第一章　上下文无关语法

说到语法（在本书中，语法均在句子层面使用，因此将不仔细区分语法和句法两个词的使用），人们可能首先会想到语言学课程。在语言学的教科书中，语法是一个主要的内容。在那些语法中，规定了如何用词构造句子，何种用法是不允许的等等。通常，语法可以用来辅助人们完成两件事情，其一是作为判定一个句子构造得是否合适的重要依据，也即,一个句子是否合乎语法 其二,依据语法来分析句子的结构,帮助人们理解句子内容，这一过程在人们学习外语时是尤为明显和重要的。（由此也可见，利用语法来进行句子结构分析对于进行自然语言理解是有一定认知依据的。）

对于计算机自然语言处理，利用认知依据来建立计算模型是一种可行的途径。因而，让计算机能够利用语法来分析句子是进行自然语言处理的一个重要阶段。与人类使用语法相同，计算机利用语法来分析句子也可以有两个层次：其一是识别一个句子是否合乎语法。通常把能完成该任务的计算机程序称为句子识别器。其二是分析句子的内部结构，确定句子的语法成分，为进一步的句子分析和理解提供足够的基础。通常把能完成第二个任务的计算机程序称为句法分析器。显然可以看出，句法分析器比识别器具有更强的能力。

为了实现句子识别器或句法分析器，需要预先赋予计算机两个东西。

第一个是语法：通常语言学教材中的语法是面向人的，为了让机器分析句子，需要让机器知道这些语法，这种面向机器处理的语法也称为形式语法，它是规定语言中允许出现的结构的形式化说明。其中很重要的是如何表示形式语法，即形式语法的表示方式。本章将介绍两种表示方式：重写规则和转移网络。

第二个是语法分析算法：机器依据形式语法来识别和分析句子并决定其结构的方式。在计算机自然语言处理中，我们更多地关心句法分析器的算法，因为句法分析器比识别器具有更强的能力，能够提供更多的信息。句法分析算法还应包括其中采用的数据结构的构造，在分析之后如何表示句子的句法结构等各个方面。在通常的人类自然语言中，未经分析的句子是线性的符号串表示。本章将介绍在经过分析后产生的句子结构的树形表示，以及两种表示对于理解句子所带来的差异，也即句子的结构歧义问题。

本章主要明确两个方面的内容，其一是形式语法的表示；其二是句子结构的表示。各部分是这样安排的：1.1 节一般性介绍形式语法的描述问题；1.2 节利用重写规则描述上下文无关语法 在 1.3 节介绍用转移网络和递归转移网络来描述上下文无关语法；在 1.4 节介绍句子在经过句法分析后产生的句法结构的树形表示；最后是对本章的小结。

1.1 形式语法描述

最简单的描述语法的方式是把一种语言中所有可能的句子都列举出来作为这种语言的语法。

这种描述语法的方式其问题是明显的，可以从以下两个方面来看。

第一，在这种语法描述方式下，为了要完成句子识别的任务，即判断一个句子是否符合该语法，也即判断该句子是否是这种语言中的一个合法的句子，就需要列出这种语言中所有可能的句子，这样要判断一个句子是否合乎语法，只需要把该句子和这种语言中的句子逐一比较，看看是否有和该句子完全相同的句子。而通常，我们所使用的语言其句子是无穷多的,无论是对于计算机还是对于人,穷举都是不可能的,对于计算机处理,一个可行的方案是编制一个程序来按某种算法生成并输出这种语言的所有句子，显然，对于有无穷个可能句子的语言而言，这个输出过程是无限的。对于这类语言，有如下的定义：

对于一种语言,如果能编写一部程序,使得能按某种次序输出该语言的所有句子,则称该语言是可递归枚举的。形式语言理论的一个结论是,可递归枚举语言是一种很强的语言，对它的句子进行是否合乎语法的判断并不一定能完全实现。假设给定某种可递归枚举语言,并编写出了一部程序能生成其所有的句子,现在来判断一个句子是否合乎语法,即该句子是否能和程序输出的某个句子完全匹配,如果找到一个完全匹配的句子,那么,可以说该句子是合乎语法的。但是,如果一直没有找到匹配的句子,也不能断定该句子不合乎语法，因为它还可能与后面输出的句子相匹配。由于在句子个数无限多时程序的输出过程是不会终止的,因而它与后面输出的句子相匹配的可能性就一直存在,也即是说,对句子的合法性判断可能不会在有限步骤结束,这对于计算机处理而言,是不可实现的。

可用计算机实现合法性判定的语言应该如下定义：

如果对于一种语言，能编写一个程序在有限步骤内完成上述判断，则该语言称为是可递归的。一种语言是可递归枚举的，却不一定是可递归的。

可见，自然语言句法分析的任务只能在可递归语言上实现，因此，相应的语法描述也应该是可递归的。

第二，如果用列举所有句子的方法作为语法描述，那么这种描述是无助于对新句子进行结构分析的，而只能实现新句子的合法性识别。其方法是通过把新句子与该语法所列举出的句子进行匹配来判定新句子是否来自于这些合法句子的集合中，即是否是一个合法的句子。除此之外，不能得出关于句子结构的进一步信息。

从上述的两点可以看出，用列举句子的方法作为语法描述难以完成对句子结构进行分析的任务。

另外，用列举句子的方法作为语法描述对于解释人类语言构造的方式没有任何帮助，好的语法应该具有推广能力，能够抓住语言现象中的共同点，这对于理解语言、发掘语言运用的认知原理，进而发掘人类思维的本质都具有重要意义。更具现实意义的是，从计算资源的角度来看，具有推广能力的语法更能节约存储空间。

