

载《东方语言学》，第8辑，上海教育出版社，1-17页，2011年。

从自然语言处理的角度看二分法

冯志伟

(教育部语言文字应用研究所，北京，100010)

摘要：本文从自然语言处理的角度分析了二分法的优点和不足，指出二分法是多分法的一种特殊情况，主张采用多分法。最后讨论了自然语言的计算复杂性。

关键词：二分法；多分法；上下文无关文法

On Binary Branching Approach based on Natural Language Processing

Feng Zhiwei

(Institute of Applied Linguistics, the Ministry of Education, Beijing, 100010)

Abstract: The author analyzes the advantages and disadvantages of binary branching approach from the view of natural language processing, he points out that the binary branching is a special case of multiple branching. He believed that multiple branching is a general approach. The computational complexity of natural language is also discussed in the last.

Key Words: binary branching approach; multiple branching approach; context-free grammar

吴道平在《“两分法”浅析》¹中介绍了语言学中二分法的起源及其在生成转换语法发展中的作用和地位，并分析了 Richard S. Kayne 提出严格二分 (Binary Branching) 的动因和内在逻辑²。这篇文章对于我们进一步了解二分法很有帮助。

我是搞自然语言处理的，近年来对于理论语言学的问题思考得不多，吴道平的文章很深邃，很多地方我还没有完全领会。这里，我只想根据我在自然语言处理研究中的感受，谈一谈自己对于二分法的一些不成熟的想法，就教于方家。

在 N. Chomsky 早期的形式语言理论的研究中，他把文法³定义为四元组

$$G = (V_N, V_T, S, P)$$

其中， V_N 是非终极符号的集合，这些符号不能处于生成的终点； V_T 是终极符号的集合，这些符号能处于生成的终点；显然， V_N 与 V_T 的并构成了 V ， V_N 与 V_T 不相交，因而有

$$V = V_N \cup V_T$$

¹ 吴道平，“两分法”浅析，《东方语言学》，第3期。

² Kayne, Richard S. (1984) *Connectedness and Binary Branching*, Foris.

³ 本文中把 grammar 一律翻译成“文法”，这是遵照计算机科学中的习惯用法，“文法”中的“文”指“规则”之意（如“吏乱法曰舞文”中的“文”就是“规则”），而不是指“文章”之意。程序语言也有 grammar，但程序语言是不能“讲”的，因此，计算机科学中一般不把 grammar 翻译为“语法”。

$$V_N \cap V_T = \phi (\phi \text{表示空集})$$

V_N 中的符号用大写拉丁字母表示, V_T 中的符号用小写拉丁字母表示。符号串用希腊字母表示(有时也可以用拉丁字母表中排在较后面的如 ω 之类的小写拉丁字母来表示符号串)。

S 是 V_N 中的初始符号。

P 是重写规则, 其一般形式为:

$$\varphi \rightarrow \psi。$$

这里, φ 是 V^+ 中的符号串, ψ 是 V 中的符号串, 也就是说, $\varphi \neq \phi$ ⁴。

在文法 $G = \{V_N, V_T, S, P\}$ 中, 其重写规则为 $\varphi \rightarrow \psi$, 并且要求 $\varphi \neq \phi$ 。为了给文法进行分类, Chomsky 对文法重写规则加上如下的限制:

限制 1: 如果 $\varphi \rightarrow \psi$, 那么, 存在 $A, \omega_1, \omega_2, \omega$, 使得 $\varphi = \varphi_1 A \varphi_2, \psi = \varphi_1 \omega \varphi_2$ 。

限制 2: 如果 $\varphi \rightarrow \psi$, 那么, 存在 $A, \omega_1, \omega_2, \omega$, 使得 $\varphi = \varphi_1 A \varphi_2, \psi = \varphi_1 \omega \varphi_2$, 并且 $A \rightarrow \omega$ 。

限制 3: 如果 $\varphi \rightarrow \psi$, 那么, 存在 $A, \omega_1, \omega_2, \omega, a, Q$, 使得 $\varphi = \varphi_1 A \varphi_2, \psi = \varphi_1 \omega \varphi_2, A \rightarrow \omega$, 并且 $\omega = aQ, \omega \rightarrow a$, 因而 $A \rightarrow aQ, A \rightarrow a$ 。

限制 1 要求文法重写规则全都具有形式 $\varphi_1 A \varphi_2 \rightarrow \varphi_1 \omega \varphi_2$, 这样的重写规则在上下文 $\varphi_1 - \varphi_2$ 中给出 $A \rightarrow \omega$ 。显然, 在这种情况下, ψ 这个符号串的长度(即 ψ 中的符号数)至少等于或者大于 φ 这个符号串的长度(即 φ 中的符号数), 如果用 $|\psi|$ 和 $|\varphi|$ 分别表示符号串 ψ 和 φ 的长度, 则有 $|\psi| \geq |\varphi|$ 。由于在重写规则 $\varphi_1 A \varphi_2 \rightarrow \varphi_1 \omega \varphi_2$ 中, 每当 A 出现于上下文 $\varphi_1 - \varphi_2$ 中的时候, 可以用 ω 来替换 A , 因此, 把加上了限制 1 的文法叫做上下文有关文法⁵(context-sensitive grammar)或 1 型文法(type 1 grammar)。

