《当代语言学》第2卷2000年第2期84-98页，北京

基于短语结构语法的自动句法分析方法

冯志伟　　语言文字应用研究所

提要　短语结构语法在计算语言学中得到广泛的应用。本文介绍了基于短语结构语法的自动句法分析方法，主要有自顶向下分析法、自底向上分析法、富田算法、左角分析法、CYK算法等。

关键词　短语结构语法　上下文无关语法　乔姆斯基范式　自顶向下分析法　自底向上分析法　移进－归约算法　富田算法　左角分析法　CYK算法

计算语言学分析自然语言的方法主要有两种：一种是基于规则的方法，一种是基于统计的方法。实践证明，这两种方法各有千秋。尽管基于统计的方法对于大规模真实文本的处理比较适合，但是我们绝不可忽视基于规则的方法。本文中介绍的基于短语结构语法的自动句法分析方法，是基于规则的方法的一个重要方面。

乔姆斯基的形式语法G是一个四元组

$$G = (Vn, Vt, S, P)$$

其中，Vn是非终极符号的集合，Vt是终极符号的集合，S表示句子，P是重写规则，P的形式为

$$A \rightarrow \omega$$

这里，A是单个的非终极符号，ω是符号串，它可以由终极符号组成，也可以由非终极符号组成，或者由终极符号和非终极符号共同组成。这样的重写规则是上下文无关的，具有这样的重写规则的语法叫做上下文无关语法(context-free grammar)，也叫做上下文无关的短语结构语法(context-free phrase structure grammar)，或者叫做短语结构语法(phrase structure grammar，简称PSG)。在本文中，这三个术语的含义是一样的。

基于短语结构语法的自动句法分析方法，主要有自顶向下分析法、自底向上分析法、富田算法、左角分析法、CYK算法等。分别介绍如下。

1. 自顶向下分析法(top-down parsing method)

根据重写规则，从初始符号开始，自顶向下地进行搜索，构造推导树，一直分析到句子的结尾为止。例如，我们提出这样的文法：

G = {Vn, Vt, S, P}

Vn = {S, NP, VP, Det, N, V, Prep}

Vt = {the, boy, rod, dog, hits, with, a}

S = S

P: S → NP VP　　(a)

NP → Det N　　(b)

VP → V NP (c)

VP → VP PP (d)

PP → Prep NP (e)

Det → [the] (f)

Det → [a] (g)

N → [boy] (h)

N → [dog] (i)

N → [rod] (j)

V → [hits] (k)

Prep → [with] (l)

在搜索过程中，搜索目标首先是初始符号S，从S开始，选择文法中适用的规则来替换搜索目标，并用文法规则的右边部分同句子中的单词相匹配。如果匹配成功，则抹去这个单词，在搜索目标中，记录下有关规则，然后，继续对输入句子中的遗留部分进行搜索；如果分析到句子的结尾，搜索目标为空，则分析成功。英语句子“the boy hits the dog with a rod”(那个男孩儿用一根棍子打狗)分析过程如下(向下的箭头“↓”旁边注明规则的号码，搜索目标中有下划线的符号表示它是所使用规则的左边部分)：

	搜索目标	输入句子中遗留部分
(1)	S	the boy hits the dog with a rod
	↓ (a)	
(2)	NP VP	the boy hits the dog with a rod
	↓ (b)	
(3)	Det N VP	the boy hits the dog with a rod
	↓ (f)	
(4)	N VP	boy hits the dog with a rod
	↓ (h)	
(5)	VP	hits the dog with a rod
	↓ (c)	
(6)	V NP	hits the dog with a rod
	↓ (k)	
(7)	NP	the dog with a rod
	↓ (b)	
(8)	Det N	the dog with a rod
	↓ (f)	
(9)	N	dog with a rod
	↓ (i)	

如果根据规则(i)抹去dog，句子中还有遗留部分with a rod，而这时搜索目标已经变空，分析无法继续进行下去，因而回溯到第(5)步，看一看是否还能利用别的规则进行分析。

(5') VP hits the dog with a rod

↓ (d)

(10) VP PP hit the dog with a rod

↓ 重复进行从(5)到(9)的搜索过程，此时，搜索目标为PP

(11) PP with a rod

↓ (e)

(12) Prep NP with a rod

↓ (l)

(13) NP a rod

↓ (b)

(14) Det N a rod

↓ (g)

(15) N rod

↓ (j)

(16) 结束，分析成功

在搜索过程中，当分析到第(5)步时，由于使用了规则(c)，使得分析无法继续进行下去，于是，我们采用了回溯(backtracking)的办法，回到第(5)步，不再使用规则(c)而改为使用规则(d)，从而使搜索得以成功。

根据这样的搜索过程，我们可以把这个英语句子“the boy hits the dog with a rod”的推导树表示如图1所示：

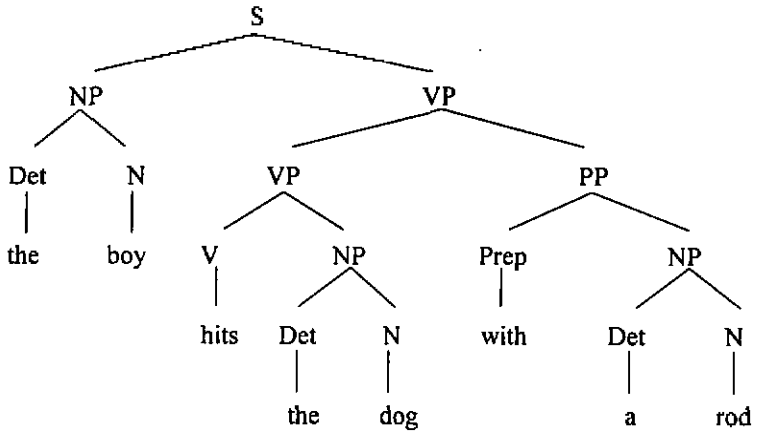


图1 the boy hits the dog with a rod

2. 自底向上分析法(bottom-top parsing method)

