绍了语义解析的技术：给定语言输入，生成这些形式的含义表示。

15.1. 表示的计算需要

让我们考虑一下为什么需要含义表示，以及它们应该为我们做什么。为了集中讨论，让我们考虑一个基于知识库向游客提供餐馆建议的系统。

①可证性

考虑下面这个简单的问题:

(15.1) Does Maharani serve vegetarian food?

要回答这个问题，我们必须知道它问的是什么，并且知道它问的问题针对 Maharini 来说是真还是假。

可验证性(verifiability)是系统比较表示形式所描述的事务状态与某个知识库中建模的世界中事务状态的能力。例如，我们需要某种表示形式诸如 Serves(Maharani，VegetarianFood)，系统可以将其与特定餐厅的事实知识库相匹配，如果找到与该命题相匹配的表示形式，则可以回答 yes；否则，如果对当地餐馆的了解是完整的，则必须说 No，或者说它不知道它是否知道自己的知识是不完整的。

②非歧义表示

语义学，就像我们所研究过的所有其他领域一样，容易出现歧义。单词和句子在不同的上下文中具有不同的含义。考虑下面的例子:

(15.2) I wanna eat someplace that's close to ICSI

这句话可能意味着说话者想在附近的某个地方吃饭，或者根据哥斯拉作为说话者的解释，说话者可能想吃附近的某个地方。这句话含糊不清； 单个语言表达可以具有两种含义之一。但是我们的含义表示本身不能是模棱两可的。输入含义的表示应没有任何歧义，以便系统可以推理表示(这意味着一件事或另一件事)来决定如何回答。

与歧义密切相关的概念是模糊性(vagueness)：其中含义表示使含义的某些部分未被充分说明。模糊性不会引起多种表示。考虑以下请求：

(15.3) I want to eat Italian food.

虽然意大利菜可以提供足够的信息来提供推荐，但对于用户真正想吃什么，它却很模糊。对于这个短语的含义的模糊表示可能适合于某些目的，而对于其他目的可能需要更具体的表示。

③典范形式(canonical form)

这一学说认为不同的输入意味着相同的东西应该有相同的含义表示。这种方法极大地简化了推理，因为系统只需要处理潜在广泛表达式的单一含义表示。

针对表达方式(15.1)，考虑以下替代方案:

(15.4) Does Maharani have vegetarian dishes?

(15.5) Do they have vegetarian food at Maharani?

(15.6) Are vegetarian dishes served at Maharani?

(15.7) Does Maharani serve vegetarian fare?

尽管这些替代方案使用不同的单词和句法，但我们希望它们映射到单一的典范含义表示。如果它们都不同，假设系统的知识库只包含对这一事实的单一表示，那么大多数表示都不会匹配。当然，我们可以在知识库中存储相同事实的所有可能的替代表示，但是这样做将导致保持知识库的一致性方面的巨大困难。

典范形式的确会使语义解析的任务复杂化。我们的系统必须得出这样的结论:vegetarian fare, vegetarian dishes, 和 vegetarian food 指的是同一件事，在这里，having 和 served 是等价的，所有这些解析结构仍然指向相同的含义表示。或者考虑以下两个例子:

(15.8) Maharani serves vegetarian dishes.

(15.9) Vegetarian dishes are served by Maharani.

尽管 serve 论元的位置不同，但系统仍必须通过利用语法知识(例如主动句和被动句结构之间的关系)将 Maharani 和 vegetarian dishes 分配给两个示例中的相同角色。

④推理和变量

对于更复杂的请求，例如：

(15.10) Can vegetarians eat at Maharani?

此请求得到的答案与其他请求的答案相同，不是因为它们的意思相同，而是因为素食者所吃的食物与素食餐厅所提供的食物之间存在常识性的联系。这是关于世界的事实。我们需要在知识库中将此请求的含义表示与关于世界的事实联系起来。系统必须能够使用**推理(inference)**-根据输入的含义表示及其背景知识得出有效的结论。系统必须有可能得出关于命题真实性的结论，这些结论没有在知识库中明确表示，但是从逻辑上可以从存在的命题中得出。

现在考虑以下一些更复杂的请求：

(15.11) I'd like to find a restaurant where I can get vegetarian food

此要求并不涉及任何特定的餐厅;该用户想要了解一家不知名的素食餐厅的信息。由于没有餐厅的名字，简单的搭配是行不通的。回答这个请求需要使用**变量(variable)**，使用如下表示:

$$Serves(x, VegetarianFood) \tag{15.12}$$

只有当变量 x 可以被知识库中的某个对象替换，并且整个命题能够匹配时，匹配才会成功。然后可使用替代变量的概念来满足用户的请求。任何含义表示语言都必须能够处理这类不确定的参考。

⑤表现力

最后，含义表示方案必须具有足够的表现力，以处理广泛的主题，理想情况下应是任何明智的自然语言语段。尽管对于任何单个表示系统来说，这可能都太过期望了，但是如 15.3 节中所述，一阶逻辑具有足够的表现力，可以处理很多需要表示的东西。

15.2. 模型理论语义学

含义表示语言是什么意思?此语言允许它们满足这些需求，弥合形式表示与表示(告诉我们一些关于世界事务状态的事情)之间的鸿沟。

答案是**模型(model)**。模型是一种正式的构造，它代表了世界上事务的特定状态。含义表示语言中的表达式可以映射到模型的元素，如对象、对象的属性和对象之间的关系。如果模型准确地捕捉了我们感兴趣的事实，那么含义表示和模型之间的一致映射就提供了含义表示和现实世界之间的桥梁。模型提供了一种惊人的简单和强大的方法来将表达式建立在含义表示语言中。

首先,一些术语。含义表示的词汇表由两部分组成:非逻辑词汇表和逻辑词汇表。非逻辑词汇表由对象、属性和关系的开放式名称集合组成，这些名称组成了我们试图表现的世界。它们以谓词、节点、连接上的标签或框架中插槽的标签的形式出现在各种方案中。逻辑词汇表由符号、操作符、量词、连接等的封闭集合组成，它们为在给定的含义表示语言中组合表达式提供了正式的手段。

非逻辑词汇表的每个元素都必须在模型中有一个**指示(denotation)**，这意味着每个元素都对应于模型中一个固定的、定义良好的部分。让我们从对象开始。模型的**领域(domain)**是被表示的对象的集合。每个不同的概念、类别或个体都表示领域中的一个独特元素。

我们通过指示具有该属性的领域元素来表示模型中对象的属性;也就是说，属性指示集合。红色属性的指示是我们认为是红色的东西的集合。类似地，对象之间的关系表示一组参与关系的领域元素的有序列表或元组：已婚关系的指示是已婚的域对象对的集合。这种处理属性和关系的方法被称为**外延的(extensional)**，因为我们通过它们的外延和它们的指示来定义概念。总结一下:

- 对象指示领域元素
- 属性指示领域元素的集合
- 关系指示领域元素的元组集合

我们现在需要一个映射，将我们的含义表示映射到相应的指示：一个函数，将我们的含义表示的非逻辑词汇映射到模型中的适当指示。我们将这种映射称为**解释(interpretation)**。

让我们回到我们的餐馆建议应用程序，让它的领域包含餐馆、顾客、关于顾客喜好和厌恶的事实，以及关于餐馆的事实，例如它们的菜肴、典型成本和噪音水平。为了开始填充我们的领域 D，让我们假设我们正在处理由非逻辑符号指定的四个顾客:Matthew、Franco、Katie 和 Caroline，表示四个唯一的领域元素。我们用常数 a、b、c、d 来表示这些领域元素。我们故意为我们的领域元素使用没有含义的、非助记符的名称，以强调这样一个事实：无论我们对这些实体的了解是什么，都必须来自模型的形式属性，而不是来自符号的名称。再继续,让我们假设我们的应用程序包括三个餐厅,设定为 Frasca、Med、和 Rio 的含义表示，以指示领域元素的 e、f 和 g。最后,让我们假设我们在处理三个菜系 Italian、Mexican、和 Eclectic,在我们的模型中用 h、i 和 j 来指示。

像 Noisy 这样的属性指示我们领域中已知是嘈杂的餐馆子集。两两（Two-place）关系概念，例如个别顾客喜欢(Like)哪个餐厅，表示领域对象的有序偶对或元组。而且，由于我们决定在模型中将菜系表示为对象，所以我们可以将哪个餐厅提供(serve)哪些菜系作为一组元组。图 15.2 给出了一种可能的情况。