限制 2 要求文法重写规则全都具有形式 $A \rightarrow \omega$, 这时上下文 $\varphi_1 - \varphi_2$ 是空的, 在运

⁴ V^* 表示由 V 中的符号构成的全部符号串(包括空符号串 ϕ)的集合, V^+ 表示 V^* 中除 ϕ 之外的一切符号串的集合。例如, 如果 $V = \{a, b\}$, 则有

$$V^* = \{\phi, a, b, aa, ab, ba, bb, aaa, \dots\},$$

$$V^+ = \{a, b, aa, ab, ba, bb, aaa, \dots\}。$$

⁵ 有的学者把“上下文有关文法”称为“上下文敏感文法”。

用重写规则时不依赖于单个的非终极符号 A 所出现的上下文，因此，把加上了限制 2 的文法叫做上下文无关文法⁶(context-free grammar)或 2 型文法(type 2 grammar)。

限制 3 要求文法重写规则全都具有形式 $A \rightarrow aQ$ 或 $A \rightarrow a$ ，其中， A 和 Q 是非终极符号， a 是终极符号，这种文法叫做有限状态文法(finite state grammar)或 3 型文法(type 3 grammar)，有时也可叫做正则文法(regular grammar)。

没有上述限制的文法，叫做 0 型文法(type 0 grammar)。

显而易见，每一个有限状态文法都是上下文无关的，每一个上下文无关文法都是上下文有关的，每一个上下文有关文法都是 0 型的。这样，Chomsky 把由 0 型文法生成的语言叫 0 型语言(type 0 language)，把由上下文有关文法、上下文无关文法、有限状态文法生成的语言分别叫做上下文有关语言(context-sensitive language)、上下文无关语言(context-free language)、有限状态语言(finite state language)，也可以分别叫做 1 型语言(type 1 language)、2 型语言(type 2 language)、3 型语言(type 3 language)。由于从限制 1 到限制 3 的限制条件是逐渐增加的，因此，不论对于文法或对于语言来说，都有

$$0 \text{ 型} \supseteq 1 \text{ 型} \supseteq 2 \text{ 型} \supseteq 3 \text{ 型}。$$

这种包含关系叫做“Chomsky 层级”(Chomsky hierarchy)。

例如，有文法 $G = \{V_N, V_T, S, P\}$

$$V_N = \{S, A, B, C\}$$

$$V_T = \{a, b, c\}$$

$$S = \{S\}$$

P :

$$\begin{array}{ll} S \rightarrow ABC & \text{(i)} \\ A \rightarrow aA & \text{(ii)} \\ A \rightarrow a & \text{(iii)} \\ B \rightarrow Bb & \text{(iv)} \\ B \rightarrow b & \text{(v)} \\ BC \rightarrow Bcc & \text{(vi)} \\ ab \rightarrow ba & \text{(vii)} \end{array}$$

这个文法可生成终极符号串 $b^n a^m cc$ ($n \geq 1, m \geq 1$)。

不难看出，规则 (vii) 是 0 型规则，因此，这个文法是 0 型文法。如果去掉规则 (vii)，那么就得到一个 1 型文法，因为规则 (vi) $BC \rightarrow Bcc$ 是 1 型规则。如果去掉规则 (vii) 和 (vi)，就得到一个 2 型文法，因为规则 (i) $S \rightarrow ABC$ 及规则 (iv) $B \rightarrow Bb$ 是 2 型规则。如果去掉规则 (vii)、(vi)、(i)、(iv)，就得到一个 3 型文法，因为剩下的规则 (ii) $A \rightarrow aA$ ，规则 (iii) $A \rightarrow a$ ，规则 (v) $B \rightarrow b$ 都是 3 型规则。

⁶ 有的学者把“上下文无关文法”称为“上下文自由文法”。

可见，任何的 3 型文法，一定包含在 2 型、1 型、0 型文法中，任何的 2 型文法，一定包含在 1 型、0 型文法中，任何的 1 型文法，一定包含在 0 型文法中。

Chomsky 还分析了上述这些文法的长处和不足。

他指出，有一些由非常简单的符号串构成的形式语言，不能由有限状态文法生成，它们是：

(i) $ab, aabb, aaabbb, \dots$ ，它的全部句子都是由若干个 a 后面跟着同样数目的 b 组成的。这种形式的语言可表示为 $L_1 = \{a^n b^n\}$ ，其中， $n \geq 1$ 。

(ii) $aa, bb, abba, baab, aabbaa, abbbba, \dots$ ，这种形式语言是镜象结构语言，如果用 α 表示集合 $\{a, b\}$ 上的任意非空符号串，用 α^* 表示 α 的镜象，那么，这种语言可表示为 $L_2 = \{\alpha\alpha^*\}$ 。

(iii) $aa, bb, abab, aaaa, bbbb, aabaab, abbabb, \dots$ ，它的全部句子是由若干个 a 或者若干个 b 构成的符号串 α 后面跟着而且仅只跟着完全相同的符号串 α 而组成的。如果用 α 表示集合 $\{a, b\}$ 上的任意非空符号串，那么，这种语言可表示为 $L_3 = \{\alpha\alpha\}$ 。