从上面的分析可以看到改进语法描述的某些端倪：它描述的应该是可递归语言，并且

它描述的句子应该是有内部结构的，而且这种内部结构是具有共性的，因而这种语法是有推广能力的。

一个让语法具有推广能力的方法是首先建立一些语法范畴（也常被称为语法类别、词性等),把具有相似语法行为的词归入一个相同的语法类别 然后,描述这些语法类别如何进一步组合的语法行为，这样构造的任何一个语法行为对于具有相似语法范畴的词都具有推广能力。

这时，语法描述就是列出语法类别的所有可能的组合模式。例如：（本章使用几种常用的语法类别,包括 ART(冠词)、N(名词)、V(动词)、ADJ(形容词)、ADV(副词)和 PRON(代词)等。)

$$\mathrm{ART} + \mathrm{N}$$
$$\mathrm{ART} + \mathrm{N} + \mathrm{V}$$
$$\mathrm{ART} + \mathrm{ADJ} + \mathrm{N} + \mathrm{V}$$

就是几个在英语中允许的语言模式，这种以语法类别为单元的模式就比以词本身为单元的列举具有更强的推广能力。例如,上述的

$$\mathrm{ART} + \mathrm{N}$$

模式就可以用来描述诸如:a book,the sentence等等很多个词串。

但是，如果模式有限，每个语法类别中的词有限，则这样的语法可以生成的句子是有限的。然而通过引入几个记号可以大大扩展上述模式的描述能力。

(1) Kleene 星,记为 $*$，例如：

$$\mathrm{ART} + \mathrm{ADJ} + \mathrm{ADJ}^{*} + \mathrm{N} \qquad (1\text{-}1\text{-}1)$$

$*$ 号出现在 ADJ 的右上角,表示 ADJ可以出现 0 次或 0次以上。这样，可以描述在一个冠词和名词之间插有多个形容词的语言模式。

(2) Kleene 加,记为 $+$,例如：

$$\mathrm{ART} + \mathrm{ADJ}^{+} + \mathrm{N} \qquad (1\text{-}1\text{-}2)$$

$+$ 号出现在 ADJ 的右上角,表示 ADJ可以出现 1 次或 1 次以上。这个式子描述的内容与模式(1-1-1)是相同的。

(3) 圆括号,记为(),例如：

$$\mathrm{ART} + (\mathrm{ADJ}) + \mathrm{N} \qquad (1\text{-}1\text{-}3)$$

ADJ 外加一个圆括号表示 ADJ可以出现 1 次,也可以 1 次也不出现。也就是说,ADJ 是可选的。

(4) 垂直线,记为|,例如：

$$\mathrm{N} \mid \mathrm{PRON} + \mathrm{V} \qquad (1\text{-}1\text{-}4)$$

N 和 PRON 中间的直线表示可以是 N,也可以是 PRON,它们都可以与后面的 V 组成这个模式,但二者不能同时出现。

在引入了这几个记号后，基于有限个语法类别的组合模式就可以构造无限多个句子，比如,在模式

$$\mathrm{ART} + \mathrm{ADJ}^{+} + \mathrm{N} + \mathrm{V}$$

中，可以通过无限次重复出现 ADJ 而产生无限多个句子。

这样形成的上述语法描述称为正则表达式。与第一种用列举的方法来作为语法描述

相比，利用正则表达式来描述语法具有了一定的推广能力，并可以利用有限的语法类别的组合模式来生成无限的句子。但是，这种语法描述所表现出的推广能力还是远远不够的，还有进一步改进的余地。例如，上述的模式（1-1-1）和模式(1-1-3）通常会在句子中处于类似的位置，起着类似的作用，因此，应该可以进入更高一层次的抽象。这可以通过简单地定义一个比语法类别具有更高抽象的概念——短语来实现。简单地说，短语是经常反复出现的符号串，它构成确定的语法成分。例如，上述的两个字符串：模式（1-1-1)和模式(1-1-3)可以进一步用一个短语名称 NP（名词短语)来概括。

在引入短语概念的基础上，可以进一步扩展正则表达式的描述能力。如果在正则表达式中，除了可以包含原有的语法类别，还可以包含短语，就可以形成一种描述语言的语法——短语结构语法。

1.2 短语结构语法

为描述短语结构语法，需要先介绍重写规则。重写规则是一种形式化表示方式，可以用来描述规则，例如：

$$S \rightarrow NP\ VP$$

就是一个重写规则。其中，S 代表一个句子；NP, VP 表示两个短语，NP 表示一个名词短语，VP 表示一个动词短语。该规则的意思是说左边的符号 S 所代表的项可以被合乎语法地替换成右边符号所代表的两个项，即被重写为右边两项的组合。

一个形式语法可以包含若干条重写规则。通常一些重写规则的集合用 P 来表示。除此之外，组成一个完整的形式语法还有另外几个要素：其一是所谓终结符号集合，用 T 来表示，一个终结符号代表一个这样的项，它在此语法中不能再被重写为其他项的组合，通常是该形式语法所描述的语言中的词汇的语法类别（如 N, V 等等),或者就是该语言中使用的词汇(如英语中的单词 a, boy 等等)；其二是非终结符号集合，用 NT 来表示，一个非终结符号代表一个这样的项，它在此语法中可能再被重写为其他项的组合，如果上述终结符号指的是语言中的词汇本身，那么非终结符号也包括词的语法类别；其三是一个特殊的非终结符号 S，表示句子。因为句法分析针对的单位均为句子，因而 S 就十分重要，它通常是对句子进行语法分析的开始或结束符号。