Domain	$\mathcal{D} = \{a, b, c, d, e, f, g, h, i, j\}$
Matthew, Franco, Katie and Caroline	a, b, c, d
Frasca, Med, Rio	e, f, g
Italian, Mexican, Eclectic	h, i, j
Properties	
Noisy	$Noisy = \{e, f, g\}$
Frasca, Med, and Rio are noisy	
Relations	
Likes	$Likes = \{\langle a, f\rangle, \langle c, f\rangle, \langle c, g\rangle, \langle b, e\rangle, \langle d, f\rangle, \langle d, g\rangle\}$
Matthew likes the Med	
Katie likes the Med and Rio	
Franco likes Frasca	
Caroline likes the Med and Rio	
Serves	$Serves = \{\langle f, j\rangle, \langle g, i\rangle, \langle e, h\rangle\}$
Med serves eclectic	
Rio serves Mexican	
Frasca serves Italian	

图 15-2：餐厅世界的模型

有了这个简单的方案，我们就可以通过参考相应模型中的适当指示来使含义表示法成为基础。例如，我们可以通过两种映射来评估一个声称 Matthew likes the Rio(或者 The Med serves Italian)的表示：

第一种映射：将含义表示中的对象映射到其对应的领域元素；

第二种映射：将含义表示中的任何连接、谓词或槽映射到模型中的适当关系。

更具体地讲，我们可以通过两种映射验证一个断言“Matthew likes Frasca”的表示：

第一种映射：首先使用我们的解释功能将符号 Matthew 映射到其符号 a，将 Frasca 映射到 e，

第二种映射：将 Likes 关系映射到适当的元组集合。然后，我们检查元组集合中是否存在元组<a，e>。在这种情况下，如果模型中存在该元组，那么我们可以得出结论 Matthew likes Frasca 是真的；如果模型中不存在该元组，那么我们就不能得出结论。

这是非常直接的——我们使用集合和集合上的操作来建立我们的含义表示的表达式。当然，更有趣的部分是当我们考虑更复杂的例子，如以下:

(15.13) Katie likes the Rio and Matthew likes the Med.

(15.14) Katie and Caroline like the same restaurants.

(15.15) Franco likes noisy, expensive restaurants.

(15.16) Not everybody likes Frasca.

对于这样的例子，我们为表示的含义建立基础的简单方案是不够的。这些示例的合理含义表示不会直

接映射到单个实体、属性或关系。相反，它们涉及诸如连词、等式、量化变量和否定等复杂问题。为了评估这些表述是否与我们的模型一致，我们必须将它们拆开，评估各部分，然后从各部分的含义确定整体的含义。

考虑上面的第一个例子。此示例的含义表示将包括两个不同的命题，它们表示单个用户的偏好，并与某种隐式或显式的连接操作符相结合。我们的模型没有将模型中所有顾客和餐馆的成对偏好编码起来的关系，也不需要这样做。从我们的模型中我们知道 Matthew likes the Med，而 Katie likes the Rio(也就是说，元组<a, f>和<c,g>是由 likes 关系表示的集合的成员)。我们真正需要知道的是如何处理操作符的语义。如果我们为英语单词 and 假设最简单的语义，那么如果模型中的每个成分都为真，则整个语句为真。在这种情况下，两个成分都为真，因为适当的元组出现了，因此整个句子为真。

我们在这个例子中所做的是为假设的连接操作符在某些含义表示中提供**真值条件语义学(truth-conditional semantics)**。也就是说，我们提供了一种方法，可以通过部分的含义(通过咨询模型)来确定一个复杂表达式的真值，通过咨询真值表来确定一个操作符的意义。含义表示语言是有真值条件的，因为它们对我们如何从复合句的部分含义来确定复合句的含义给出了正式的说明。特别地，我们需要知道正在使用的含义表示模式的整个逻辑词汇的语义。

请注意，尽管这种情况如何发生的细节取决于所使用的特定含义表示的细节，但应该清楚的是，评估此类例子的真值条件除了我们已经讨论过的简单集合操作之外，不涉及任何其他内容。我们在下一节讨论一阶逻辑的语义时再讨论这些问题。

15.3. 一阶逻辑

一阶逻辑(First-Order Logic，FOL)是一种灵活的、易于理解的、在计算上易于处理的含义表示语言，它满足了 15.1 节中给出的许多要求。它为可验证性、推理和表现力需求提供了良好的计算基础，以及良好的模型理论语义。

FOL 的另一个吸引人的特点是，对于事物应该如何被表示，FOL 很少做出具体的承诺，而它所做的那些都是前面提到的许多方案所共有的:被表示的世界由对象、对象的属性和对象之间的关系组成。

本节的其余部分将介绍 FOL 的基本句法和语义，然后描述 FOL 在事件表示中的应用。

15.3.1. 一阶逻辑的基本要素

让我们通过首先检查 FOL 的各种原子元素，然后展示如何组合它们来创建更大的含义表示来探索 FOL。

图 15.3 为我们将要使用的特定 FOL 句法提供了一个完整的上下文无关的语法，这是本节的路线图。

首先，我们来研究**术语(term)**的概念，即用于表示对象的 FOL 设备(device)。从图 15.3 可以看出，FOL 提供了三种表示这些基本构件的方法：常量、函数和变量。这些手段中的每一个都可以被认为是在考虑中的对象的指定对象。

FOL 中的**常量(constants)**指的是被描述世界中的特定对象。这些常量通常被描述为单个大写字母，如 A 和 B，或者首字母大写单词，这常常让人联想到专有名词，如 Maharani 和 Harry。像编程语言常量一样，FOL 常量只指向一个对象。但是，对象可以有多个引用它们的常量。

FOL 中的**函数(function)**对应的概念在英语中通常表示为所有格或者属格(genitives)，如 Frasca's location。这样一个表达式的 FOL 翻译可能如下所示：

$$LocationOf(Frasca) \tag{15.17}$$

FOL 函数在语法上与单论元谓词相同。但是，重要的是要记住，虽然它们看起来像谓词，但实际上它们是术语，因为它们指的是唯一的对象。函数提供了一种方便的方法来引用特定的对象，而不必将命名常量与它们关联起来。这在许多命名对象(如餐馆)具有独特概念(如与之相关的位置)的情况下尤其方便。

变量(variable)是我们引用对象的最终 FOL 机制。变量被描述为单个小写字母，我们可以断言并推断对象，而不必引用任何特定的命名对象。这种对匿名对象进行声明的能力有两种:对特定未知对象进行声明，以及对任意对象世界中的所有对象进行声明。我们将在介绍量词(使变量有用的 FOL 元素)后，再讨论变量。

$$
\begin{aligned}
Formula &\rightarrow AtomicFormula \\
&\mid Formula\ Connective\ Formula \\
&\mid Quantifier\ Variable, \ldots\ Formula \\
&\mid \neg\ Formula \\
&\mid (Formula) \\
AtomicFormula &\rightarrow Predicate(Term, \ldots) \\
Term &\rightarrow Function(Term, \ldots) \\
&\mid Constant \\
&\mid Variable \\
Connective &\rightarrow \wedge \mid \vee \mid \Longrightarrow \\
Quantifier &\rightarrow \forall \mid \exists \\
Constant &\rightarrow A \mid VegetarianFood \mid Maharani \cdots \\
Variable &\rightarrow x \mid y \mid \cdots \\
Predicate &\rightarrow Serves \mid Near \mid \cdots \\
Function &\rightarrow LocationOf \mid CuisineOf \mid \cdots
\end{aligned}
$$

图 15-3：一阶逻辑表示的句法的上下文无关语法规范

现在我们有了引用对象的方法，我们可以继续讨论用于声明对象之间的关系的 FOL 机制。谓词是指(或称)给定领域中固定数量的对象之间的关系的符号。回到第 15.1 节非正式介绍的例子，一个合理的 FOL 表示 Maharani serves vegetarian food 可能看起来像下面的公式:

$$\text{Serves(Maharani,VegetarianFood)} \tag{15.18}$$

这个 FOL 语句断言 serve 是一个双位置谓词，位于常量 Maharani 和 vegeanfood 所表示的对象之间。下面是一个非常典型的句子，例如 Maharani is a restaurant，这说明了谓词的不同用法:

$$\text{Restaurant(Maharani)} \tag{15.19}$$

这是一个使用单个位置(one-place)谓词的示例，它不是用于关联多个对象，而是用于断言单个对象的一个属性。在这种情况下，它编码了 Maharani 的类别成员。