L_1, L_2, L_3 都不能由有限状态文法生成，可见，有限状态文法的生成能力不强。

Chomsky 认为，上下文无关文法比较适合于描述自然语言。为此，他提出了“Chomsky 范式” (Chomsky Normal Form)。

Chomsky 证明了，任何的上下文无关语言，均可由重写规则为

$$A \rightarrow BC$$

或 $A \rightarrow a$

的文法生成。其中， $A, B, C \in V_N, a \in V_T$ 。

从 Chomsky 范式的重写规则的形式 $A \rightarrow BC$ 不难看出，Chomsky 范式是严格实行“二分法”的； $A \rightarrow a$ 不过是当规则左部的两个非终极符号 BC 蜕化成一个单独的终极符号 a 时的一种特殊情况，从实质上说也是二分的。

利用 Chomsky 范式，可把任何的上下文无关文法的推导树简化为二元形式。

例如，上下文无关语言 $\{a^n cb^{2n}\}$ 的文法重写规则为

$$S \rightarrow aCbb$$

$$C \rightarrow aCbb$$

$$C \rightarrow c$$

如果要生成符号串 $aacbbbbb$ ，其推导树为：

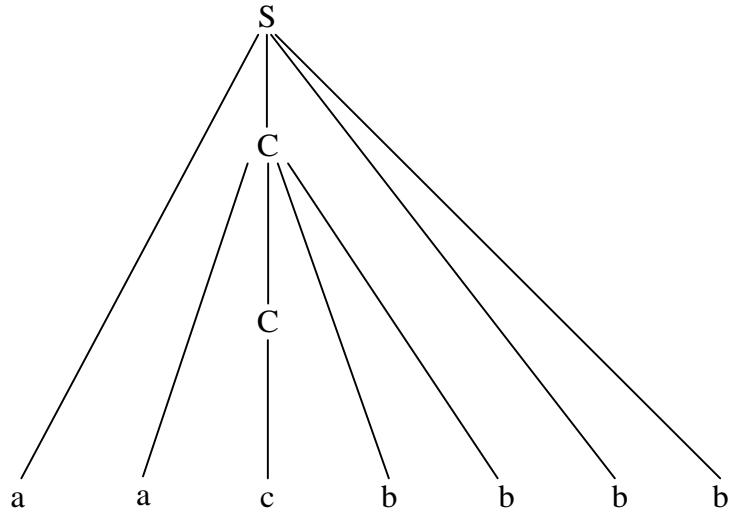


图 1 生成符号串 $aacbbbb$ 的推导树

现在我们把这个文法的三个重写规则改写为 Chomsky 范式。

在这三个重写规则中， $C \rightarrow c$ 是符合 Chomsky 范式要求的，不必再变换。我们先把 $S \rightarrow aCbb$ 及 $C \rightarrow aCbb$ 的右边换为非终极符号，用 $S \rightarrow ACBB$ 及 $A \rightarrow a$ ， $B \rightarrow b$ 替换 $S \rightarrow aCbb$ ，用 $C \rightarrow ACBB$ 及 $A \rightarrow a$ ， $B \rightarrow b$ 替换 $C \rightarrow aCbb$ 。然后，再把 $S \rightarrow ACBB$ ， $C \rightarrow ACBB$ 的右边换成二元形式，用 $S \rightarrow DE$ ， $D \rightarrow AC$ 及 $E \rightarrow BB$ 替换 $S \rightarrow ACBB$ ，用 $C \rightarrow DE$ ， $D \rightarrow AC$ 及 $E \rightarrow BB$ 替换 $C \rightarrow ACBB$ 。这样，便得到了符合 Chomsky 范式要求的文法重写规则：

- $S \rightarrow DE$
- $D \rightarrow AC$
- $E \rightarrow BB$
- $C \rightarrow DE$
- $A \rightarrow a$
- $B \rightarrow b$
- $C \rightarrow c$

用 Chomsky 范式，可将符号串 $aacbbbb$ 的推导树简化为二元形式，这种二元形式的推导树叫“二叉树” (binary tree)，如下图所示：

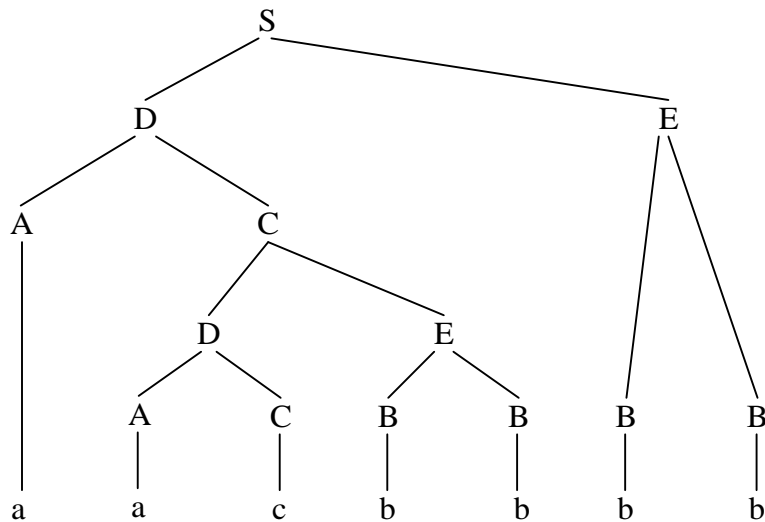


图 错误!文档中没有指定样式的文字。 二叉树

在 Chomsky 范式中，重写规则及推导树都有二元形式，这就为自然语言的形式描写提供了数学模型。

自然语言中的句法结构一般都是二分的，因而一般都具有二元形式 (binary form)。例如，在汉语中，除了联合结构、双宾语结构和递系结构(即“兼语式”)之外，具有二元形式的句法结构占大多数：