从输入句子的句首开始顺次取词向前移进(shift)并根据文法的重写规则逐级向上归约(reduce)，直到构造出表示句子结构的整个推导树为止。

自底向上分析法实际上是一种“移进-归约算法”(shift-reduce algorithm)，它对句子中的单词取词是顺次“移进”，而在利用文法中的重写规则时是按条件“归约”。这种算法类似于编译技术中的LR算法(自左(Left)向右(Right)算法)。移进-归约算法的信息存放方式主要是“栈”(stack)，信息操作方式主要有移进、归约、拒绝、接受。这种算法利用一个栈来存放分析过程中的有关“历史”信息(即关于已经走过的过程的信息)，并且根据这种历史信息和当前正在处理的符号串来决定究竟是移进还是归约。所谓“移进”，就是把一个尚未处理过的符号移入栈顶，并等待更多的信息到来之后再做决定；所谓“归约”，就是把栈顶部分的一些符号，由文法的某个重写规则的左边的符号来替代。这时，这个重写规则的右边部分必须与栈顶的那些符号相匹配。用这样的办法对栈中的符号以及输入符号串进行移进和归约两种操作，直到输入的符号串处理完毕并且栈中仅仅剩下初始符号S的时候，就认为输入符号串被接受。如果在当前状态，既无法进行移进，也无法进行归约，并且栈并非只有唯一的初始符号S，或者输入符号串中还有符号未处理完毕，那么，输入符号串就被拒绝。

在某个时刻，往往既可以进行移进操作，又可以进行归约操作，这种情况，称之为“移进-归约冲突”，简称“移归冲突”；在某个时刻，往往会有多个规则都能满足归约条件，这种情况称之为“归约-归约冲突”，简称“归归冲突”。什么时候进行移进操作，什么时候进行归约操作，怎样定义归约的条件，这些问题是移进-归约算法的中心问题。

下面，我们以“the boy hits the dog with a rod”为例子，说明采用这种移进-归约算法的自底向上分析法分析句子的过程。

	栈	操作	输入句子中的遗留部分
(1)			the boy hits the dog with a rod
(2)	the	移进	boy hits the dog with a rod
(3)	Det	用规则(f)归约	boy hits the dog with a rod
(4)	Det boy	移进	hits the dog with a rod
(5)	Det N	用规则(h)归约	hits the dog with a rod
(6)	NP	用规则(b)归约	hits the dog with a rod
(7)	NP hits	移进	the dog with a rod
(8)	NP V	用规则(k)归约	the dog with a rod
(9)	NP V the	移进	dog with a rod
(10)	NP V Det	用规则(f)归约	dog with a rod
(11)	NP V Det dog	移进	with a rod
(12)	NP V Det N	用规则(i)归约	with a rod
(13)	NP V NP	用规则(b)归约	with a rod

(14)	NP VP	用规则(c)归约	with a rod
(15)	S	用规则(a)归约	with a rod
(16)	S with	移进	a rod
(17)	S Prep	用规则(l)归约	a rod
(18)	S Prep a	移进	rod
(19)	S Prep Det	用规则(g)归约	rod
(20)	S Prep Det rod	移进	
(21)	S Prep Det N	用规则(j)归约	
(22)	S Prep NP	用规则(b)归约	
(23)	S PP	用规则(e)归约	

这时，文法中不再有适合的规则，分析无法进行，于是返回到(14)，先不采用规则(a)归约，而是移进下一个单词 with，然后采用规则(l)归约。

(24')	NP VP	回溯	with a rod
(24)	NP VP with	移进	a rod
(25)	NP VP Prep	用规则(l)归约	a rod
(26)	NP VP Prep a	移进	rod
(27)	NP VP Prep Det	用规则(g)归约	rod
(28)	NP VP Prep Det rod	移进	
(29)	NP VP Prep NP	用规则(b)归约	
(30)	NP VP PP	用规则(e)归约	
(31)	NP VP	用规则(d)归约	
(32)	S	用规则(a)归约	

这时，栈中只剩下初始符号 S，而且输入符号串为空，该句子被接受，分析成功。

在英语中，介词短语 PP 不仅可以修饰动词，而且还可以修饰名词。上面的文法只考虑了 PP 修饰动词的情况，没有考虑 PP 修饰名词的情况。如果我们在文法重写规则中，再加上一条新的规则：NP→NP PP，那么，除了得到上面的分析结果之外，还可以按别的分析进程得到另一个结果，其树形图如图 2 所示。

这个结果在语义上虽然不十分完善(其意思是“那个男孩儿打身上带着棍子的狗”，“狗”身上带着“棍子”，这种情况不太多见)，但在句法结构上是合格的。

如果我们按照这个句子的结构把句子中的单词换成“the man saw a boy with a telescope”，那么，这个句子显然有两种不同的意思：一个意思是“那个人看见一个带着望远镜的男孩儿”，一个意思是“那个人用望远镜看见一个男孩儿”。这就产生了歧义。如果我们在介词词组 PP“with a telescope”的前面再加上一个介词词组 PP“in the park”，造出新句子“the man saw the boy in the park with a telescope”，那么这两个 PP 可以同时修饰名词 boy，也可以同时修饰动词 saw，也可以