通过引用对象、断言关于对象的事实以及将对象彼此关联起来的能力，我们可以创建基本的复合表示。这些表示对应于图 15.3 中的原子公式层面。然而，这种组合复杂表示的能力并不仅限于使用单个谓词。更大的组合表示也可以通过使用**逻辑连接词(logical connectives)**组合在一起。从图 15.3 可以看出，逻辑连接词让我们通过使用三种操作符中的一种来连接逻辑公式，从而创建更大的表示。例如，考虑下面的 BERP 语句和它的一种可能的表示:

(15.20) I only have five dollars and I don't have a lot of time.

$$\text{Have(Speaker,FiveDollars)} \wedge \neg \text{Have(Speaker,LotOfTime)} \tag{15.21}$$

这个例子的语义表示是通过使用∧和¬操作符，以一种简单的方式从单个从句的语义建立起来的。请注意，图 15.3 中的语法的递归性质允许通过使用这些连接词创建无限个逻辑公式。因此，与句法一样，我们可以使用一个有限的手段来创建无限个表示。

15.3.2. 变量和量词

现在，我们已经具备了一切必要的机制，可以回到我们先前关于变量的讨论上来。如上所述，在 FOL 中变量以两种方式使用:引用特定的匿名对象和引用集合中的所有对象。这两种用法是通过使用称为**量词(quantifiers)**的操作符实现的。对 FOL 来说，两个基本的操作符是：

存在量词(existential quantifier)，用∃表示“存在”，发音为“there exists”；

全称量词(universal quantifier)，用∀表示“对于所有的”，发音为“for all”。

英语中经常出现不定名词短语来表示存在性量化变量的需要。考虑下面的例子:

(15.22) a restaurant that serves Mexican food near ICSI.

这里引用的是具有特定属性的特定类别的匿名对象。以下是对这一短语含义的合理解释:

$$\exists x Restaurant(x) \wedge Serves(x,MexicanFood) \wedge Near((LocationOf(x),LocationOf(ICSI)) \quad (15.23)$$

这个句子开头的存在量词告诉我们如何在这个句子的上下文中解释变量 x。非正式地说，这个句子要为真，就必须至少有一个对象。因此，如果我们用它来代替变量 x，那么得到的句子就会为真。例如，如果 AyCaramba 是 ICSI 附近的一家墨西哥餐厅，那么将 AyCaramba 代入 x 的结果，逻辑公式如下:

$$Restaurant(AyCaramba) \wedge Serves(AyCaramba,MexicanFood) \wedge Near((LocationOf(AyCaramba),LocationOf(ICSI)) \quad (15.24)$$

根据∧操作符的语义，如果它的三个组成原子公式都为真，则这个句子为真。如果它们存在于系统的知识库中，或者可以从知识库中的其他事实推断出来，那么它们将依次为真。

全称量词的使用还具有基于将已知对象替换为变量的解释。全称量词的替代语义从字面上理解为“对于一切(far all)”。∀操作符指出，要使所讨论的逻辑公式正确，用知识库中的任何对象替换通用量化变量都应得出一个真实公式。这与∃操作符形成了鲜明的对比，∃操作符只要求单个有效的替换语句为 ture。

考虑以下示例：

(15.25) All vegetarian restaurants serve vegetarian food

该句子的合理表示应类似于以下内容：

$$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x,VegetarianFood) \quad (15.26)$$

要使这个句子为真，每个已知对象对 x 的替换都必须产生一个为真的句子。我们可以把所有可能的替代品的集合分为由素食餐厅组成的对象集合和由其他所有东西组成的集合。让我们首先考虑这样一种情况:被取代的对象实际上是一家素食餐厅;一个这样的替换将产生下列句子:

$$VegetarianRestaurant(Maharani) \Rightarrow Serves(Maharani,VegetarianFood) \quad (15.27)$$

如果我们假设我们知道结果从句:

$$Serves(Maharani,VegetarianFood) \quad (15.28)$$

为真，那么这句话整体上也一定为真。因为前提和结果的值都为真，因此，根据图 15.4 的前两行，句子本身的值为真。这个结果将是相同的所有可能的替代术语代表素食餐厅 x。

然而，请记住，要使这个句子为真，它必须对所有可能的替换都为真。当我们从一组不是素食餐厅的对象中考虑替换时，会发生什么呢？考虑一个非素食餐厅，比如 AyCaramba 来替代变量 x:

$$VegetarianRestaurant(AyCaramba) \Rightarrow Serves(AyCaramba,VegetarianFood)$$

因蕴涵(implication)的前提是 False，故我们可以从图 15.4 确定句子始终为 True，再次满足∀约束。

请注意，AyCaramba 可能仍然会在没有成为素食餐厅的情况下提供素食。还要注意，尽管我们选择了示例，但是对于可以用这种推理代替 x 的对象没有隐含的绝对限制。换句话说，x 不受餐厅或与餐厅有关的概念的限制。考虑以下替换：

$$VegetarianRestaurant(Carburetor) \Rightarrow Serves(Carburetor,VegetarianFood)$$

在这里，前提仍然是假的，所以规则在这种不相关的替换下仍然是真。回顾一下，逻辑公式中的变量必须是存在的(∃)或全称的(∀)量化的。为了满足一个存在的量化变量，至少有一个替换必须产生一个真实的句子。为了满足一个全称的量化变量，所有的替换都必须产生真实的句子。

15.3.3. Lambda 表示法

我们需要完成对 FOL 的讨论的最后一个元素称为λ表示法(lambda notation)(Church, 1940)。这种表示法提供了一种从完全指定的 FOL 公式中抽象出来的方法，这种方法对语义分析特别有用。λ表示法扩展了 FOL 的句法，包括以下形式的表达式:

$$\lambda x.P(x) \quad (15.29)$$

这样的表达式由希腊符号λ组成，后面跟着一个或多个变量，然后是一个利用这些变量的 FOL 公式。这些λ表达式的有用性基于将它们应用于逻辑项的能力，从而产生新的 FOL 表达式，其中形式参数变

量绑定到指定的项。这个过程被称为**λ化简(λ-reduction)**，由λ变量的简单文本替换和λ的移除组成。下面的表达式说明了一个λ表达式对常数 A 的应用，以及对该表达式进行λ化简的结果:

$$\lambda x.P(x)(A) \tag{15.30}$$
$$P(A)$$

这种技术的一个重要而有用的变化是使用一个λ表达式作为另一个λ表达式的主体，如下所示:

$$\lambda x.\lambda y.Near(x, y) \tag{15.31}$$

这个相当抽象的表达可以解释为某物接近另一物的状态。下面的表达式说明了用这种嵌入λ表达式进行单个λ应用和随后的化简:

$$\lambda x.\lambda y.Near(x, y)(Bacaro) \tag{15.32}$$
$$\lambda y.Near(Bacaro, y)$$

这里的重点是得到的表达式仍然是λ表达式;第一个化简限定了变量 x，去掉了外部的λ，从而揭示了内部表达式。正如所期望的那样，这个λ表达式可以反过来应用到另一项，从而得到一个完全指定的逻辑公式，如下所示：

$$\lambda y.Near(Bacaro, y)(Centro) \tag{15.33}$$
$$Near(Bacaro,Centro)$$

这种通用技术称为**柯里化(currying)**(Schonfinkel, 1924)[1]，是一种将带有多个论元的谓词转换为单个论元谓词序列的方法。正如我们在第 16 章中所展示的，λ符号提供了一种方法，当谓词的论元没有全部作为解析树中谓词的子项一起出现时，可以将它们递增地收集到谓词中。

15.3.4. 一阶逻辑的语义

在 FOL 知识库中表示的各种对象、属性和关系通过它们与被建模的外部世界中的对象、属性和关系的对应来获得它们的含义。我们可以使用第 15.2 节介绍的模型理论方法来实现这一点。回想一下，这种方法使用简单的集合理论概念来提供从含义表示中的表达式到正在建模的事件状态的真条件映射。我们可以将这种方法应用到 FOL 中，遍历图 15.3 中的所有元素，并指定每个元素应如何解释。

我们可以从断言我们世界中的对象(FOL 术语)表示领域中的元素开始，并断言原子公式是作为属性的领域元素集捕获的，还是作为关系的元素元组集合捕获的。作为一个例子，考虑以下情况:

(15.34) Centro is near Bacaro.