- 述宾结构： 思考问题
 └─┬─┘
- 主谓结构： 他走了
 └─┬─┘
- 偏正结构： 汉语语法
 └─┬─┘
- 动补结构： 洗干净
 └─┬─┘

事实上，语言学中正是采用二分法来分析句子的。二分法就是所谓的层次分析法，这就是说，一个复杂的语言形式，不能一下子就把它分析为若干个词，而要按下面的步骤分析：

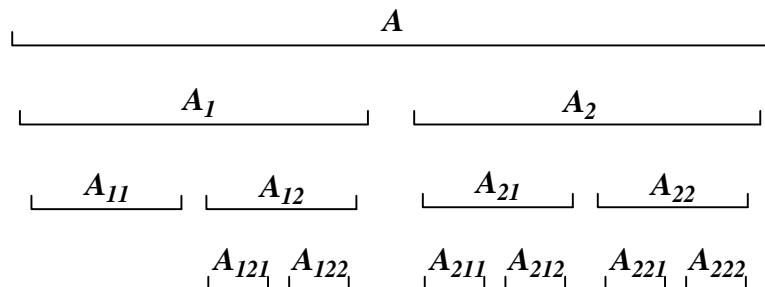


图 3 层次分析法示意图

我们不是把 A 一下子就分成 A_{11} , A_{121} , A_{122} , A_{211} , A_{212} , A_{221} , A_{222} ，而是先把 A

分成 A_1 和 A_2 两部分，然后把 A_1 分成 A_{11} 和 A_{12} 两部分，把 A_2 分成 A_{21} 和 A_{22} 两部分， A_{12} 又可再分为 A_{121} ， A_{122} 两部分，……，这样分析下去，一直分析到词为止。人们通常把 A_1 和 A_2 叫做 A 的直接成分，把 A_{11} 和 A_{12} 叫做 A_1 的直接成分，把 A_{21} ， A_{22} 叫做 A_2 的直接成分。

这种顺次找出语言格式的直接成分的方法，叫做直接成分分析法或层次分析法。因此，在语言学界，又有人把 Chomsky 的上下文无关短语结构语言模型叫做直接组成成分模型，把用直接组成成分模型分析语言的方法叫做直接成分分析法 (immediate constituent analysis, 简称 IC 分析法)。

可见，Chomsky 范式反映了自然语言结构的这种二分特性，因而通过 Chomsky 范式这一重要工具，上下文无关文法可以在自然语言研究中得到广泛的应用。

不少语言学家在他们描写自然语言的工作中，已经认识到了自然语言结构的这种二分特性。

吴道平在他的文章中举出了 Wilhelm Wundt 等人使用二分法描述自然语言的例子。我们这里再补充一下面的材料。

-- 我国语言学家马建忠在《马氏文通》中提出了两端两语说，指出：“盖意非两端不明，而句非两语不成”。

-- 美国语言学家 E. A. Nida(奈达)在 1949 年出版的《形态学》中，指出：“根据经验，我们发现语言结构倾向于二分”⁷。

-- 美国语言学家 C. C. Fries(福里斯)在《英语结构》一书中，更是明确地提出了二分的观点，他指出：“在英语里，一个结构层次通常只有两个成分。当然，每一个成分都可以由好几个单位组成，不过在同一层次上，结构的直接成分通常只有两个”⁸。

Fries 还把这一观点具体地应用于英语句子的分析中。例如，他对“The recommending committee approved his promotion”(“推荐委员会批准了他的提升”)这个句子的分析是：

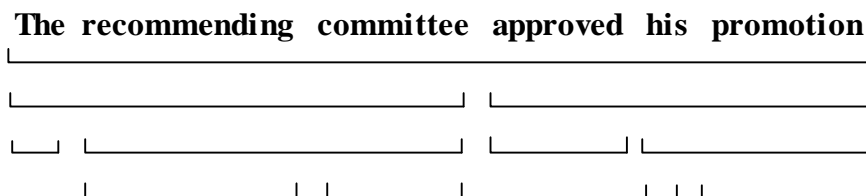


图 4 英语句子的层次分析图示

完全采用了二分法。

在自然语言处理中，二分法也得到了广泛的应用。CYK 算法 (CYK algorithm) 就是严格遵循二分法的。

CYK 算法是 Cocke-Younger-Kasami 算法的首字母缩写。这是一种并行的句法分析算法。这种算法是以 Chomsky 范式 (Chomsky normal form) 为描述对象的句法分析算法。

例如，如果我们有如下的上下文无关文法的规则：

$$S \rightarrow NP VP$$

⁷ E. A. Nida, 《Morphology》, University of Michigan Press, 1949, P. 91~93.

⁸ C. C. Fries, 《英语结构》(中译本), 264 页, 商务印书馆, 英文本原名《English Structures》.