第一个PP修饰名词boy，第二个 PP修饰动词saw，还可以第一个PP修饰动词saw，第二个PP修饰名词boy，从而得到4种不同的结构，相应地得到4种不同的意义："那个人看到一个在公园中带着望远镜的男孩儿"，"那个人在公园中用望远镜看见一个男孩儿"，"那个人用望远镜看见一个在公园中的男孩儿"；"那个人在公园中看到一个带着望远镜的男孩儿"。从纯粹句法的观点看来，第一个PP中的名词park还可以受第二个PP"with a telescope"的修饰，其含义是"装有望远镜的公园"。如果把这种情况也算进去，那么，还可以得到第五种结构，相应地得到第五种不同的意思："那个人在装了望远镜的公园中看见一个男孩儿"。

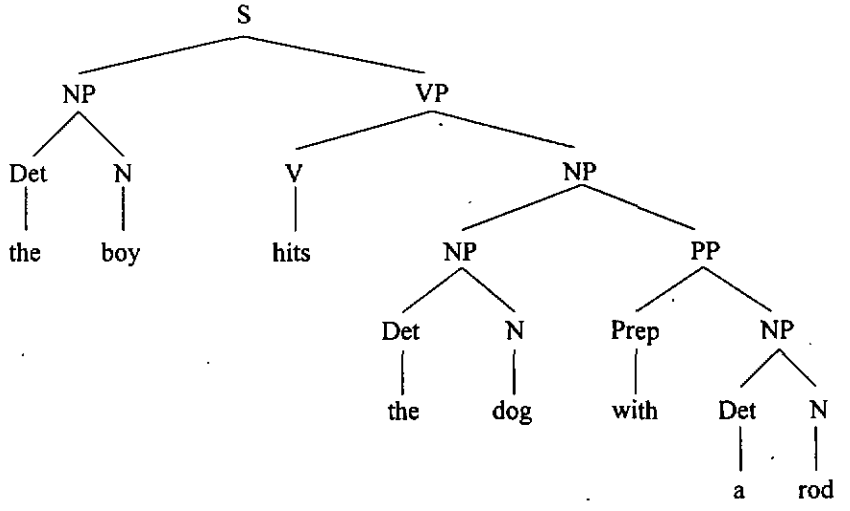


图2 移进-归约算法树形图

这样一来，分析的结果将是有歧义的。歧义是自然语言中普遍存在的现象，移进-归约算法在本质上是一种自左向右的LR算法，标准的LR算法最初是为程序语言设计的，效率虽然较高，但在处理自然语言的歧义问题时却进退维谷。因此，用基于LR算法的普通的移进-归约分析技术难以处理自然语言中的歧义问题。

3. **富田算法**(Tomita algorithm)

美国卡内基-梅隆大学的计算语言学家富田胜(M. Tomita)于1985年提出富田算法(Tomita algorithm)，这是一种扩充的LR算法，也是一种基于上下文无关的短语结构语法的高效的自然语言分析算法。富田胜在这种算法中，引入了图结构栈、子树共享和局部歧义紧缩等技术，提高了算法的效率。

图结构栈是由栈表技术、树结构栈技术发展而来的。使用栈表技术时，对进程的操作是并行进行的，每一个进程对应于一个栈，每一个进程的动作与标准的LR分析一样。栈表技术的缺点是各个进程之间没有关系，任何一个进程都无法利用其他进程已经做过的分析结果，而且当出现歧义时，栈表数目会呈指数增长。

为了克服栈表技术的缺点，引入了树结构栈。树结构栈的具体做法是：如果几个进程处于相同的状态，那么，这几个栈的工作就会一样，直到进行到某一时刻，该栈顶顶点被某一归约动作

弹出。为了消除冗余，可以把这几个进程归结为一个进程，只要在几个进程之间，对应的栈顶顶点具有相同的状态，就将这几个进程合并。这时，这些栈就变成树形结构，树的根结点便是栈的顶点，所以叫做树结构栈。在树结构栈中，当栈顶被弹出时，树结构栈又会分解为原来的几个栈。实际上，系统可能会并行地存在几组树结构栈，因此，系统中的栈从总体来看构成了一个森林。

尽管树结构栈可以大大地缩减计算量，但是，树结构栈的枝干数目仍然会随着歧义的增加而呈指数上升。为了解决这个问题，富田胜提出了“图结构栈”。采用树结构栈技术，当栈分裂时，要将整个栈复制若干个。但在实际上，不一定整个栈都复制，只要将栈的某些部分分裂一下就可以了。当一个栈分裂时，就被表示为一棵树，栈底对应于树的根。利用栈合并技术，可以将栈表示为有向无圈图，这样，就形成了图结构栈。采用图结构栈技术，富田算法不会对输入句子的任何部分以同样的方式做两次或两次以上的分析。这是因为，如果两个进程以同样的方式分析句子的某一部分，那么，这两个进程就会处于同样的状态，就一定会被组合成一个进程。由于采用了图结构栈，分析算法的空间复杂度和时间复杂度都大大地降低了。