这样，一个完整的用来描述一种语言的形式语法就可以表示为四元组（T, NT, S, P)，且 $T \cap NT = \phi$，即一个符号不能同时既是终结符号又是非终结符号。令 $V = T \cup NT$, V^* 表示由 V 中的符号所构成的全部符号串（包括空符号串 ϕ)，而 V^+ 表示 V^* 中除 ϕ 之外的一切符号串的集合。P 中的每条规则形如：

$$a \rightarrow b$$

其中，$a \in V^+$, $b \in V^*$，且 $a \neq b$。

一个简单的例子：

$$NT = \{S, NP, VP, ART, N, V\}$$

$$T = \{the,\ a,\ boy,\ sees,\ cat,\ dirty\}$$

T 中的符号串均为英语单词。

P 中包含如下几条重写规则：

$$S \rightarrow NP\ VP \quad (1\text{-}2\text{-}1)$$

$$NP \rightarrow ART\ N \quad (1\text{-}2\text{-}2)$$

$$NP \rightarrow ART\ ADJ\ N \quad (1\text{-}2\text{-}3)$$

$$VP \rightarrow V\ NP \quad (1\text{-}2\text{-}4)$$

$$ART \rightarrow the \mid a \quad (1\text{-}2\text{-}5)$$

$$N \rightarrow boy \mid cat \quad (1\text{-}2\text{-}6)$$

$$V \rightarrow saw \quad (1\text{-}2\text{-}7)$$

$$ADJ \rightarrow dirty \quad (1\text{-}2\text{-}8)$$

其中，(1-2-5) ~ (1-2-8)表明非终结符号的所属语法类别，四元组(S,NT,T,P) 就表示了一个语法。

利用上述语法，不仅可以进行句子的合法性识别，还可以对一些句子进行结构分析。下面对这种合乎语法的可识别性以及结构分析的内容进行定义。

导出：某个句子被称为由一个语法导出的（一个语法导出了某个句子），如果能由 S 开始依据语法中的一系列重写规则重写出该句子。如果一个句子能由某个语法导出，则称这个句子是合乎该语法的。

看看下面的句子：

$$The\ boy\ saw\ a\ cat. \quad (1\text{-}2\text{-}9)$$

是否合乎上述语法,如果合乎语法,其结构又是怎样的。

(1) 由规则(1-2-1)，合乎该语法的句子都是应该能被重写为一个名词短语加一个动词短语，因此，只需看句子(1-2-9)是否是由这两部分组成的。名词短语和动词短语是非终结符号,还可以进一步分解。

(2) 名词短语是应该能被重写为由一个冠词加一个名词组成的（规则（1-2-2)),或者由一个冠词加一个形容词再加一个名词组成的（规则(1-2-3)) 而动词短语是应该能被重写为由一个动词加一个名词短语组成的（规则(1-2-4))。显然,句子(1-2-9) 中 the boy 可以组成一个名词短语,see a cat 可以组成一个动词短语,而其中 a cat 又是一个名词短语。

(3) 由上述两条，可以看到句子(1-2-9)是能够依据一系列重写规则导出的，规则的使用次序可以如下：

规则(1-2-1),(1-2-2) ,(1-2-5),(1-2-6),(1-2-4),(1-2-7),(1-2-2),(1-2-5),(1-2-6)

同时,得到其组成结构：

(NP1 (The boy) VP1(saw NP2(a cat)))

括号由内向外，反映了句子的组成,The 和 boy 组成一个名词短语 NP1,a 和 cat 组成一个名词短语 NP2,saw 和 NP2 组成一个动词短语 VP1,NP1 和 VP1 组成句子 S。

显然,很多自然语言的句子在上述几条规则(语法)下是不合法的,即有很多自然语言的句子不能由上述语法导出。通过增加重写规则、增加（非）终结符号都可以增加语法能导出的句子的数量。一般地，我们称能生成更多个句子的语法具有更强的生成能力，显然，对语法具有较少约束的语法具有更强的生成能力。

按照上一节定义的可递归枚举语言与可递归语言来分，上述的一般的短语结构语法是可以描述可递归枚举语言的，即某些短语结构语法导出的语言是可递归枚举的而不是可递归的。

通过对一般的短语结构语法进行限制，可以得到被称为乔姆斯基体系的 4 类语法。下面，按照生成能力由弱到强(约束由多到少)的次序分别简单介绍。

1. 正则语法(3 型语法)

正则语法分为左线性语法和右线性语法。

在左线性语法中，所有重写规则必须采用如下的形式：

$$A \to Bt \text{ 或 } A \to t$$

其中，A，B 是非终结符号；t 而为终结符号。

而在右线性语法中，所有重写规则必须采用如下的形式：

$$A \to tB \text{ 或 } A \to t$$

正则语法是乔姆斯基体系中生成能力最弱的一个，一些常见的语言现象都不能用正则语法来生成。一个简单的例子是任意符号“x”两边成对匹配添加括号，通过不断嵌套的方式可以实现一系列句子：

$$x,(x),((x)),(((x))),((((x)))),\cdots$$

为了生成这种语言的句子，当生成到“x”时必须知道前面已经生成了多少个“(”，以便能生成同样数量的“)”相匹配。而对于正则语法，无论是左线性语法还是右线性语法，都只能独立地生成“x”某一侧的符号，无法进行匹配。

在自然语言中，也存在着类似的匹配模式。例如：“如果 A 那么……”“因为 A 所以……”等句子结构(其中 A 表示一个符号串)，通常都需要匹配出现，这种模式也可以进行不断地嵌套形成复杂句子：