要在 FOL 中捕获这个例子的含义，需要识别与句子中的各种语法元素对应的术语和谓词，并创建逻辑公式，以捕获句子中的单词和句法所暗示的关系。对于这个例子，这样的努力可能会产生如下结果:

$$Near(Centro,Bacaro) \tag{15.35}$$

这个逻辑公式的含义取决于术语 Centro 和 Bacaro 所表示的领域元素是否包含在由当前模型中的谓词 Near 所表示的关系所表示的元组中。

对包含逻辑连接词的公式的解释是基于公式中成分的含义和它们所包含的连接词的含义的结合。图 15.4 给出了图 15.3 中所示逻辑操作符的解释。

∧(and，与)和¬(not，非)操作符的语义相当简单，至少与相应英语术语的某些含义相关。然而，值得指出的是，∨(or，或)操作符与相应的英文单词不以同样的方式析取，而且⇒(imply,蕴涵)操作符只是松散地以任何有关含义或因果的常识概念为基础。

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Longrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

图 15-4：真值表给出了各种逻辑连接词的语义

[1] Currying 是标准的术语，尽管 Heim 和 Kratzer(1998)提出了一个关于 Currying 的术语“Schonkfinkelization”的有趣的争论，因为 Curry 后来建立在 Schonfinkel 的工作之上。

我们需要处理的最后一点涉及变量和量词。回想一下，在我们的基于集合的模型中没有变量，只有领域的元素和它们之间的关系。我们可以利用第 15.3.2 节介绍的替换概念，为有变量的公式提供基于模型的解释。如果变量的替换项的结果公式在模型中是真实的，那么涉及∃的公式是真实的，。包含∀的公式在所有可能的替换下都必须是正确的。

15.3.5. 推理

一种含义表示语言必须支持推理，以便向知识库中添加有效的新命题，或者确定没有明确包含在知识库中的命题的真实性(第 15.1 节)。本节简要讨论一种由 FOL 提供的最广泛实施的推理方法。

取式推理(Modus ponens)是一种推理形式，与非正式的 if-then 推理相对应。我们可以抽象地定义取式推理如下，其中α和β应作为 FOL 公式:

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta} \tag{15.36}$$

这样的模式表明，可以通过某种形式的推理从该行上面的公式推导出该行下面的公式。取式推理表明，如果一个蕴涵规则的左手边是真的，那么右手边的规则就可以推断出来。在接下来的讨论中，我们将把一个蕴涵的左手边称为前提，右手边称为结果。

对于取式推理的典型用法，考虑下面的例子，它使用了上一节的规则:

$$\frac{VegetarianRestaurant(Leaf) \qquad \forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)}{Serves(Leaf, VegetarianFood)} \tag{15.37}$$

在这里，公式 VegetarianRestaurant(Leaf)与规则的前提相匹配，因此我们可以用取式推理的方法得出 Serves(Leaf ,VegetarianFood)的结论。

取式推理的实际应用有两种方式：正向链和反向链。

在**正向链(forward chaining)**系统中，取式推理使用正是如上所述的方式。当单个事实添加到知识库时，使用取式推理来触发所有适用的蕴涵规则。在这种安排中，只要向知识库中添加一个新的事实，就会找到并应用所有适用的蕴涵规则，每一个都导致向知识库中添加新的事实。这些新的命题反过来也可以用来提出适用于它们的蕴涵规则。这一过程继续进行，直到无法推断出进一步的事实为止。

正向链方法的优点是，事实将在需要时出现在知识库中，因为从某种意义上说，所有推理都是提前执行的。这可以大大减少回答后续查询所需的时间，因为它们都应该是简单的查询。这种方法的缺点是，可能会推断和存储永远不需要的事实。

在**反向链(backward chaining)**系统中，取式推理用逆向思维的方式来证明被称为查询的特定命题。第一步是通过确定查询公式是否存在于知识库中来判断它是否为真。如果不是，那么下一步就是搜索知识库中存在的适用的蕴涵规则。适用的规则是规则的结果与查询公式匹配的规则。如果存在这样的规则，那么只要其中任何一个规则的前提都可以证明为真，就可以证明该查询。这可以通过将前提作为新查询进行反向连接来递归地执行。Prolog 编程语言是一个实现这种策略的反向链系统。

为了了解这是如何工作的，让我们假设我们被要求验证命题 Serves(Leaf ,VegetarianFood)的真实性，假设公式(15.37)中的直线以上给出的事实。由于这个命题在知识库中不存在，因此开始搜索一个适用的规则，结果是上面给出的规则。在用常数 Leaf 代替变量 x 之后，我们的下一个任务是证明这个规则的前提，VegetarianRestaurant(Leaf)，当然，这是我们给定的事实之一。

请注意，区分从查询到已知事实的反向链推理和从已知结果到未知前提的反向推理是非常重要的。具体地说，通过反向推理，我们的意思是，如果一个规则的结论为真，我们就假设前提也为真。例如，让我们假设我们知道 Serves(Leaf ,VegetarianFood)为真。由于这个事实与我们的规则的结果相匹配，我们可能会反向推理到 VegetarianRestaurant(Leaf)为真的结论。

虽然反向链是一种合理的推理方法，但反向推理是一种无效的、但通常有用的似是而非的推理形式。从结论到前提的似是而非的推理被称为**诱导法(abduction)**，正如我们在第 22 章中所展示的那样，在解释人们在分析扩展话语时做出的许多推论时，通常是有用的。

虽然向前推理和向后推理都是可靠的，但两者都不是**完备的(complete)**。这意味着有一些有效的推论是系统不能单独使用这些方法来发现的。幸运的是，有一种替代推理技术，称为**归结法(resolution)**，它是可靠的和完备的。不幸的是，基于归结法的推理系统在计算代价上比正向或反向链系统要昂贵得多。因此，在实践中，大多数系统使用某种形式的连接，并给知识库开发人员增加了负担，以允许得出必要的推断的方式对知识进行编码。

15.4. 事件和状态表示

我们希望捕获的许多语义包括状态和事件的表示。状态是在一段时间内保持不变的条件或属性，事件表示某些事件状态的变化。状态和事件的表示都可能涉及大量的参与者、内容(props)、时间和地点。

到目前为止，我们使用的事件和状态的表示都是由单个谓词组成的，这些谓词具有与给定示例相关联的所有角色所需的论元。例如，Leaf served vegetarian fare 的表示由一个谓词组成，该谓词带有执行服务的实体和被服务的事物的论元。

$$Serves(Leaf, VegetarianFare) \tag{15.38}$$

这种方法假设用于表示事件动词的谓词具有与该动词的句法子范畴框架中存在的相同数量的论元。不幸的是，情况显然并非总是如此。来看看动词 eat 的以下例句:

(15.39) I ate.

(15.40) I ate a turkey sandwich.

(15.41) I ate a turkey sandwich at my desk.

(15.42) I ate at my desk.

(15.43) I ate lunch.

(15.44) I ate a turkey sandwich for lunch.

(15.45) I ate a turkey sandwich for lunch at my desk.

显然，为表示 eat 意义的谓词选择正确数量的论元是一个棘手的问题。这些例子以一组不同的句法形式、位置和组合来介绍五个不同的论元或角色。不幸的是，FOL 中的谓词有固定的**元数(arity)**——它们接受固定数量的论元。

为了解决这个问题，我们引入了事件变量(event variable)的概念，以允许我们对特定事件作出断言。为此，我们可以重构事件谓词，使其具有一个存在量化变量作为其第一个也是唯一的论元。使用这个事件变量，我们可以引入其他谓词来表示我们拥有的关于事件的其他信息。这些谓词将事件变量作为它们的第一个论元，相关的 FOL 术语作为它们的第二个论元。下面的公式用我们前面讨论过的 15.40 的含义表示来说明这个方案。

$$\exists e\ Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TurkeySandwich)$$

这里，量化变量 e 代表吃的事件，用于将事件谓词与通过命名角色 Eater 和 Eaten 提供的核心信息绑定。为了处理更复杂的示例，我们只需添加额外的关系来捕获所提供的信息，对于 15.45,如下所示：

$$\exists e\ Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TurkeySandwich) \wedge Meal(e, Lunch) \wedge Location(e, Desk) \tag{15.46}$$

这种事件表示被称为**新戴维森(neo Davidsonian)**事件表示(Davidson 1967, Parsons 1990)，以哲学家 Donald Davidson 引入事件变量的概念(Davidson, 1967)命名。总而言之，在新戴维森主义的事件表示方法中:

- 用谓词捕获事件，谓词以单个事件变量作为论元。
- 没有必要为一个给定的谓语指定一个固定数量的论元;相反，只要输入中提供的角色和填充项可以被粘在一起。
- 除了输入中提到的角色外，没有更多的角色是假定的。

•可以满足共享相同谓词的密切相关输入之间的逻辑连接，而不需要额外的推理。

这种方法仍然给我们留下了一个问题，即确定所需的谓词集合，以表示与特定事件(如 Eater 和 Eaten)相关的角色，以及更一般的概念(如位置和时间)。我们将在第十章更详细地回到这个问题。

15.4.1. 时间表示

在我们对事件的讨论中，我们没有认真处理捕获所表示事件应该发生的时间的问题。将这些信息以一种有用的形式表示出来就是**时序逻辑(temporal logic)**的领域。本讨论介绍了时序逻辑的最基本问题，并简要讨论了人类语言传递时间信息的方法，其中包括**时态逻辑(tense logic)**，动词时态传递时间信息的方式。关于时间表达式表示和分析的鲁棒方法的更详细的讨论将在第 17 章中介绍。

最直接的时间理论认为，时间不可阻挡地向前流动，事件与时间上的点或间隔有关，如在时间轴上。我们可以将不同的事件排列在时间轴上;如果时间从第一个事件流向第二个事件，则一个事件先于另一个事件。在大多数理论中，伴随这些概念的是当前时刻的概念。将这一概念与时间顺序关系相结合，就产生了我们熟悉的过去、现在和未来的概念。

许多方案都可以表示这种时间信息。这里展示的是一个相当简单的，它停留在我们一直在追求的具体化事件的 FOL 框架内。考虑以下例子:

(15.47) I arrived in New York.