一个自然语言的自动分析系统应该能够求出歧义句子的所有分析结果，并且将它们以合理的方式表达和存储起来，以便在后面的阶段进行歧义消解处理。但是，由于分析时所得到的歧义句子的总数(它们形成“分析森林”)可能随着句子长度的增加而呈指数增长，即使是输出所有的分析结果，也得花费指数时间的代价。因此，必须有效地表达句法分析的结果，使得分析森林的规模不会呈指数增长。为此，富田胜提出了“子树共享”和“局部歧义紧缩”。

所谓“子树共享”，就是指如果几棵树存在一个共同的子树，那么，这样的子树只表达一次，构成一个“共享森林”。例如，英语句子“I saw a boy with a telescope”(“我看见一个带着望远镜的男孩儿”或者“我用望远镜看见一个男孩儿”)这个歧义句子，其共享森林可表示为图3：

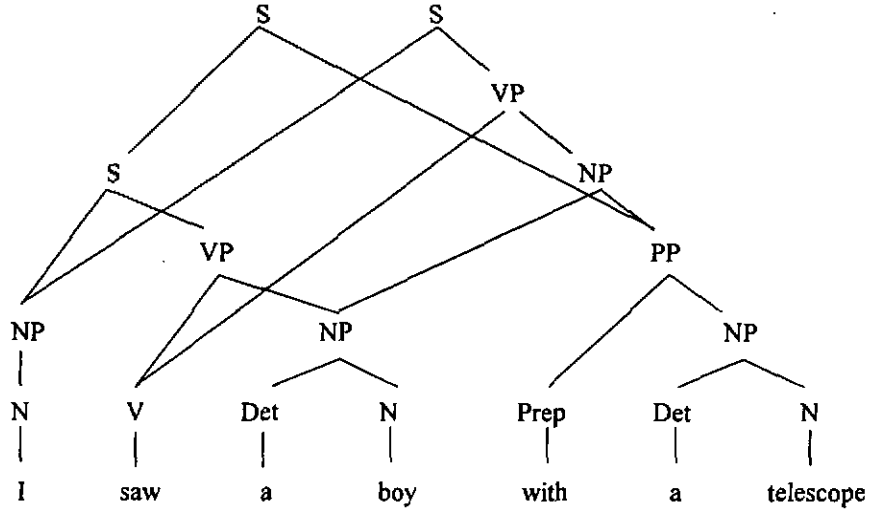


图3 歧义共享森林树形图

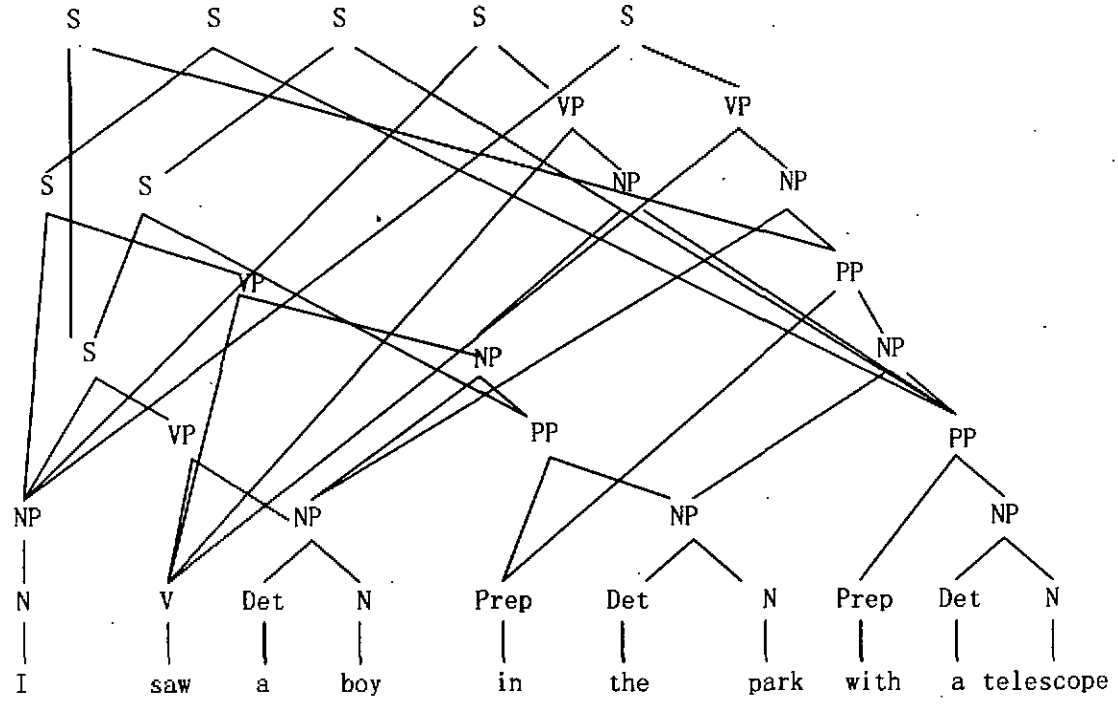


图4 紧缩前共享森林树形图

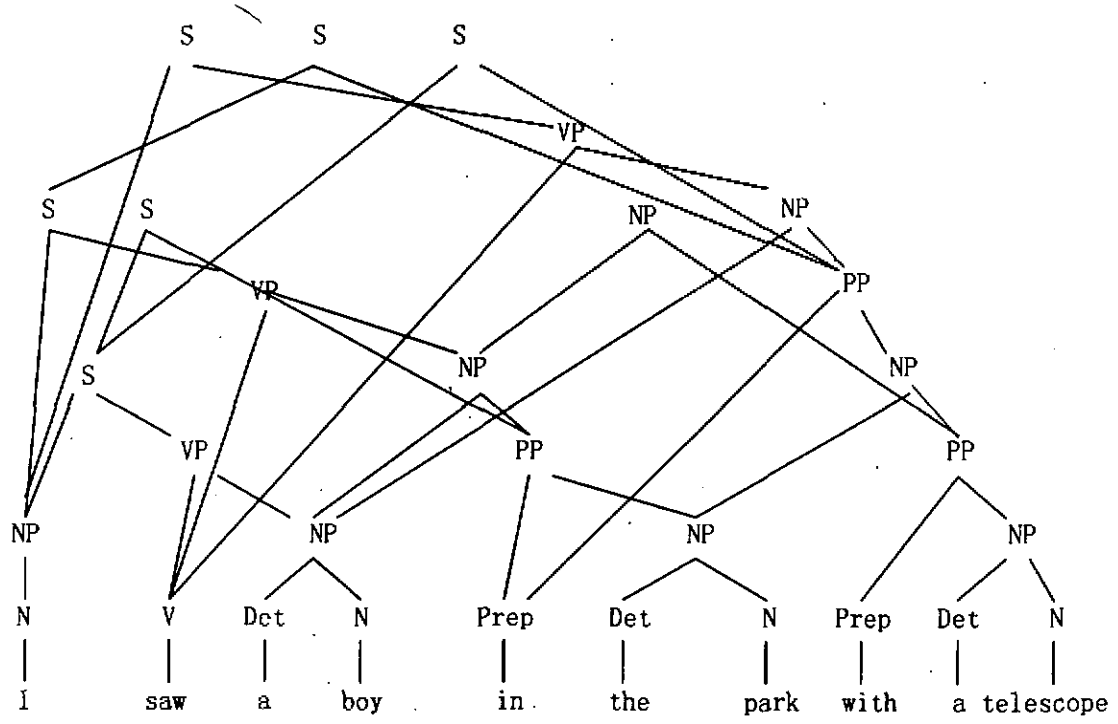


图5 紧缩后共享森林树形图

为了实现“子树共享”的技术，不再将具体的语法范畴符号放到栈上，而只在栈中存放指向共享森林某一结点的指针。当分析器移进某一个单词时，它就会产生一个叶结点，这个叶结点的标记是该词及其词性，此时不是直接将该词及其词性压入栈中，而是只压入一个指向这个新创结点的指针。如果已经有一个相同的叶结点存在，则把一个指向已经存在的结点的指针压入栈中，就不再为它创建新的结点了。当分析器在栈中归约时，只须从栈中弹出相关结点的指针，创建一个新的结点，这个新结点的后继结点，就是这些弹出的指针所指的结点，然后，再将一个指向这个新创建的结点的指针压入栈中。

当两个或两个以上的子树具有相同的结点，并且这几棵子树的根具有相同的非终极符号时，这若干个子树就构成一个“局部歧义”。如果一个句子的局部歧义太多，那么，这个句子的总的歧义数量就会呈指数增长。因此，需要进行“局部歧义紧缩”。