如果 A 那么……，如果……如果 A 那么……那么……，……

同样，当生成到 A 时，也必须知道前面已经生成了多少个“如果”，以便能生成同样数量的“那么”相匹配。

2. 上下文无关语法(2 型语法)

在上下文无关语法中，每一条规则都采用如下的形式：

$$A \to x$$

其中，A 是非终结符号，$x \in V^*$。这种规则的应用不依赖于 A 出现在什么上下文环境中，因此称为上下文无关语法。

上下文无关语法比正则语法具有更强的生成能力，能反映更多的自然语言现象。但是，还有一些自然语言现象并不能由上下文无关语法来描述，有些情况下，一条重写规则的应用是受上下文制约的。

3. 上下文有关语法(1 型语法)

在上下文有关语法中，每一条重写规则都是这样的：

$$x \to y$$

其中，$x,y \in V^*$，且 y 的长度(即符号串 y 中的符号个数)总是大于或等于 x 的长度。

上下文有关语法的重写规则也可以这样来表示：

$A \rightarrow y/x\_z$

其中，A 是非终结符号；$y \in V^+$；$x, z \in V^*$，在这种表示中，可以很明显地看出所谓上下文有关的含义来 如果 A 出现在上下文 x_z 中，即前面紧挨着符号串 x，后面紧挨着符号串 z，则 A 可重写为 y，可以看到 A 可重写为 y 是有上下文约束的。

4. 无约束短语结构语法（0 型语法）

0 型语法对规则没有任何约束，其定义的语言可能不是递归的，因而就不可能设计一个程序来判别一个输入的符号串是否是 0 型语言中的一个句子，所以 0 型语言很少被用来处理自然语言。

总而言之，在乔姆斯基体系中，如果一种语言可以被一部 i $(i = 0, 1, 2, 3)$ 型语法所生成，就称它为 i 型语言。

由于在乔姆斯基体系中，语法的型号越高，对重写规则所附加的限制也越多，所以 3 型语言是 2 型语言的一个子集，2 型语言是 1 型语言的一个子集，依此类推，有：0 型语言 $\supseteq$ 1 型语言 $\supseteq$ 2 型语言 $\supseteq$ 3 型语言。从语法的生成能力看，0 型语言最强，1 型到 3 型依次递减，3 型最弱。

在上述乔姆斯基体系的四种语法中，上下文无关语法是计算语言学的重要研究对象。由于其描述能力强，足以描述自然语言中的大部分结构，同时又是可递归的，可以构造有效的句法分析器来进行句子的分析，因此，目前大多数计算机处理用的语法都是基于上下文无关语法的。

1.3 转 移 网 络

除了用重写规则来描述语法之外，转移网络也是一种方式。

转移网络是一个图，图由结点集合和边集合组成，每条边都是带标记的。结点集合中有一个结点是初始状态或称开始状态，还有一个或多个终止状态。

有限状态转移网络是一种较为简单的转移网络。上述乔姆斯基体系中的正则语法可以用有限状态转移网络来等价地描述。图 1-1 是一个有限状态转移网络。

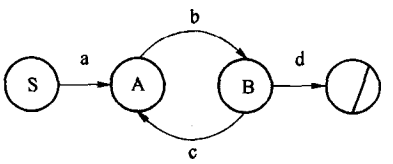

图 1-1 有限状态网络

图中标为 S 的结点是句子分析的起始点，从结点 S 有一条指向结点 A 的有向弧，弧上有一个语法类别标记，意为句子的第一个词如果具有语法类别 a，则 a 可以转移到结点 A 然后如果第二个词具有语法类别 b，则可转移到结点 B 在结点 B 时，有两个有向弧分别指向两个结点，其中标有 d 的有向弧指向的是终止结点（终止结点中标有一斜划线）。如果一个句子能沿着有向弧最终达到终止结点，同时句子也到最后的话，就可以断定该句子是符合该有限状态转移网络所描述的语法。

显然，上述转移网络能用来等价地描述如下的一部正则语法：

$$S \rightarrow aA$$
$$A \rightarrow bB$$
$$B \rightarrow cA$$
$$B \rightarrow d$$

因此，正则语法也可称之为有限状态语法。

对有限状态转移网络进行扩展，可以建立具有更强描述能力的递归转移网络。递归转移网络的描述能力与上述乔姆斯基体系中的上下文无关语法等价。例如，如下一个上下文无关语法：

$$S \rightarrow NP\ VP$$
$$NP \rightarrow ART\ ADJ^{*}\ N\ (PP)$$
$$VP \rightarrow V\ (NP)$$
$$PP \rightarrow PREP\ (NP)$$