(15.48) I am arriving in New York.

(15.49) I will arrive in New York.

这些句子都指同一种事件，只是动词的时态不同。在我们当前的事件表示方案中，所有三个事件都将共享以下类型的表示，它缺乏任何时间信息:

$$\exists e Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \quad (15.50)$$

动词的时态提供的时间信息可以利用预测关于事件的附加信息变量 e。具体地说,我们可以添加时间变量代表间隔对应于事件,事件的终点,时间谓词有关这个终点当前时间的动词的时态。对于我们的 arriving 例子，这种方法产生了以下表示:

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge EndPoint(i,n) \wedge Precedes(n,Now)$$

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge MemberOf(i,Now)$$

$$\exists e,i,n\ Arriving(e) \wedge Arriver(e,Speaker) \wedge Destination(e,NewYork) \wedge IntervalOf(e,i) \wedge EndPoint(i,n) \wedge Precedes(Now,n)$$

该表示引入了一个变量来表示与事件相关的时间间隔，以及一个变量来表示该时间间隔的结束。前面的二元谓词 Precedes 表示第一个时间点论元在时间上先于第二个时间点;常量 Now 指的是当前时间。对于过去的事件，间隔的结束点必须在当前时间之前。类似地，对于未来的事件，当前时间必须在事件结束之前。对于当前发生的事件，当前时间包含在事件间隔内。

不幸的是，简单的动词时态和时间点之间的关系绝不是直接的。考虑以下例子:

(15.51) Ok, we fly from San Francisco to Boston at 10.

(15.52) Flight 1390 will be at the gate an hour now.

在第一个例子中，动词 fly 的现在时指将来发生的事，而在第二个例子中，将来时指过去发生的事。

当我们考虑其他动词时态时，会出现更多的复杂情况。考虑以下例子:

(15.53) Flight 1902 arrived late.

(15.54) Flight 1902 had arrived late.

尽管两者都涉及过去的事件，但以同样的方式来描述它们似乎是错误的。第二个例子似乎有另一个未命名的事件潜伏在背景中(例如，1902 航班已经到达晚了，这时发生了另一件事)。为了解释这种现象，Reichenbach(1947)引入了**参考点(reference point)**的概念。在我们简单的时间模式中，当前时刻等同于**说话(utterance)**的时间，并用作事件发生时的参考点(在之前、在此时或之后)。在 Reichenbach 的方法中，

参考点的概念是分离的说话时间和事件时间。下面的例子说明了这种方法的基础:

(15.55) When Mary's flight departed, I ate lunch.

(15.56) When Mary's flight departed, I had eaten lunch.

在这两个例子中，吃饭(eating)事件都发生在过去，也就是说，在说话之前。然而，在第一个例子中的动词时态表明吃饭事件在航班起飞时就开始了，而第二个例子表明吃饭是在航班起飞前完成的。因此，用 Reichenbach 的术语来说，起飞(departure)事件指定了参考点。这些事实可以通过与吃饭和起飞事件相关的额外限制得到满足。在第一个示例中，参考点在吃饭事件之前，在第二个示例中，吃饭在参考点之前。图 15.5 展示了 Reichenbach 对初级英语时态的研究方法。练习 15.6 要求你用 FOL 写出这些例子。

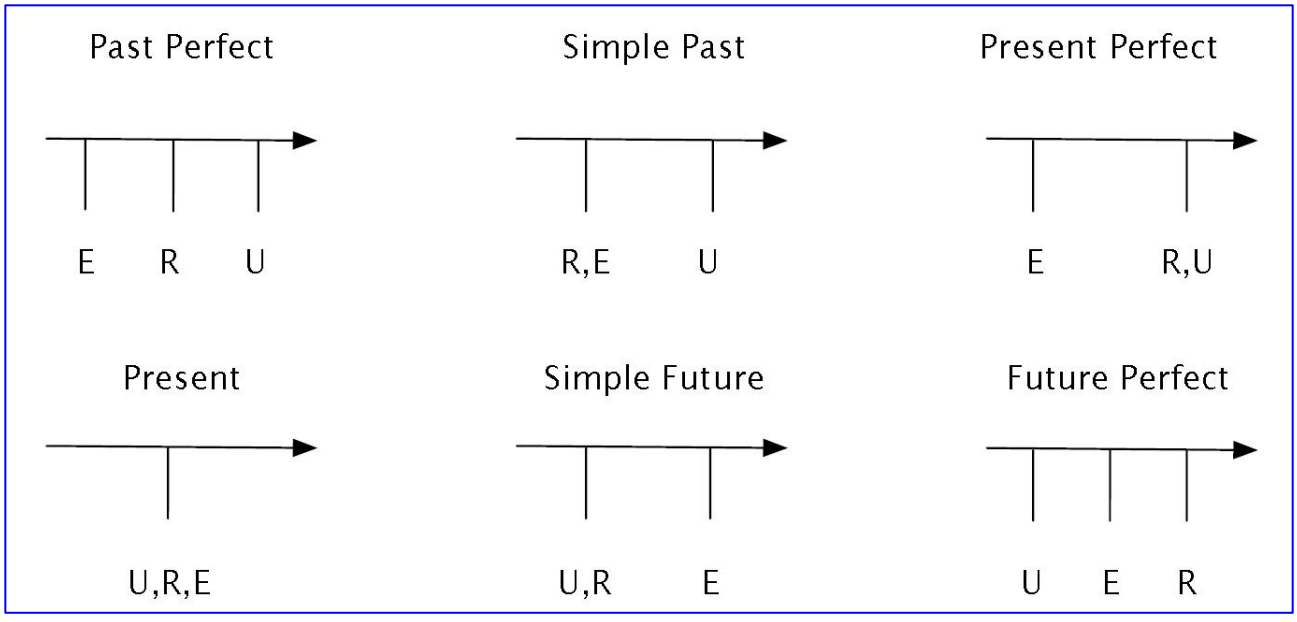

图 15-5：Reichenbach 的研究方法应用于各种英语时态

图注：在这些图中，时间从左向右流动，E 表示事件时间，R 表示参考时间，U 表示说话时间。

这次讨论集中在过去、现在和未来的广义概念上，以及它们是如何被各种英语动词时态所表示的。当然，语言还有很多其他的方式来传递时间信息，包括时间表达式:

(15.57) I'd like to go at 6:45 in the morning.

(15.58) Somewhere around noon, please.

正如我们在第 17 章中所展示的，这种时态表达式的语法对于信息提取和问题回答应用程序具有相当大的实际重要性。

最后，我们应该注意到，系统的概念组织反映在这样的例子中。特别地，时间表达在英语中经常用空间术语来表达，正如这些例子中 at、in、somewhere 和 near 的各种用法所说明的(Lakoff 和 Johnson 1980, Jackendoff 1983)。像这样的隐喻组织，一个领域用另一个领域系统地表达，在世界上的语言中非常常见。

15.4.2. 体

在最后一节中，我们讨论了用描述事件说话的时间来表示事件时间的方法。在本节中,我们强调**体(aspect)**的概念,涉及聚类相关的话题,包括一个事件是否已经结束或正在进行,它是否被概念化为在某个时间点或某个时间间隔内发生，以及世界上是否有任何特定状态是由于此而发生的。基于这些概念和相关概念，传统上将事件表达式分为以下四个示例中所示的四个通用类：

(静态体)**Stative:** I know my departure gate.

(活动体)**Activity:** John is flying.

(完成体)**Accomplishment:** Sally booked her flight.

(达成体)**Achievement:** She found her gate.