局部歧义紧缩的工作方式如下：当出现局部歧义时，表达局部歧义的子树的根就被合并为一个结点，这个结点叫做“紧缩结点”，而合并之前的子树的根结点，叫做这个紧缩结点的子结点。在图结构栈中，如果两个或两个以上的符号顶点具有相同的状态顶点在其左边，又有相同的状态顶点在其右边，就表示存在一个局部歧义，因此，这些符号顶点指向的结点就会被紧缩为一个结点。例如，“I saw a boy in the park with a telescope”这个句子紧缩之前和紧缩之后的共享森林如图4、图5所示(可以看出，紧缩之后结点数目大大减少了)。

富田算法对于自然语言的分析效率很高，在自然语言的计算机处理中得到广泛的应用。

4. 左角分析法 (left-corner parsing method)

左角分析法是一种把自顶向下分析法和自底向上分析法结合起来的分析法。所谓“左角”是指表示句子句法结构的树形图的任何子树(subtree)中左下角的那个符号。例如，在表示句子“the boy hits the dog with a rod”的树形图中，the是Det的左角，Det是NP的左角，NP是S的左角，hits是V的左角，V是VP的左角，with是Prep的左角，Prep是PP的左角。从重写规则的角度来看，“左角”是重写规则右边部分的第一个符号。如果重写规则的形式是A→BC，则B就是左角。

重写规则A→BC可以表示为图6：

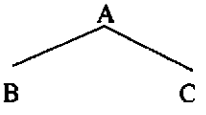


图6

如果采用自顶向下分析法，其分析过程应该是A→B→C，是先上后下；如果采用自底向上分析法，其分析过程应该是B→C→A，是先下后上；如果采用左角分析法，其分析过程就应该是B→A→C，是有下有上。把数码记在相应的结点上，这三种分析法的分析顺序可以表示为图7：

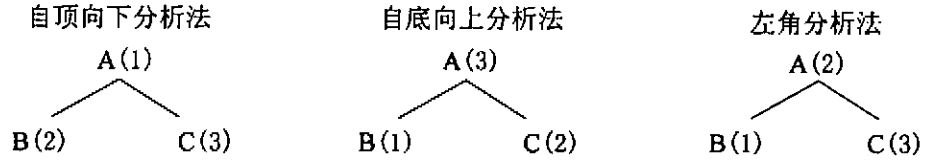
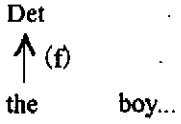