可等价地用如图 1-2 中的递归转移网络来描述(PP 表示介词短语),图中每一个单独的网络与上述的一个规则相对应。

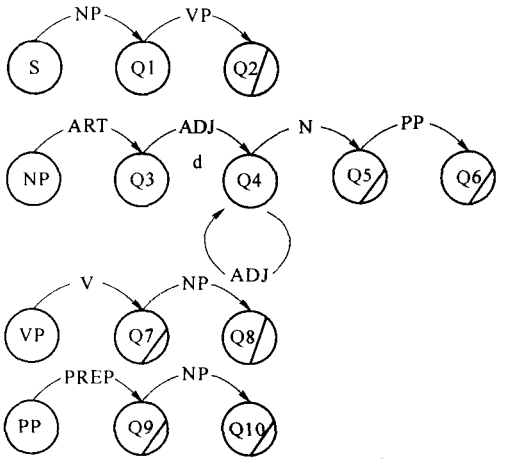

图 1-2 递归转移网络

可以看到，在递归转移网络中，有向弧上的标记不仅可以是某个语法类别，还可以是某个短语结构，这是递归转移网络与有限状态转移网络的一个重要区别所在，这使得递归转移网络能支持递归结构。例如：在图 1-2 的四个网络所构成的递归网络描述中，主网络是起始结点为 S 的网络,一旦在这个网络中沿着有向弧能到达终止结点，则可判定相应的句串是合乎该网络所描述的语法的。而为了从 S结点转移到终止结点 Q2,首先要转移到 Q1 结点。而为了转移到 Q1 结点，首先要判定句子前面是否有一个有向弧上标的结构:NP。而为了判定句子中的 NP结构,需要调用第二个网络,即 NP'网络。在 NP 网络中有两个终止结点，如果在第二个终止结点终止的话,将意味着在 N后面有一个 PP结构。为判定 N 后面是否有一个 PP 结构,需要调用第四个网络,即 PP网络。在 PP网络中若要达到终止结点，首先要找到一个 PREP，而后要判断后面是否有一个 NP 结构，这样反过来又需要调用第二个 NP 网络，如此形成递归调用。

1.4 短语结构与句法树

自然语言，无论是语音形式还是文本形式都表现为线性的，语音在时间上顺序出现，文本在空间上顺序出现。例如，下面以文本形式出现的句子：

I saw a boy.

其中的符号从左到右地顺序出现。

在这种线性描述下，没有任何关于句子构成的信息，无法知道句子是依据哪些语法规则以及如何构成的。而如前所述，一般认为，知道句子组成结构的信息，就能够帮助人们理解句子内容。为此，希望能有一种具有刻画句子语法结构的表示方法，这种表示能描述句子是如何由其子部分构成的。

最常用来描述句子语法结构的表示是树，通常描述语法结构的树称为句法树。树形结构反映了语言中小单元组成大单元，大单元组成句子的递增层次结构，我们可以在句法树中获得在线性结构中所不能得到的句子结构信息。例如，上述句子的句法树如图1-3所示。

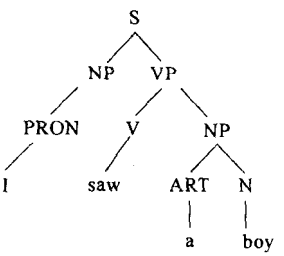

图1-3 句法树

图中所示的树表现了这样一些句法结构信息：整个句子(S)是由一个名词短语(NP)和一个动词短语(VP)构成。而名词短语就包含一个代词(PRON)，这个代词就是I。而动词短语由一个动词和另一个名词短语组成，其中动词是saw，名词短语还有一个内部结构，由一个冠词和一个名词组成，冠词是a，名词是boy。这是一个自顶向下的分析过程。或者也可以相反，自底向上地说，这个句子中，首先a作冠词，boy作一个名词，二者组成一个名词短语，这个名词短语与动词saw一起构成了一个动词短语；而与此同时，I作为代词形成一个名词短语，这个名词短语和动词短语一起构成了整个句子。

从句法树中同时也可以看到一个句子是如何运用多个语法规则组合而成的。如上面的句子中，就是运用了如下的几条语法规则：

S→NP VP
VP→V NP
NP→PRON
NP→ART N
PRON→I
V→saw
ART→a
N→boy

而生成的（其中后4条我们后面通常会归入词典中，而非放在语法中）。其过程如下：

S
⇒NP VP　　（重写 S）
⇒PRON VP　　（重写 NP）
⇒I VP　　（重写 PRON）
⇒I V NP　　（重写 VP）
⇒I saw NP　　（重写 V）
⇒I saw ART N　　（重写 NP）

⇒I saw a N　　(重写 ART)

⇒I saw a boy　　(重写 N)

其中，使用重写规则的次序是可以不同的。比如，也可以先重写 VP 再重写 NP，最终的句子是一样的。这样，在得到一个句子的句法结构树之后，就很容易判定该句子是否合乎语法，这只需要看它在构成句子时所用的规则是否都是该语法中的规则。

本书后面将主要用树作为句法结构的主要描述工具，因而对其中的概念作如下的介绍和规定：

树是图的一种，是由结点集合和边集合组成的，树与其他图的主要差别在于树中不包含任何循环。一条边上端的结点管辖边下端的结点，没有被任何结点管辖的结点称为根结点，而没有管辖任何结点的结点是叶子结点。

除了用句法树之外，也可以通过增加辅助记号，用接近线性的方式来表示和句法树同样多的句法结构信息。例如，上述的句法树可以用如下加括号的方式等价描述：

```
(S ( NP ( PRON I))
   (VP (V saw)
       (NP (ART the)
           (N cat )
       )
   )
)
```

在一些线性结构的句子中，各个词之间的结构关系可能用多个树来描述，例如下面的句子：

I saw a boy with a telescope.

句子中的介词短语结构 with a telescope 有可能是 boy 的后置修饰成分，与 a boy 一起构成一个名词短语；也有可能是谓语动词 saw 的状语，表明该行为所用的工具。