尽管 Aristotle(亚里士多德)讨论了该分类的最早版本，但此处介绍的版本归功于 Vendler(1967)。

静态体(stative)表达式表示事件参与者在给定时间点具有特定属性或处于某种状态的概念。因此，这些表达式可以被认为是在单个时间点上捕捉世界的一个方面。考虑下面的 ATIS 示例。

(15.59) I like Flight 840 arriving at 10:06.

(15.60) I need the cheapest fare.

(15.61) I want to go first class.

在这样的例子中，由主语指示的事件参与者可以被视为在特定的时间点对某件事有经验。经验者是否在之前或将来处于同样的状态是未知的。

活动体(activity)表达式描述由参与者承担的事件，并且没有特定的终点。与静态体不同，活动体被看作是在一定时间段内发生的，因此与单个时间点无关。考虑以下例子:

(15.62) She drove a Mazda.

(15.63) I live in Brooklyn.

这两个例子都说明了主语在一段时间内正在或已经从事动词所指定的活动。

最后一个体类，**达成体表示(achievement expression)**，与**完成体**相似，因为这些表示会导致一种状态。考虑以下:

(15.64) She found her gate.

(15.65) I reached New York.

与完成体不同，达成体事件被认为是瞬间发生的，不等同于任何特定的活动导致的状态。更具体地说，这些例子中的事件之前可能有扩展的 searching 或者 traveling 事件，但与 find 和 reach 直接对应的事件被认为是**时间点(points)**，而不是**时间段(intervals)**。

注意，由于完成体和达成体都是导致状态的事件，所以它们有时被描述为单一体类的子类。这种组合类的成员被称为**终结体可能事件(telic eventities)**。

15.5. 描述逻辑

正如本章开头所指出的，已经发明了相当多的表示方案来捕捉语言说话(utterances)的含义。现在人们普遍认为，在这些不同的方法中表示的含义，原则上可以相对容易地在 FOL 中转换成等效的语句。困难在于，在许多这些方法中，语句的语义都是按程序定义的。也就是说，含义来自于解释它的系统。

描述逻辑(DL)是一种努力，旨在更好地指定这些早期结构化网络表示的语义，并提供一个特别适合于某些领域建模的概念框架。在形式上，术语描述逻辑是指一系列逻辑方法，它们对应于不同的 FOL 子集。对描述逻辑的可表达性施加的限制可确保各种关键类型推理的易处理性。但是，我们的重点是描述逻辑的建模方面，而不是计算复杂性问题。

在使用描述逻辑为应用领域建模时，重点是表示有关类别、属于那些类别的个人以及这些个人之间可以保持的关系的知识。组成特定应用领域的一组类别或概念称为它的**专业术语(terminology)**。知识库中包含专业术语的部分传统上称为 **TBox**。这与包含有关个人事实的 **ABox** 相反。该专业术语通常被安排到一个称为**本体知识体系(ontology)**的层次结构中，该本体捕获类别之间的子集/超集关系。

回到我们前面的烹饪领域，我们使用一阶谓词表示领域概念，如 Restaurant(x);等效的 DL 省略了这个变量，因此餐厅类别被简单地写成 **Restaurant**[2]。为了捕捉特定的领域元素(如 Frasca)是餐厅这一事实，我们断言 Restaurant (Frasca)与在 FOL 中断言的方式基本相同。这些类别的语义的指定方式与前面第 15.2 节中介绍的完全相同：像 Restaurant 这样的类别只是表示一组作为餐馆的领域元素。

在特定域中指定了感兴趣的类别后，下一步就是将它们排列成层次结构。捕获专业术语中存在的层次关系有两种方法：我们可以直接断言与层次相关的类别之间的关系；或者也可以为概念提供完整的定义，然后依靠推理来提供层次关系。这些方法之间的选择取决于所使用类别的用途以及为许多自然发生的类别制定精确定义的可行性。我们将在这里讨论第一个选项，并在本节稍后部分返回到定义的概念。

为了直接指定层次结构，我们可以断言专业术语中适当概念之间的**包含(subsumption)**关系。

[2] DL 语句通常采用 sans serif 字体排版。在这里，我们将遵循这种惯例，在给出与 DL 语句等价的 FOL 时，回到我们的标准数学符号。

包含关系常写成 C⊑ D，读作 C 被 D 包含，意即类别 C 的所有成员也是类别 D 的成员。毫不奇怪，这个关系的形式语义是由一个简单的集合关系提供的；在集合 C 中的任何定义域元素也在集合 D 中。

在 TBox 中添加以下语句将断言所有的餐馆都是商业场所，此外，还有各种各样的餐馆子类型。

$$\text{Restaurant} \sqsubseteq \text{CommercialEstablishment} \tag{15.66}$$

$$\text{ItalianRestaurant} \sqsubseteq \text{Restaurant} \tag{15.67}$$

$$\text{ChineseRestaurant} \sqsubseteq \text{Restaurant} \tag{15.68}$$

$$\text{MexicanRestaurant} \sqsubseteq \text{Restaurant} \tag{15.69}$$

像这样的本体知识体系通常用图 15.6 所示的图来说明，其中包含关系由表示类别的节点之间的连接来指示。

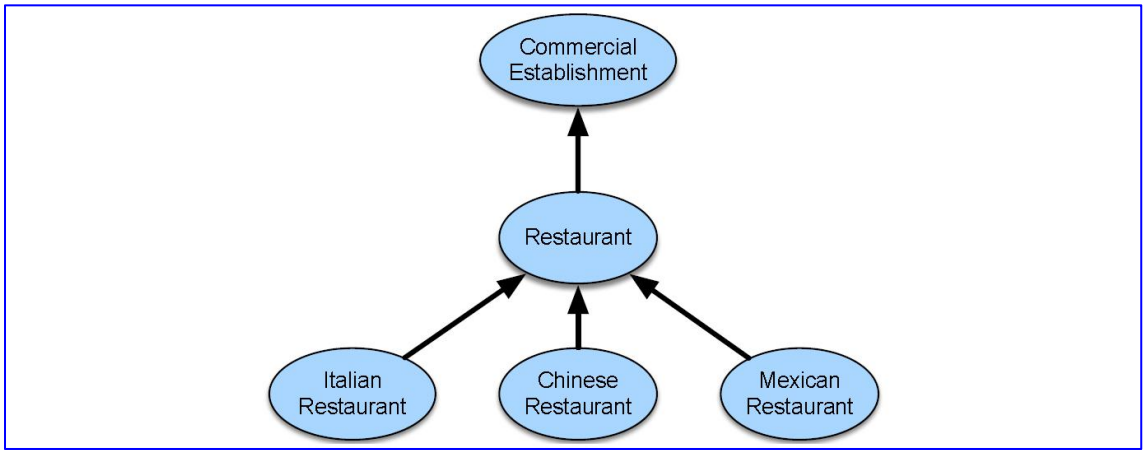

图 15-6：一组包含关系的图形网络表示

注意，正是这样的语义网络图的模糊性质促使了描述逻辑的发展。例如，从该图中我们无法确定给定的类别集合是详尽的还是不相交的。也就是说，我们无法确定这些是我们在该领域中要处理的所有餐厅，还是可能还有其他餐厅。我们也无法确定单个餐厅是否必须仅属于这些类别之一，或者是否有可能(例如，餐厅既可以是意大利餐厅又可以是中国餐厅)。上面给出的 DL 语句的含义更加透明； 他们只是断言类别之间的一系列归属关系，而没有关于覆盖范围或相互排斥的主张。

如果一个应用程序需要覆盖范围和不相交的信息，那么这些信息必须明确地说明。捕获这类信息的最简单方法是使用否定和析取(disjunction)操作符。例如，下面的断言会告诉我们，中国餐馆不可能也是意大利餐馆。

$$\text{ChineseRestaurant} \sqsubseteq \textbf{not}\ \text{ItalianRestaurant} \tag{15.70}$$

指定一组包含一个类别的子概念可以通过析取来实现，如下所示:

$$\text{Restaurant} \sqsubseteq (\textbf{or}\ \text{ItalianRestaurant}\ \text{ChineseRestaurant}\ \text{MexicanRestaurant}) \tag{15.71}$$

图 15.6 所示的层次结构几乎没有告诉我们其中的概念。我们当然不知道什么是餐厅，更不用说意大利的、中国的、或者昂贵的。我们需要的是对成为这些类别中的一员意味着什么的额外断言。在描述逻辑中，这样的语句以被描述的概念和领域中的其他概念之间的关系的形式出现。为了与它在结构化网络表示中的起源保持一致，描述逻辑中的关系通常是二元的，通常被称为角色或角色-关系。