图7

左角分析法的分析从左角 B 开始，然后根据重写规则 A → BC，自下而上地推导出 A，最后再自顶向下地推导出 C。

下面，我们用左角分析法来分析句子“the boy hits the dog with a rod”。

(1) 首先从句首的 the 开始，根据文法的规则(f)，从规则(f)的左角 the，作出 Det。



(2) 因为规则(b)的左角为 Det，所以，从 Det 出发，选择文法(b)，并由此预测 Det 后面的 N(见图 8)。

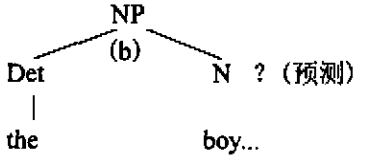


图 8

(3) 根据规则(h)，从 boy 作出 N (见图 9)。

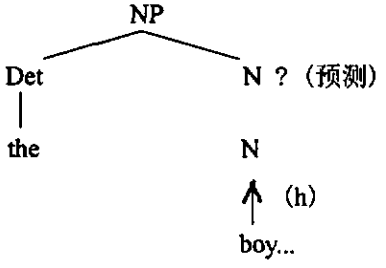


图 9

(4) 由于 boy 的父结点(father node)恰好是 N，可见我们对于 N 的预测是正确的，于是作出子树 NP (见图 10)。

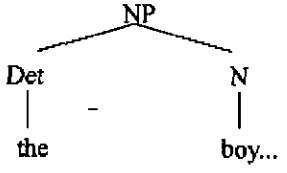


图 10

(5) NP 是规则(a)的左角，由 NP 选择规则(a)，并预测 VP(见图 11)。

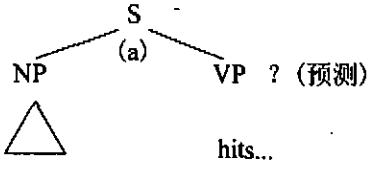


图 11

(6)根据规则(k)，由 hits 作出 V(见图 12)。

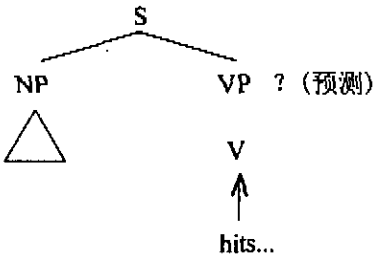


图 12

(7)由于 V 是规则(c)的左角，所以选择规则(c)，并预测 NP(见图 13)。

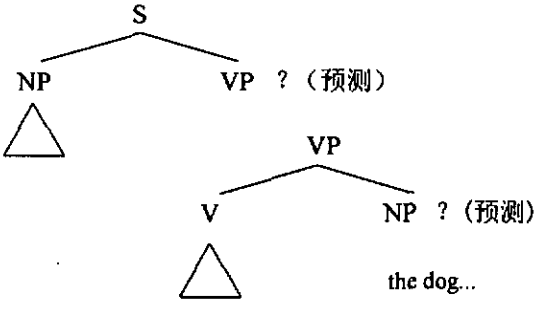


图 13

(8)从 the dog 作成 NP，对于 NP 的预测得到证实，由于 NP 得到证实，因此可继续证实对于 VP 的预测(见图 14)。

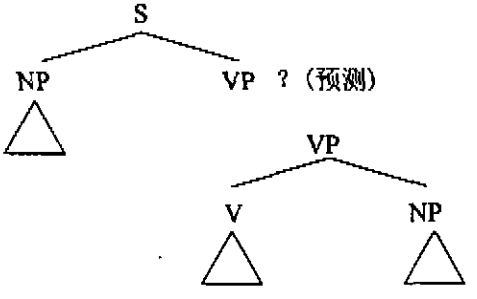


图 14

(9)由于 VP 还可以是规则(d)的左角，而且，the dog 之后还有 with 等单词，说明还不能过早地归约，需要进行回溯，以 VP 为规则(d)的左角，选择规则(d)来预测 PP(见图 15)。

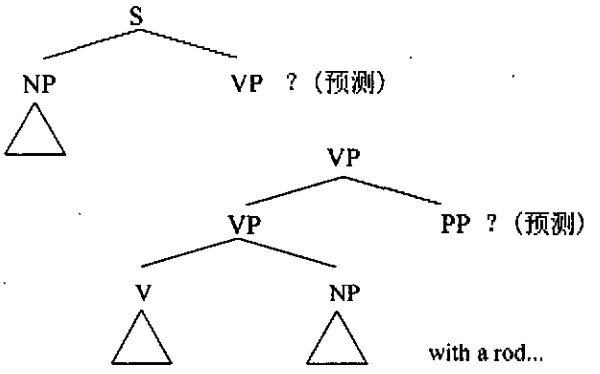


图 15

(10) VP 的预测得到证实，于是，完成句子 S（见图 16）。

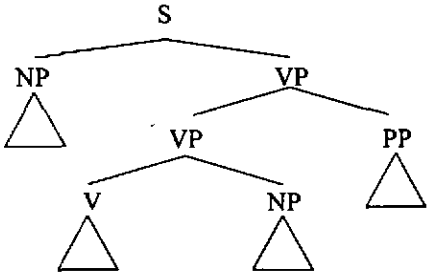


图 16

上述分析法都使用了回溯。当输入的符号串属于这种文法所描述的语言时，加入回溯机制能够保证输入符号串被接受。但是，当输入的符号串不属于这种文法所描述的语言时，通过多次回溯而没有新的选择可以回溯，输入符号串就将被拒绝。系统地回溯能够保证算法的正确性，但回溯同时也夹着大量的重复和多余的计算。