对应于第一种可能和第二种可能，可以分别用两棵不同的句法树来描述，如图 1-4 中的(a)、(b)所示。

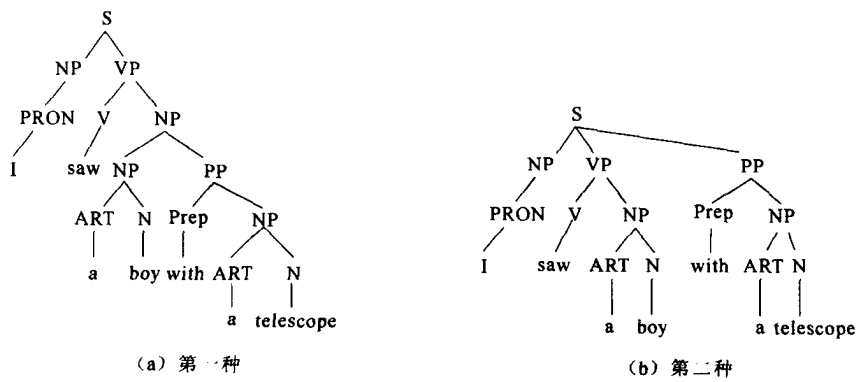

图 1-4　同一个句子的两棵句法树

可以看到，两棵句法树分别揭示了两种可能的句子结构方式。同一个句子存在多个

不同的句法树表明了句子存在结构上的歧义，揭示这种可能存在的歧义，把所有可能的句法结构分析出来正是句法分析的重要任务。

小　结

本章主要介绍了自然语言语法处理中的表示问题。其一是形式语法的表示方法，包括利用重写规则表示语法和用转移网络来表示语法；其二是句子语法结构的表示形式，利用树来表示句子，可以消除线性结构中存在的语法结构上的歧义，这是自然语言处理的一个重要任务。在下面的一章中，将介绍如何基于上述不同的表示下的语法，来得到一个线性结构句子的所有可能的句法树，即如何进行句法分析。

第二章　上下文无关句法分析器

一个句法分析器可以实现以下两个方面的目标：

(1) 确认输入句子是否可以由给定的语法来描述，即输入句子是否合乎给定的语法；

(2) 识别句子各部分是如何依据语法规则组成合法句子，同时生成句法树。

要为某个语言建立一个达到上述目标的句法分析器，需要有两个方面的准备。一方面是该语言的形式化语法描述，规定在该语言中允许出现的结构。其中也包括在该语言各种语法类别中可以出现的所有终结符号，这部分通常以词典的形式单独存储。另一个方面是根据语法来分析句子并决定其结构的方法，即句法分析技术。这两个部分共同组成一个完整的句法分析器。

本章是这样安排的，首先简单考察构造句法分析器的第一个方面，即如何构造一个好的语法；之后主要的内容是基于一个简单的语法来介绍几种句法分析算法。

由于本章的句法分析算法所依据的语法都是上下文无关语法，因此，本章的句法分析器均为上下文无关句法分析器。

2.1　语　　法

第一章介绍了如何形式化描述一个语法系统，包括利用重写规则和转移网络来描述语法规则。但是并没有提到这些语法规则本身是如何构造的，因为这是与语言相关的。我们可以限定一个语法系统是上下文无关语法，但它也只是对每条语法规则的形式具有约束，而对每条语法规则中具体内容的确定没有帮助。

一般在为一个语言构造语法的时候，最感兴趣的有三个方面：语法的推广能力，这是指能被语法正确分析的句子范围；选择能力，即能被语法识别出问题的非句子的范围；可理解性，即语法本身的简单性。

现有语法书中的语法系统，可以作为一个有益的借鉴，它是语言学家们长期研究语言现象所得出的一些规律性的结论，其中的很多规则是具有上述三个方面的特点的。但已有的实践表明，由于自然语言的句子是无限的，而且处于不断地增长和更新中，语言中的新现象、例外现象、歧义现象等不断出现，所以单单利用语法书中的规则难以精确覆盖一种自然语言的所有语言现象。

对于复杂的语言现象，一种简化可行的研究方式是对要研究的语言对象进行一些限定，比如排除一些例外、排除一些不常用的词汇等等，或者根据语言使用的领域的特点进

行规则的限定等等。总之，是在庞大语言现象的一个限定子集上构造语法，这样就可能使上述三个方面都达到较好的性能。实际上，人们在日常高频使用的一般也就是所有语言现象的一个子集,所以,这种方法是有实际意义的。

在本章以下的部分，为了解释句法分析的算法，均采用一个十分简单的语法，它只包含如下几条重写规则：

S→NP VP (2-1-1)

VP→V (2-1-2)

VP→V NP (2-1-3)

VP→AUX VP (2-1-4)

NP→PRON (2-1-5)

NP→ART N (2-1-6)

NP→ART ADJ N (2-1-7)

NP→ADJ N (2-1-8)

PRON→I (2-1-9)

V→saw (2-1-10)

ART→a (2-1-11)

N→boy (2-1-12)

其中 AUX 表示系动词，规则(2-1-9) ~ (2-1-12) 通常放在词典中，余下的几条语法规则(2-1-1) ~ (2-1-8)在后面统称为语法 2.1。此外，在本书后面的例子中，如果出现其他的词，都假设其词汇类别信息已经存放在一个可用的词典中了。

2.2 基于符号串的句法分析

一个句法分析算法可以表述为一个搜索过程，其搜索空间是语法规则，搜索过程就是检查各种语法规则所有可能的组合方式，搜索目的是最终找到一种组合，其中的语法规则能够生成一棵用来表示句子结构的句法树。通常在算法中，句法树都不是被显式地构造出来，而是隐含在所搜索出的语法规则的组合序列中，通过这个语法规则的序列，可以很容易构造出相应的句法树。我们会在后面的例子中看到这一点。搜索算法同时也可以确定该输入句子是否是合乎语法的。