VP → V NP

NP → Det N

这些规则都是二分的，满足 Chomsky 范式的要求。

根据这种满足 Chomsky 范式要求的上下文无关文法，对于英语句子“the boy hits a dog”（那个男孩儿打狗），使用 CYK 分析法，我们可以得到图 5 中的表：

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

图 5 CYK 算法中的表

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，

$Det \in b_{1,1}$ 表示 Det 处于第 1 列第 1 行，

$N \in b_{2,1}$ 表示 N 处于第 2 列第 1 行，

$V \in b_{3,1}$ 表示 V 处于第 3 列第 1 行，

$Det \in b_{4,1}$ 表示 Det 处于第 4 列第 1 行，

$N \in b_{5,1}$ 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 $b_{1,2}$ 表示 ($NP \in b_{1,2}$)，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 $b_{4,2}$ 表示 ($NP \in b_{4,2}$)，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 $b_{3,3}$ 表示 ($VP \in b_{3,3}$)，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 $b_{1,5}$ 表示 ($S \in b_{1,5}$)，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。

由于语法规则都用 Chomsky 范式表示，因此，在语法规则 $A \rightarrow BC$ 中，对于某个 k ($1 \leq k < j$) 来说，如果 $b_{i,k}$ 中包含 B， $b_{i+k,j-k}$ 中包含 C，则 $b_{i,j}$ 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，就可以作出如下的表示 A 的树形图（图 6）：

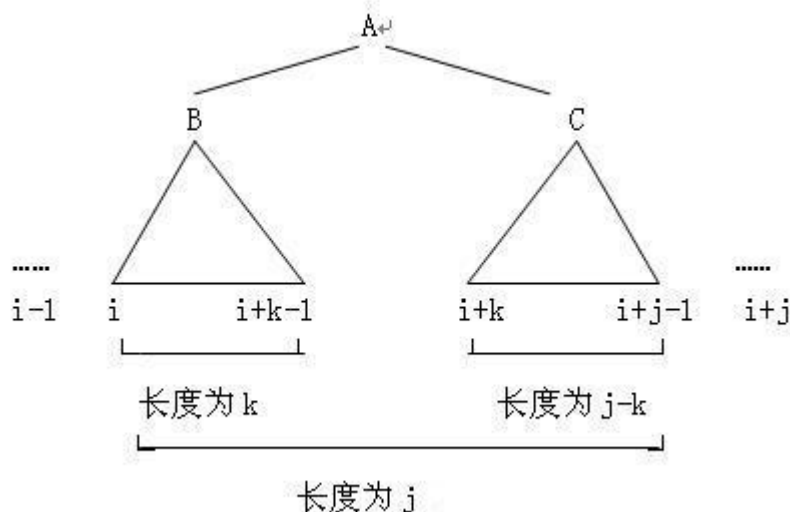


图 6 CYK 算法中的标号

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 $NP \rightarrow Det N$ 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 W_i ，显然都有重写规则 $A \rightarrow W_i$ ，因此，顺次给每一个单词 W_i 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 $1 \leq h < j$ 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：

$b_{ij} = \{A \mid \text{对于 } 1 \leq k < j, B \text{ 包含在 } b_{ik} \text{ 中, } C \text{ 包含在 } b_{i+k-jk} \text{ 中, 并且, 存在文法规则 } A \rightarrow BC\}$ 。

第二步相当于构造句子的句法结构。根据文法重写规则，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 $NP \rightarrow Det N$ 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 $NP \rightarrow Det N$ 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 $VP \rightarrow V NP$ 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 $S \rightarrow NP VP$ 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。

由于基于Chomsky范式的CYK算法严格遵循了二分法，算法简单明快，效率较高。在自然语言处理中强制性地使用二分法，采用二叉树来描述自然语言的层次结构和线性顺序，可以提高自然语言处理系统的性能。基于Chomsky范式的二分法，在自然语言处理中受到了欢迎。

然而，自然语言处理的实践表明，这种二分法还是有缺陷的。我们认为，主要的缺陷有如下三方面。

第一，上下文无关文法的规则是 $A \rightarrow \omega$ ，其中规则右部的 ω 可以是终极符号，也可以是非终极符号，还可以是由终极符号和非终极符号混合组成的符号串，所以， ω 不一定总是二分的。如果上下文无关文法的规则不是严格二分的，那么，在 CYK 算法中，就必须首先把这些规则都转写成严格二分的 Chomsky 范式，才有可能用 CYK 算法进行自动分析。这样，在很多情况下，我们需要对于规则进行转换，CYK 算法使用起来就不是很方便，自动分析的效率也会降低。