为了了解这种关系是如何运作的，让我们考虑一下本章前面讨论的一些关于餐馆的事实。我们将使用 hasCuisine 关系来获取关于餐馆提供哪种食物的信息，使用 hasPriceRange 关系来获取特定餐馆的价格倾向。我们可以使用这些关系来更具体地说明我们不同类别的餐馆。让我们从我们的意大利餐厅概念开始。作为第一种近似说法，我们可能会说一些没有争议的事情，比如意大利餐馆提供意大利美食。为了捕获这些概念，让我们首先添加一些新的概念到我们的专业术语中来代表各种各样的美食。

$$\text{MexicanCuisine} \sqsubseteq \text{Cuisine} \qquad \text{ExpensiveRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{ItalianCuisine} \sqsubseteq \text{Cuisine} \qquad \text{ModerateRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{ChineseCuisine} \sqsubseteq \text{Cuisine} \qquad \text{CheapRestaurant} \sqsubseteq \text{Restaurant}$$

$$\text{VegetarianCuisine} \sqsubseteq \text{Cuisine}$$

接下来，让我们修改我们早期版本的意大利餐厅，以捕捉美食信息。

$$\text{ItalianRestaurant} \sqsubseteq \text{Restaurantu} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.72}$$

正确读取这个表达式的方法是，类别意大利餐厅中的个体既包含在类别餐厅中，也包含在存在从句定义的未命名类中——提供意大利美食的实体集合。用 FOL 表示的等价表述是：

$$\forall x \mathit{ItalianRestaurant}(x) \rightarrow \mathit{Restaurant}(x) \tag{15.73}$$
$$\wedge (\exists y \mathit{Serves}(x;y) \wedge \mathit{ItalianCuisine}(y))$$

该 FOL 转换应使上面给出的 DL 断言做什么和不做什么都很清楚。特别是，他们并没有说被归类为意大利餐馆的领域实体不能被加入其他关系，例如价格昂贵甚至是中国美食。至关重要的是，他们对我们知道确实提供意大利美食的领域实体并没有多说。实际上，对 FOL 转换的检查清楚地表明，我们不能根据其特征推断任何新实体都属于该类别。我们所能做的最好的事情就是推断有关餐厅的新事实，我们被明确告知餐厅是该类别的成员。

当然，推断具有特定特征的个体的类别成员资格是我们需要支持的常见且关键的推理任务。这使我们回到了用专业术语创建层次结构的另一种方法：实际上以类别成员资格的必要和充分条件的形式提供对我们正在创建的类别的定义。在这种情况下，我们可能会明确地为 ItalianRestaurant 提供一个定义，即提供意大利美食的餐厅，将 ModerateRestaurant 提供为价格范围适中的餐厅。

$$\text{ItalianRestaurant} \equiv \text{Restaurant} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.74}$$

$$\text{ModerateRestaurant} \equiv \text{Restaurant} \sqcap \text{hasPriceRange:ModeratePrices} \tag{15.75}$$

虽然我们前面的陈述为这些类别的成员提供了必要条件，但这些陈述同时提供了必要和充分条件。

最后，让我们来看看表面上类似的素食餐厅的例子。显然，素食餐厅是提供素食的餐厅。但他们不只是提供素食，这是他们提供的全部。我们可以通过在我们之前对 VegetarianRestaurants 的描述基础上增加一个全称量词的形式来适应这种限制，如下所示:

$$\text{VegetarianRestaurant} \equiv \text{Restaurant} \tag{15.76}$$
$$\sqcap \exists \text{hasCuisine:VegetarianCuisine}$$
$$\sqcap \forall \text{hasCuisine:VegetarianCuisine}$$

推理

与描述逻辑对类别、关系和个体的关注并行的是对逻辑推理的受限子集的处理关注。DL 推理系统强调包含和实例检查这两个紧密耦合的问题，而不是使用 FOL 允许的全部推理。

包含(Subsumption)，作为一种推理形式，是基于专业术语中断言的事实来确定两个概念之间是否存在超集/子集关系的任务。相应地，**实例检查(instance checking)**要求(ask)，根据我们所知道的关于个体和专业术语的事实，个体是否可以成为特定类别的成员。包含和实例检查背后的推理机制不仅仅是检查专业术语中显式声明的包含关系。它们必须明确地使用有关专业术语的关系信息进行推理，以推断出适当的包含和成员关系。

回到我们的餐厅领域，让我们使用以下语句添加一种新型餐厅：

$$\text{IlFornaio} \sqsubseteq \text{ModerateRestaurant} \sqcap \exists \text{hasCuisine:ItalianCuisine} \tag{15.77}$$

鉴于此，我们可能会问，IlFornaio 连锁餐厅是否可以归类为意大利餐厅或素食餐厅。更准确地说，我们可以对我们的推理系统提出以下问题:

$$\text{IlFornaio} \sqsubseteq \text{ItalianRestaurant} \tag{15.78}$$

$$\text{IlFornaio} \sqsubseteq \text{VegetarianRestaurant} \tag{15.79}$$

第一个问题的答案是肯定的，因为 IlFornaio 符合我们为 ItalianRestaurant 分类所指定的标准：这是一家餐厅，因为我们将其明确分类为 ModerateRestaurant，它是 Restaurant 的子类型，并且满足 has.Cuisine 类型限制，因为我们已经直接断言了。

第二个问题的答案是否定的。回想一下，我们对素食餐厅的标准包含两个要求：它必须提供素食，而且它只能提供素食。我们目前对 IlFornaio 的定义在这两个方面都失败了，因为我们还没有断言任何声称 IlFornaio 提供素食的关系，而我们断言的关系 hasCuisine.ItalianCuisine 与第二个标准相矛盾。

基于基本包含推论的相关推理任务是，在给定有关专业术语类别的事实的情况下，得出该专业术语的

蕴涵层次结构(implied hierarchy)。该任务大致对应于将包含操作符重复应用到成对的专业术语中。给定我们当前的语句集合，可以推断出图 15.7 中所示的扩展层次结构。您应该使自己确信，该图仅包含所有包含的连接，而根据我们的当前知识，这些连接应该存在。

实例检查的任务是确定一个特定的个体是否可以被分类为特定类别的成员。这个过程以关系和明确的直言陈述的形式获取已知的关于特定个体的信息，然后将这些信息与已知的关于当前专业术语的信息进行比较。然后，它返回个体可以归属的最特定类别的列表。

作为一个分类问题的例子，假设我们被告知是一家提供意大利美食的餐厅。

Restaurant(Gondolier)

hasCuisine(Gondolier,ItalianCuisine)

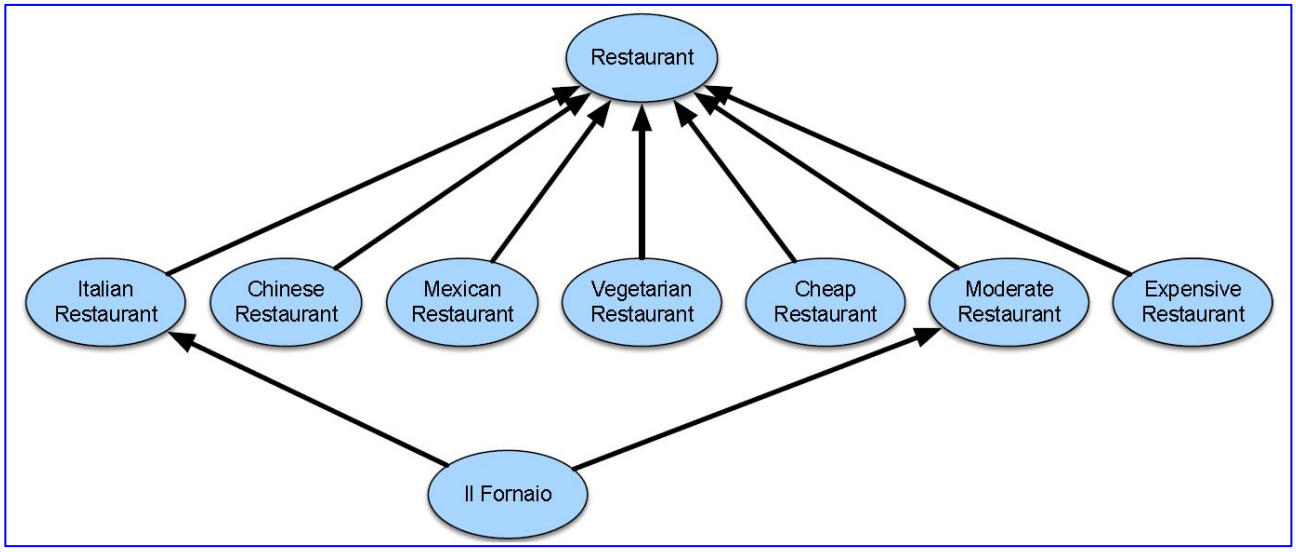

图 15-7：完整包含关系集的图形网络表示

图注：给定 TBox 中的当前断言集，餐饮领域中完整包含关系集的图形网络表示

在这里，我们被告知 Gondolier 这个术语所指的实体是一家餐厅，供应意大利食物。根据这个新信息和我们目前 TBox 的内容，我们可能会有理由问一下，这是一家意大利餐厅，是一家素食餐厅，还是价格适中。