句法分析的搜索过程可以有两个相反的方向，一个是自顶向下；另一个是自底向上。本节先介绍自顶向下的方法，在下一节介绍自底向上的方法。

自顶向下的分析方法是从符号(S)开始，S 称为这种句法分析的初始状态。算法试图通过搜索并应用语法中的重写规则来不断改变算法的状态序列，直到最终生成与输入句子的词汇类别序列相匹配的符号序列，就可以断定该输入句子是合乎语法的，并且最终所用到的那些重写规则序列就蕴涵了句子的句法结构。或者，当所有可能性都尝试后还不能生成输入的句子，则可以断定该输入句子不能由该语法分析，或该句子在该语法下是不合法的。

在算法进行的任何时刻，算法的分析状态都可以表示为一个符号列表，这个列表通常

称为符号串。例如，初始状态的符号串为（S），在应用重写规则 S→NP VP 后，状态序列符号串就变成了（NP VP），对于其中的 NP，可以进一步再用规则 NP→ART N，这时状态序列符号串为（ART N VP）。当然这时候也可以用规则 NP→ART ADJ N，则符号串为（ART ADJ N VP）等等。过程将一直进行到符号串完全由终结符号（在句法分析时，一般把词汇类别作为终结符号）组成，然后检查是否与输入句子的词汇类别序列相匹配。但是这种方式是十分浪费的，因为在上述的方案中，每一条路径都要等到一次搜索走到尽头（符号串完全重写为终结符号）后才能判断其对错。而实际上，大量的错误路径通常在完成几个较早的搜索步骤后就可以判定出来，无需进一步试探该路径，从而可以减少后面无用的搜索步骤。而错误发现得越早，可以节省的搜索量就越多。因此，一个好的算法就是要能尽早发现搜索路径中的错误，在由语法生成可能的词类序列的同时，不断与输入句子可能的词汇类别序列对比，在这个过程中不断尽早排除一些不可能的结构，直到最终完全相符。

下面介绍一个简单的自顶向下的句法分析算法，例子中采用语法 2.1，并假设所需的词汇类别信息均已在另外的词汇类别词典中给出。

为描述算法，需要做以下一些准备工作。

引入位置标记，包括两个方面的标记，一方面是在输入句子上的标记，即对于给定的输入句子中的词汇按次序标记其所在的位置。例如，对于句子：

The boy cried.

加上位置标记为：

$_1$The $_2$boy $_3$cried$_4$

每个词前面的下标数字表示该词的位置，而句子最后一个下标数字标志句子的结束。在上面的句子中，单词 The 在句子的位置是 1，达到位置 4 就表明句子已经结束。通过引入标记，句法分析程序可以判定其当前处理的词是哪一个以及是否已完成对一个句子的处理。在句子上的这种标记在句子输入后就可以立即完成。对句子的位置标记在后面的其他部分还会用到，本书将一直沿用这里的约定方法。

另一方面，句法分析过程中产生的符号串也包含某种标记。例如，在分析过程中有这样一个符号串：

((N VP)2)

该符号串后面的标记 2 表明算法对输入句子的第一个词的分析已完成，分析算法已经进入到对第二个词的匹配 同时，由前面的（N VP）部分可知 算法期望第二个词的句法类别是 N，并且在其后是一个 VP。

符号串在分析过程中是不断更新变化的，其改变状态的操作根据当前符号串的第一个符号是否是词汇类别符号而不同。

如果是词汇类别符号（例如在上述的符号串（(N VP)2）中第一个符号为 N，就是一个词汇类别符号），那么就继续检查此时符号串后面的数字标记所对应的句子中的单词是否属于该词汇类别（例如在上例中，符号串中数字 2 对应的单词为 boy，其词汇类别为 N），如果是，就可以把该词汇类别从符号串中删去，生成新符号串（上例中把 N 去掉后符号串就剩下（VP）），同时位置标记加 1（综合起来得到新的状态序列为（(VP)3））。

如果不是词汇类别符号（例如上述新生成的符号串（VP），VP 不是词汇类别符号），那么就从语法规则中寻找所有可能的规则来重写该符号（例如在语法 2.1 中可以用来重写

符号 VP 的规则有规则(2-1-2),(2-1-3)和(2-1-4),如果有多个可用的规则,那么就会产生多个可能的新符号串(例如用规则(2-1-2),(2-1-3)和(2-1-4),产生的新符号串分别为(V),(V NP)和(AUX VP)),这时先取一个符号串,其余未用的符号串要保存以备回溯之用。所谓回溯就是如果用先选的符号序列在后面的分析中得不到句子的句法结构,那么就退回到有多个选择的地方,选择未使用的其他可能的符号串来进行下一步的分析。

下面举例说明这个算法是如何分析一个句子的。句子:

I saw a boy.

显然在进行算法之前,需要对句子加位置标记:

$_1$I $_2$saw $_3$a $_4$boy$_5$

算法过程如表 2-1 所示。

表 2-1 句子 I saw a boy 的算法过程

步骤	当前状态	备份状态(用于回溯)	步骤	当前状态	备份状态(用于回溯)
1	((S)1)		7	((V NP)2)	
2	((NP VP)1)		8	((NP)3)	
3	((PRON VP)1)	((ADJ N VP)1) ((ART ADJ N VP)1) ((ART N VP)1)	9	((PRON)3)	((ADJ N)3) ((ART ADJ N)3) ((ART N)3)
4	((VP)2)		10	((ART N)3)	
5	((V)2)	((V NP)2) ((AUX VP)2)	11	((N)4)	
6	(3)		12	(5)	

说明:

第一步是自顶向下句法分析算法的初始状态,符号串为(S),分析算法处于句子开头,位置标记为 1。

第二步判定符号串。第一个符号显然是非词汇类别符号,因此搜索语法中可用来重写该符号的规则。在语法 2.1 只有规则(2-1-1)可以,因此依据语法规则更新符号串为(NP VP)。