例如，如果上下文无关语法具有如下的规则：

$S \rightarrow NP VP$

$NP \rightarrow PrN$

$NP \rightarrow DET N$

$NP \rightarrow N WH VP$

$NP \rightarrow DET N WH VP$

$VP \rightarrow V$

$VP \rightarrow V NP$

$VP \rightarrow V \text{ that } S$

我们用这个语法来剖析句子“the table that lacks a leg hits Jack”。

由于这些规则不全是二分的，首先我们需要把重写规则转换为 Chomsky 范式。这样，就需要做如下的转换工作，逐一地审查每一条规则是不是二分的：

$S \rightarrow NP VP$ 这个规则是二分的，是 Chomsky 范式，不需要转换。

$NP \rightarrow PrN$ 这个规则不是 Chomsky 范式，因此转换为如下的 Chomsky 范式：

$NP \rightarrow Jack$

$NP \rightarrow John$

$NP \rightarrow Maria$

$NP \rightarrow DET N$ 这个规则是二分的，是 Chomsky 范式，不需要转换。

$NP \rightarrow N WH VP$ 这个规则不是 Chomsky 范式，因此转换为如下的 Chomsky 范式：

$NP \rightarrow N CL$

$CL \rightarrow WH VP$

$NP \rightarrow DET N WH VP$ 这个规则不是 Chomsky 范式，因此转换为如下的 Chomsky 范式：

$NP \rightarrow NP CL$

$NP \rightarrow DET N$

$CL \rightarrow WH VP$

这里 CL 是一个 WH 从句 (WH clause)，它由 that 和 VP 组成。

$VP \rightarrow V$ 这个规则不是 Chomsky 范式，因此转换为如下的 Chomsky 范式：

$VP \rightarrow cough$

$VP \rightarrow walk$

$VP \rightarrow \dots$

$VP \rightarrow V NP$ 这个规则是二分的，是 Chomsky 范式，不需要转换。

$VP \rightarrow V \text{ that } S$ 这个规则不是 Chomsky 范式，因此转换为如下的 Chomsky 范式：

VP → V TH
 TH → WH S

这里 TH 是一个 that 从句，它有 that 和 S 组成。

我们必须把全部规则转换成 Chomsky 范式之后，才能够根据 CYK 算法来计算非终极符号 b_{ij} 的列号和行号。例如，我们可以按照句子中的词序排列表示词类 (POS) 的非终极符号 b_{ij} ，逐一地计算它们的列号和行号：

“The table that lacks a leg hits Jack”
 DET N WH V DET N V NP
 b_{11} b_{21} b_{31} b_{41} b_{51} b_{61} b_{71} b_{81}

计算出表示短语的非终极符号 b_{ij} 的列号和行号之后，为我们得到如下的方框和表：

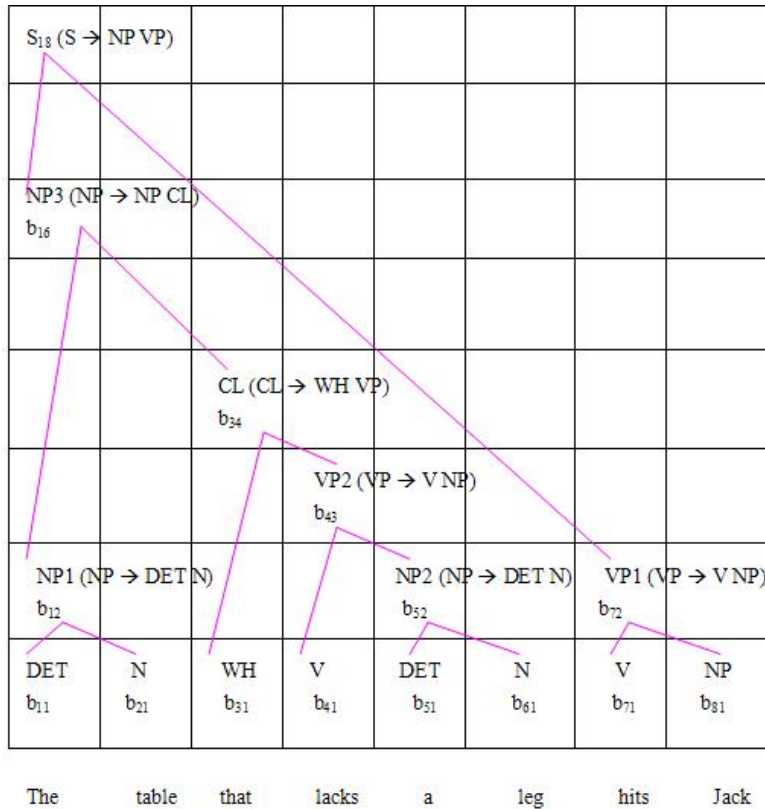


图 7 句子的方框和表

其中，各个方框中的 b_{ij} 计算详情如下：

- b_{ij} (NP1): $i=1, j=1+1=2$
- b_{ij} (NP2): $i=5, j=1+1=2$
- b_{ij} (VP1): $i=7, j=1+1=2$
- b_{ij} (VP2): $i=4, j=1+2=3$
- b_{ij} (CL): $i=3, j=1+3=4$
- b_{ij} (NP3): $i=1, j=2+4=6$
- b_{ij} (S): $i=1, j=2+6=8$

这个句子的长度为 8，我们得到的 S 的方框中的行号也为 8，因此句子剖析成功。

我们使用 CYK 算法构造出图 7 中的表中的各个结点可以系连起来形成一个金字塔 (pyramid)，这个金字塔也就是一个树形图，它可以表示句子的结构。为了得到这样的金字塔，我们首先必须把全部的规则转换为 Chomsky 范式，影响了自动分析的效率。