美国计算语言学家马尔库斯(M. Marcus)于1980年提出用人工的方法对归约的条件加以控制，从而避免了回溯。这就是“马尔库斯确定性分析算法”。马尔库斯的确定性算法由两部分组成：模式部分和行为部分。模式部分说明栈及缓冲区的内容在什么样的情况下，分析算法可以执行行为部分所表明的操作。马尔库斯引入的缓冲区是输入概念的推广，它从左到右按顺序存放一些已经建成的句子成分，允许查看的缓冲区的内容是有限的，这就避免了规则的复杂化。在行为部分允许的操作，有的类似于归约、移进，有的将栈顶元素移到缓冲区，有的将缓冲区的成分移出，挂到栈顶所放成分的结点之下，等等。

美国计算学者伊尔利(J. Earley)于 70 年代提出了伊尔利算法(Earley algorithm)。这种算法在左角分析法的基础上，把自顶向下分析法和自底向上分析法结合起来，在分析过程中交替地使用这两种分析法。首先自顶向下预测某个语言成分的起点，找出起点之后，再自底向上长成一棵子树。伊尔利算法提出了“点规则”，这种“点规则”采用在规则中加点的方式来系统地表示已经建成的结构部分和有待进一步分析的结构部分，从而步步为营地从左到右对句子进行分析，提高了分析的效率。

5．CYK 算法

CYK 算法是 Cocke-Younger-Kasami 算法的缩写，这是一种并行的句法分析算法。CYK 算法是以乔姆斯基范式(Chomsky normal form)为描述对象的句法分析算法。乔姆斯基范式的重写规则形式为

$$A \rightarrow BC$$

其中，A、B、C 都是非终极符号。乔姆斯基范式把单个的非终极符号重写为两个非终极符号 B 和 C，反映了自然语言的二分特性，在语言信息处理中便于用二叉树来表示自然语言的数据结构，更加适合于描述自然语言。显而易见，乔姆斯基范式的重写规则是上下文无关的短语结构语法的重写规则 $A \rightarrow \omega$ 中，当 ω =BC 时的一种特殊情况。由于任何的乔姆斯基范式与上下文无关的短语结构语法都是等价的，因此，这样的限制并不失其一般性。

对于英语句子“the boy hits a dog”(那个男孩儿打狗)，使用 CYK 分析法，我们可以得到表 1：

5	S				
4					
3			VP		
2	NP			NP	
1	Det	N	V	Det	N
	1	2	3	4	5
	the	boy	hits	a	dog

表 1

在这个表中，行方向(横向)的数字表示单词在句子中的位置，列方向(纵向)的数字表示该语言成分所包含的单词数。语言成分都装在框子(box)内，我们用 b_{ij} 来表示处于第 i 列第 j 行的框子的位置。这样，每一个语言成分的位置就可以确定下来。例如，

$Det \in b_{11}$ 表示 Det 处于第 1 列第 1 行；

$N \in b_{21}$ 表示 N 处于第 2 列第 1 行；

$V \in b_{31}$ 表示 V 处于第 3 列第 1 行；

$Det \in b_{41}$ 表示 Det 处于第 4 列第 1 行；

$N \in b_{51}$ 表示 N 处于第 5 列第 1 行。

这样一来，处于第 1 列第 2 行的 NP 的位置可用 b_{12} 表示($NP \in b_{12}$)，这种记法说明，这个 NP 处于句首，包含 2 个单词(the 和 boy)，也就是说，这个 NP 是由 Det 和 N 组成的；处于第 4 列第 2 行的 NP 的位置可用 b_{42} 表示($NP \in b_{42}$)，这种记法说明，这个 NP 处于第 4 个词的位置，包含 2 个单词（a 和 dog)，也就是说，这个 NP 是由 det 和 N 组成的；处于第 3 列第 3 行的 VP 的位置可用 b_{33} 表示($VP \in b_{33}$)，这种记法说明，这个 VP 处于第 3 个词的位置，包含 3 个单词(hits，a

和dog)，也就是说，这个VP是由V(包含1个词)和NP(包含2个词)组成的；处于第1列第5行的S的位置可用b_{15}表示($S \in b_{15}$)，这种记法说明，这个S处于句首，包含5个单词(the，boy，hits，a和dog)，也就是说，这个S是由NP(包含2个单词)和VP(包含3个单词)组成的。这些框子里的标记，明确地说明了这个句子中的句法结构关系，因此，如果我们能够通过有限步骤造出这样的表，就等于完成了句子的句法结构分析。

由于文法规则都用乔姆斯基范式表示，因此，在文法规则A→BC中，对于某个k($1 \leqslant k < j$)来说，如果b_{ik}中包含B，$b_{i+k\ j-k}$中包含C，则b_{ij}中必定包含A。也就是说，如果从输入句子中的第i个单词开始，造成了表示由k个单词组成的成分B的子树（这时，B的长度为k，其首词标号为第*i列*，末词标号第*i+k-1列*，例如，如果B的长度为4，如首词标号为3，则末词标号为i+k-1=3+4-1=6，即这4个词的标号分别为3，4，5，6），从第i+k个单词开始，造成了表示由j-k个单词组成的成分C的子树（这时，C的长度为j-k，其首词标号为第i+k列，末词标号为第i+j-1列。例如，如果A的长度j=6，C的长度为j-k=6-4=2，则其首词标号为i+k=3+4=7，末词标号为i+j-1=3+6-1=8)，那么，我们就可以作出表示A的树形图17：