第三步同样判定后,语法中有四条规则(2-1-5),(2-1-6),(2-1-7)和(2-1-8)可以重写 NP,先用规则(2-1-5)重写 NP 产生的新状态,其余三种可能的状态按次序保存,在回溯时使用。

第四步,由于此时符号串的第一个符号为词汇类别符号 PRON,因此可以按后面的标记位置找到句子中的对应的词。在例子中 1 号位置为 I,它的词汇类别恰好是 PRON,这样可以把该符号从符号串中去掉得到新的符号串;同时,后面的位置标记加 1,即得到第四步给出的状态序列((VP)2)。

第五步,在语法 2.1 中可以找到三条规则重写 VP,先使用第一条规则(2-1-2),使用另两条规则产生的可能状态保存以备回溯用。

第六步和第四步类似,由于符号串的第一个符号已是词汇类别符号 V,句子的 2 号位置为 saw,其词汇类别是 V,则去掉已匹配的符号,得到新的符号串。此时的符号串为空,即按所用的语法规则来看,句子应该结束了,但后面的位置标记按算法是加 1,即为 3,这表明对句子的分析才进行到位置 3,而没有到句子结束的位置 5。因此可以断定前面的语

法规则的使用有问题，需要回溯。

第七步，回溯就是从最近保存的可能状态按照后进先出的原则取出来，这里最近的保留状态是在第五步保存的((V NP)2)。

第八步与第六步的前面部分一样，从符号串中去掉 V,但此时的符号串不为空,而是有(NP),位置标记加 1,得到新的状态序列((NP)3)。

第九步和第三步完全一样，依据语法 2.1 可以产生四种可能的状态，先取规则(2-1-5)所产生的状态，其余三种状态按次序保存，在回溯时使用。

第十步之前省略了一个和第六步一样的分析，即按所用语法规则句子应该结束了，但位置标记却没有达到句子结束位置，因此要回溯，把在第九步最后保留的一个状态取出来,按后进先出原则,选用((ART N)3)。

第十一步，由于此时符号串的第一个符号为词汇类别 ART,回到句子的第 3 个位置，为 a,其词汇类别为 ART,从符号串中删去 ART,位置标记加 1,为 4。

第十二步，此时符号串的第一个符号为词汇类别 N,回到句子的第 4 个位置,为 boy.其词汇类别为 N,从符号串中删去 N。此时符号串为空,位置标记为 4 + 1 = 5,语法规则的使用和句子的位置标记均标识了句子的结束，因此，该句子的分析完成。

在上述的搜索过程中，省略掉被回溯的规则，最终使用的重写规则及其使用次序如下，它们构成了一条可以最终生成句子的路径。

S
⇒NP VP　　(重写 S)
⇒PRON VP　　(重写 NP)
⇒I VP　　(重写 PRON)
⇒I V NP　　(重写 VP)
⇒I saw NP　　(重写 V)
⇒I saw ART N　　(重写 NP)
⇒I saw a N　　(重写 ART)
⇒I saw a boy　　(重写 N)

按照上述句子真正使用的句法规则集及其使用的顺序，就可以获得其句法树，如第一章图 1-3 所示。

现在，可以把上述算法一般地描述如下，其中用堆栈来保存供回溯之用的状态序列，称为回溯栈。

第一步，初始化状态序列为((S)1)，回溯栈为空。

第二步，选择当前状态序列：

```
if(该状态序列为空)
{
  if(状态序列的位置标记是句子的最后位置)
    then{算法停止(成功地进行了句法分析)}
  else (状态序列的位置标记不是句子的最后位置)
    {
      if (回溯栈为空)
```

then｛算法停止（没有成功的进行句法分析）｝
else(回溯栈不为空)
then｛从回溯栈弹出一个状态序列作为当前状态｝
｝
｝
else（该状态序列不为空）
then｛从状态序列中取出第一个状态，把它称为 C｝
第三步，处理当前状态序列的第一个符号：
if(C 是终结符号)
｛
if(当前状态序列的位置标记在句子中的下一个词可以是该终结符号类)
then｛把 C 从当前状态序列中去掉，得到一个新的当前状态序列，位置标记 + 1，转到第二步｝
else（当前状态序列的位置标记在句子中的下一个词不可能是该终结符号类）
｛
then｛从回溯栈弹出一个状态序列作为当前状态｝
if （回溯栈为空）
then｛算法停止（没有成功的进行句法分析）｝
｝
｝
else(C 不是终结符号)
｛
then｛按重写规则重写该符号，并替换进当前状态序列，生成新的当前状态序列｝
if(有多个可应用的重写规则)
then｛取任意一个，而把其他几个压入回溯栈，转到第二步｝
｝

对于自顶向下的句法分析过程，要避免出现左递归的问题。如果在语法中有类似如下的直接递归(左递归)的重写规则

$$VP \rightarrow VP\ NP$$

就有可能出现无穷循环的问题。即当使用上述规则重写 VP 时,替换后的符号仍含有 VP,则又可以用上述规则进行重写，这个过程可以无限进行下去，这就是所谓的左递归规则的问题。对于这个问题，一种解决办法是规定在使用一次左递归规则后，就要用非左递归规则来代替它。例如可以用

$$VP \rightarrow V\ NP$$

来重写 VP。

上面的算法实际上是一个深度优先的搜索过程。在搜索过程的每一步，都有一个期望，只有当前一步的期望实现后，才能进行下一步。这个过程是顺序进行的。

与此不同的另一种搜索策略是广度优先。在图 2-1 中,标记了对同一个句子用两种搜索方式进行分析时，状态序列出现的次序。图中，每个状态左边的数字是按深度优先来