为了严格遵守二分的原则，在规则的转换的时候，有时要进行分解，有时要进行合并。

在上面的例子中，我们是将规则分解为 Chomsky 范式；有时我们还需要把一些规则合并成二分的 Chomsky 范式。

如果我们使用 CYK 算法来剖析句子“book that flight”，就必须对规则进行合并。

用于分析这个句子的上下文无关文法规则是：

- S → VP
- VP → Verb NP
- NP → Det Nominal
- Nominal → Noun

由于第一条规则 S → VP 的右边边只包含一个单独的非终极符号 VP，这不是 Chomsky 范式，但第二条规则 VP → Verb NP 是 Chomsky 范式，因此，我们把第一条规则和第二条规则合并，形成如下的符合 Chomsky 范式要求的规则：

$$S \rightarrow \text{Verb NP}$$

第四条规则 Nominal → Noun 的右边边也只包含一个单独的非终极符号，也不是 Chomsky 范式，但第三条规则 NP → Det Nominal 是 Chomsky 范式，因此，我们把第四条规则和第三条规则合并，形成如下的符合 Chomsky 范式要求的规则：

$$NP \rightarrow \text{Det Noun}$$

经过规则合并之后的上下文无关语法的规则如下：

- S → Verb NP
- NP → Det Noun

这些规则都符合 Chomsky 范式的要求了。只有在规则合并之后，我们才可以根据这样符合 Chomsky 范式要求的规则，使用 CYK 算法分析上述句子。结果如下：

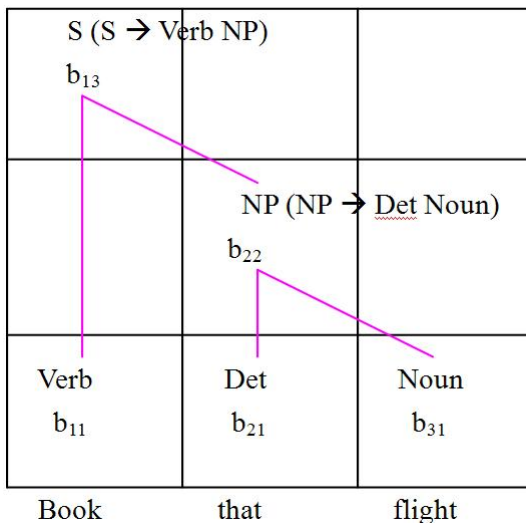


图 8 句子的方框和表

其中，各个方框中的 b_{ij} 计算详情如下：

$$b_{ij} \text{ (NP): } i=2, j=1+1=2$$

$$b_{ij} \text{ (S): } i=1, j=1+2=3$$

用 CYK 算法造出的金字塔也就是表示句子结构的树形图。

由此可见，CYK 算法虽然是一种简单而有效的算法，但是，这种算法的前提是：所有的规则必须是 Chomsky 范式，否则，算法无法运行。这样，如果我们要使用 CYK 算法，首先必须进行规则的转换工作，对于不符合 Chomsky 范式要求的规则进行分解或合并，这样，就必然会影响到自动分析的效率。这是二分法的第一个严重缺陷。

第二，自然语言中的许多文法形式不便于用基于二分法的二叉树来描述，而应该采用多叉树（multiple-branching tree）来描述。这是二分法的第二个严重缺陷。

下面我们以汉语为例来说明。

二叉树难以描述汉语中的如下结构：

① 兼语式：

在“我们|请|他|做报告”中，“他”是“请”的宾语，又是“做报告”的主语，如用二叉树表示，就会前后交叠，而用多叉树就描述得很清楚。

② 状-述-宾式：

在“努力学习|英语”中，如用二叉树来描述，是先二分为“努力学习|英语”呢，还是先二分为“努力学习|英语”呢？常常令人踌躇不决，举棋不定，而用多叉树将其切分为三部分：“努力学习|英语”，就避免了用二叉树描述的困难。

③ 双宾语：

在“给|弟弟|一本书”中，由于动词“给”有两个宾语，难于用二叉树描述，而应该用多叉树描述。

④ 多项并列结构：

在“英语、法语、俄语、西班牙语和汉语都是联合国的工作语言”中，多项并列结构“英语、法语、俄语、西班牙语和汉语”如果采用二叉树来描述就显得很勉强，而且，层次太多，使人眼花缭乱，而如果采用多叉树来描述就简洁得多。

因此，汉语的特点说明了二分法在汉语描述中的局限性。对于上述汉语中的结构，如果勉强地把它们转换为二分形式，往往会削足适履，使我们陷入左右为难的困境。

其实，Chomsky 本人也注意到了二叉树的这个局限。

吴道平的文章告诉我们，早在 1957 年，Chomsky 在《句法结构》⁹中，谈到一种上下文有关的现象，他指出，英语中动词 hit 现在时第三人称加-s 时，这个动词前面的 NP 必须是单数的，因此，hit 现在时第三人称加-s 与上文的 NP 的数有关，Chomsky 主张用

$$NP_{\text{sing}} + \text{Verb} \rightarrow NP_{\text{sing}} + \text{hits}$$

来表达这种上下文有关现象，这里的 NP_{sing} 就是 Verb 的上文。对于这种上下文有关的现象，Chomsky 已经感觉到难以用二分法来加以描述了，因此，他只好给 NP 加下标来说明这种上下文有关现象，把 NP 改为 NP_{sing} 。

英语中的非连续成分也是难以用二分法来描述。

例如，短语 *pick it up*（“把它拣起来”），应当采用三分的结构来表达如下：

$$VP \rightarrow V \text{ NP Part}$$

如果使用 Chomsky 范式，把上述三分的规则改用两条二分的规则来代替它：

$$VP \rightarrow VP \text{ Part}$$

$$VP \rightarrow V \text{ NP}$$

这样虽然采用了二分法，但是与实际的语言现象有矛盾。因为 $VP \rightarrow VP \text{ Part}$ 对应的语言符号系列是 *pick it*，而这个符号系列并不是英语中一个可以独立存在的结构，显然不宜于把这个符号序列叫做 VP。

⁹ Chomsky, *Syntactic Structures*, Mouton, 1957.