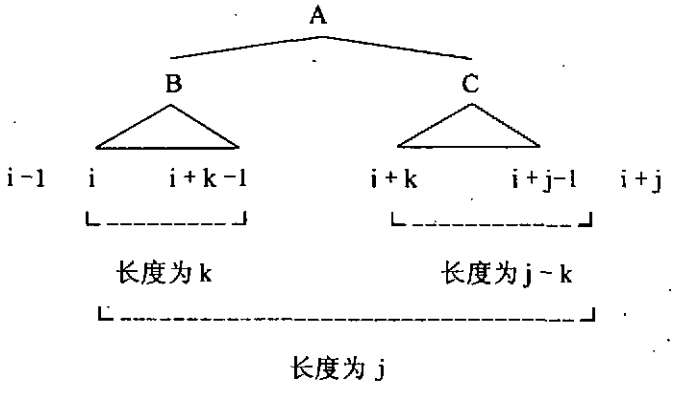


图17

例如，在上表的b_{12}中包含NP，b_{11}中包含Det，b_{21}中包含N，这反映了文法规则NP→Det N的情况。这时，k=1，i=1，j=2。

CYK算法就是顺次构造上述表的算法，当输入句子的长度为n时，CYK算法可分为如下两步。

第一步：从i=1开始，对于长度为n的输入句子中的每一个单词Wi，显然都有重写规则A→Wi，因此，顺次给每一个单词Wi相应的非终极符号A记入框子b_{i1}中。在我们的例句“the boy hits a dog”中，根据相应的重写规则，顺次把Det记入b_{11}中，把N记入b_{21}中，把V记入b_{31}中，把Det记入b_{41}中，把N记入b_{51}中。

第一步相当于确定输入句子中各个单词所属的词类，如果一个单词属于若干个词类，可以把它所属的词类都记入表中。

第二步：对于$1 \leqslant h < j$以及所有的i，造出b_{ih}，这时，包含B_{ij}的非终极符号的集合定义如下：

b_{ij} = {A| 对于$1 \leqslant k < j$，B包含在b_{ik}中，C包含在$b_{i+k\ j-k}$中，并且，存在文法规则A→BC}。

第二步相当于构造句子的句法结构。根据文法的重写规则，从句首开始，顺次由1到n取词

构造框子 b_{ij}，如果框子 b_{1n} 中包含开始符号S，也就是说，$S \in b_{1n}$，那么，就说明输入句子是可以接受的。

例如，根据规则 NP → Det N 以及 $det \in b_{11}$ 和 $N \in b_{21}$，可知此时 i=1, k=1, j=2, 因此，NP 的框子的编号应为 b_{12}；根据规则 NP → Det N 以及 $Det \in b_{41}$ 和 $N \in b_{51}$，可知此时 i=4, k=1, j=2, 因此，这个 NP 的框子的编号应为 b_{42}；根据规则 VP → V NP 以及 $V \in b_{31}$ 和 $NP \in b_{42}$，可知此时 i=3, k=1, j=3, 因此，VP 的框子的编号应为 b_{33}；根据规则 S → NP VP 以及 $NP \in b_{12}$ 和 $VP \in b_{33}$，可知此时 i=1, k=2, j=5, 因此，S 的框子的编号 b_{51}。由于句子长度 n=5，因此，有 $S \in b_{n1}$，所以输入句子被接受，分析成功。

CYK 算法由小型分析树开始逐渐扩大，同样的分析树绝不重复运算，不需要进行回溯，规则都采用乔姆斯基范式，这是它的优越之处。

短语结构语法具有结构清晰、简洁明确、易于操作等优点，给自然语言的计算机处理带来了许多方便。因此，上述基于短语结构语法的自动句法分析方法，在计算语言学中得到广泛的应用，目前仍然有着很强的生命力。

参考文献

Aho, A. V., and J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling, Vol. 1*. Prentice-Hall.

Chomsky, Noam. 1956. Three models for the description of language, I. R. E. *Transaction on Information Theory Vol. IT-2, Proceedings of the Symposium on Information Theory.*

----. 1957. *Syntactic Structure*. The Hague: Mouton & Co. 中译本：邢公畹等译，1979 年，《句法结构》北京：中国社会科学出版社。

----. 1959. On certain formal properties of grammars. *Information & Control, II.* Pp.137-167.

Earley, J. 1970. An efficient context-free parsing algorithm. *C.ACM* 13, 2.

Marcus, W. P. 1980. *A Theory of Syntactic Recognition for Natural Language*. The MIT Press.

Tomita, M. 1985. An efficient context-free parsing algorithm for natural languages. *Proceedings of the 9th International Joint Conference on Artificial Intelligence.*

Winograd, T. 1983. *Language as a Cognitive Process, Vol. 1*. Addison Wesley.

Woods, W. A. 1970. Transition network grammar for natural language analysis. *C.ACM* 13, 10.

冯志伟，1982，从形式语言理论到生成转换语法。《语言研究论丛》第二辑，天津：天津人民出版社。

———，1983，生成语法的公理化方法。《哈尔滨生成语法讨论会论文集》22-31 页。

———，1996，《自然语言的计算机处理》上海：上海外语教育出版社。

作者通讯地址：北京朝内南小街 51 号　语言文字应用研究所　邮编 100010