假设前面给出了定义性声明，我们确实可以将 Gondolier 归类为意大利餐厅。也就是说，我们所获得的信息符合该类别成员资格的必要和充分条件。与 IlFornaio 类别一样，此个体也无法满足针对 VegetarianRestaurant 规定的条件。最后，Gondolier 可能还算是一家价格适中的餐厅，但由于我们对其价格一无所知，因此我们目前无法透露。这意味着在给定我们目前的知识的情况下，对查询 ModerateRestaurant(Gondolier)的答案将为假，因为它缺少所需的 hasPriceRange 关系。

包含、实例检查以及实际应用程序所需的其他类型的推理的实现，根据所使用的描述逻辑的表达性而不同。然而，对于即使是中等能力的描述逻辑，主要的实现技术是基于满足性方法的，而满足性方法又依赖于本章前面介绍的基于模型的底层语义。

网络本体语言和语义网

迄今为止，描述逻辑最引人注目的角色是作为语义网开发的一部分。语义网是一项持续的努力，它提供了一种正式指定网络内容语义的方法(Fensel 等人，2003)。这项工作的一个关键成分涉及为各种感兴趣的应用程序领域创建和部署本体。用来表示这些知识的含义表示语言是**网络本体语言(Web Ontology Language，OWL)** (McGuiness 和 van Harmelen, 2004)。OWL 包含了一个描述逻辑，大致对应于我们在这里描述的逻辑。

15.6. 总结

本章介绍了**含义**的**表示**方法。以下是本章的一些重点内容:

•在计算语言学中，一个主要的意义方法涉及到创建**形式含义表示**，以捕获语言输入含义的相关内容。这些表示的目的是弥合语言与世界常识知识之间的鸿沟。

•定义这些表示的句法和语义的框架被称为**含义表示语言**。在自然语言处理和人工智能中使用了各种各样的这种语言。

•这样的表示需要能够支持语义处理的实际计算需求。这些要求包括:确定**命题的真值**、支持**无歧义表示**、表示**变量**、支持**推理**以及足够的**表现力**。

•人类语言有各种各样用来传递含义的特性。其中最重要的是传递**谓词-论元结构**的能力。

•**一阶逻辑**是一种易于理解的，易于计算的含义表示语言，它提供了含义表示语言所需的大部分功能。

•可以在 FOL 中捕获语义表示的重要元素，包括状态和事件。

•可以在 FOL 框架内捕获**语义网络**和**框架**。

•现代**描述逻辑**由完整的一阶逻辑的有用且易计算的子集组成。描述逻辑的最主要用途是语义网规范中使用的**网络本体语言(OWL)**。

15.7. 文献和历史说明

声明性含义表示在自然语言处理中的最早计算性使用是在问答系统的背景下(Green 等人 1961,Raphael 1968, Lindsey 1963)。这些系统对回答问题所需的事实采用了特别的表示。然后，问题被翻译成一种可以与知识库中的事实相匹配的形式。Simmons(1965)概述了这些早期的努力。

Woods(1967)研究了在回答问题时使用 FOL-like 表示，作为当时使用的临时表示的替代。Woods(1973)在具有里程碑意义的月球(Lunar)系统中进一步发展和扩展了这些想法。有趣的是，在 Lunar 中使用的表示既有真值条件，也有程序语义。Winograd(1972)在他的 SHRDLU 系统中采用了一种基于微规划语言的类似表示。

在此期间，对语言和记忆认知模型感兴趣的研究人员一直在研究各种形式的联想网络表示。Masterman(1957)是第一个将类似语义网络的知识表示方法用于计算的人，尽管语义网络通常被认为是 Quillian(1968)的作品。语义网络框架的大量工作是在这一时期进行的(Norman 和 Rumelhart 1975, Schank 1972, Wilks 1975c, Wilks 1975b, Kintsch 1974)。正是在这一时期，一些研究人员开始将 Fillmore 的案例角色概念(Fillmore, 1968)纳入他们的表述中。Simmons(1973)是最早将案例角色作为自然语言处理表示的一部分的使用者。

Woods(1975)和 Brachman(1979)为了弄清语义网络的真正含义而进行的详细分析，导致了许多更复杂的类网络语言的发展，包括 KRL (Bobrow 和 Winograd, 1977)和 KL-ONE (Brachman 和 Schmolze, 1985)。随着这些框架变得越来越复杂和定义得越来越好，很明显，它们是 FOL 的受限变体，外加专门的索引推理过程。在 Brachman 和 Levesque(1985)的著作中可以找到一个有用的论文集，涵盖了大部分这方面的工作。Russell 和 Norvig(2002)描述了这些代表性努力的现代视角。

在生成时代，语言学为自然语言句子分配语义结构的努力始于 Katz 和 Fodor(1963)的工作。它们简单的基于特征的表示方式的局限性，以及逻辑对当时许多语言学问题的自然契合，很快导致了各种谓词-参论元结构作为首选语义表示方式的采用(Lakoff 1972a, McCawley 1968)。随后 Montague(1973)将真值条件模型理论框架引入语言学理论，使形式句法理论与广泛的形式语义框架更加紧密地结合在一起。Dowty 等人(1981)和 Partee(1976)对 Montague 语义学及其在语言学理论中的作用有很好的介绍。

事件作为具体化对象的表示应归功于 Davidson(1967)。这里提出的方法明确地使事件的参加者身份化，这归功于 Parsons(1990)。

目前大多数时间推理的计算方法都是基于 Allen 的时间间隔概念(Allen, 1984);参见第 17 章。

Meulen(1995)提供了对时态和体的现代处理。Davis(1990)描述了 FOL 的使用来代表广泛的常识性领域的知识，包括数量、空间、时间和信念。

van Benthem 和 ter Meulen(1997)对逻辑和语言进行了全面的论述。Lyons(1977)是一个经典的语义文本。McCawley(1993)是一本不可或缺的教科书，涵盖了逻辑和语言的广泛主题。Chierchia 和 McConnell-Ginet(1991)也从语言学的角度广泛地涵盖了语义问题。Heim 和 Kratzer(1998)是从当前生成理论的角度写的一个更近期的文本。

15.8. 练习

15.1 Peruse your daily newspaper for three examples of ambiguous sentences or headlines. Describe the various sources of the ambiguities.

15.2 Consider a domain in which the word *coffee* can refer to the following concepts in a knowledge-based system: a caffeinated or decaffeinated beverage, ground coffee used to make either kind of beverage, and the beans themselves. Give arguments as to which of the following uses of coffee are ambiguous and which are vague.

1. I’ve had my coffee for today.
2. Buy some coffee on your way home.
3. Please grind some more coffee.

15.3 The following rule, which we gave as a translation for Example 15.25, is not a reasonable definition of what it means to be a vegetarian restaurant.

$\forall x \mathit{VegetarianRestaurant}(x) \Rightarrow \mathit{Serves}(x, \mathit{VegetarianFood})$

Give a FOL rule that better defines vegetarian restaurants in terms of what they serve.

15.4 Give FOL translations for the following sentences:

1. Vegetarians do not eat meat.
2. Not all vegetarians eat eggs.

15.5 Give a set of facts and inferences necessary to prove the following assertions:

1. McDonald’s is not a vegetarian restaurant.
2. Some vegetarians can eat at McDonald’s.

Don’t just place these facts in your knowledge base. Show that they can be inferred from some more general facts about vegetarians and McDonald’s.

15.6 For the following sentences, give FOL translations that capture the temporal relationships between the events.

1. When Mary’s flight departed, I ate lunch.
2. When Mary’s flight departed, I had eaten lunch.

15.7 On page 316, we gave the representation *Near*(*Centro,Bacaro*) as a translation for the sentence *Centro is near Bacaro*. In a truth-conditional semantics, this formula is either true or false given some model. Critique this truth conditional approach with respect to the meaning of words like *near*.