由此可见，不仅在汉语分析中有必要采用多叉树，就是在英语分析中，也有必要采用多叉树。

第三，一些长的句子，如采用二叉树来描述，其层次会多到十层八层，计算机处理在这里这样的多层次的二叉树时，需逐层进行，运算量很大。这是二分法的第三个严重缺陷。

而如果采用多叉树，就可以大大地减少层次，提高计算机处理自然语言的效率。因此，采用多叉树来代替基于二分法的二叉树可以减少在编制程序时的程序量。采用多叉树还有利于抓住句子的主干，把句子的格局清楚地显示出来，便于研究和检查。

基于如上的理由，我在20世纪80年代初期提出了汉语句子的多叉多标记树形图分析法¹⁰，这种分析法又叫做“多叉多标记树形图模型”(Multiple-branched, Multiple-labeled Tree Model)，简称MMT模型；在1981年开发的汉-法/英/日/俄/德多语言机器翻译系统FAJRA中，我采用多叉树来描述汉、法、英、日、俄、德等6种语言，用“多分”(multiple-branching)来代替“二分”(binary-branching)，提高了机器翻译系统的功能。FAJRA系统时世界上第一个把汉语翻译成多种外语的机器翻译系统，开汉译外机器翻译的先河。可见，在自然语言处理中采用多分法是大有好处的。

其实，如果我们多叉树形图看成一种普遍的树形图格式，那么，二叉树便是多叉树形图的一种特殊情况。所谓“多叉”，可以是“三叉”、“四叉”，也可以是“二叉”、“一叉”，它是一种更为一般的形式，而“二叉”只不过当“多叉”的“多”等于“二”时的一种特殊情况罢了。

所以，我们认为，从本质上说来，Chomsky的上下文无关文法多分法的。

Chomsky范式的重写规则形式为

$$A \rightarrow BC$$

$$A \rightarrow a$$

其中，A、B、C都是非终极符号，a是终极符号。Chomsky范式把单个的非终极符号重写为两个非终极符号B和C，便于在自然语言处理中用二叉树来表示自然语言的数据结构。显而易见，Chomsky范式的重写规则是上下文无关文法重写规则 $A \rightarrow \omega$ 中，当 $\omega = BC$ 或者 a 时的一种特殊情况。由于任何符合Chomsky范式的上下文无关文法与重写规则为 $A \rightarrow \omega$ 的上下文无关文法都是等价的，因此，这样的限制并不失一般性。也就是说，在上下文无关文法重写规则 $A \rightarrow \omega$ 中，当 ω 变成AB或者A的时候，就变成了Chomsky范式，因此，我们有理由认为，二分法应当看成多分法的一种特殊情况。

Richard S. Kayne 提出了“线性相应公理”(Linear Correspondence Axiom, LCA)，主张严格实行二分法，尽管Richard S. Kayne提出了严格实行二分法的动因和内在逻辑，但是，从自然语言处理的技术要求来考虑，我们认为，Richard S. Kayne的主张是站不住脚的。

吴道平在他的文章中也对于 Richard S. Kayne 的结论提出质疑。他说：“两分本身是否真的反映了语言结构的本质关系？照作者看来，认为两分就是短语结构的本质关系非常勉强，反证很多”。他还举出了汉语中多重并立结构句子“他的爸爸妈妈弟弟妹妹都住在上海”来证明二分法的弊病，指出，硬要把这个句子中做主语的并立结构“爸爸妈妈弟弟妹妹”首先两分，然而被两分的成分再次两分，实在说不出多少道理来。我们完全赞同吴道平的这种主张。

陆丙甫在1983年指出，如果把自然语言的句子以句法功能划分为“块”的话，那么任一句子“块”的数量只在四至七之间，最多不超过七块。陆丙甫提出：“处理句子的过程中每时每刻脑子里记住的离散结构块不会超过七块左右，也就是说，如果超过了七块左右，那么一定要及时组块以减少块数。而所谓组块就是按照结构模式把一些(通常是连续的)离散块组成一个大块。由此可

¹⁰ 冯志伟，汉语句子的多叉多标记树形图分析法，《人工智能学报》，1983年，第2期。

推知，语流中出现的结构模式都不会超过七项左右的成分”¹¹。

陆丙甫后来又指出，当脑子里记住的离散块超过了四块时，处理后面的材料时难度就急速增加。人们在分段记忆电话号码时，每段不能超过3-4个数字。他还注意到，4左右的语段反映句子结构难度的一个敏感点：平均结构难度不到4和超过4的语段，在难度感觉上会有明显的差别¹²。

陆丙甫关于自然语言句子“块”的数量为四至七之间的限制，也为多分法在语言学理论上提供了有力的支持。

吴道平的文章还讨论了自然语言是否是上下文无关语言的问题。

吴道平介绍了美国计算语言学家 S. Shieber 在《上下文无关性质的反证实例》一文中对于这个问题的研究¹³。

S. Shieber 于1983年发现，在瑞士德语中存在着词序的交叉对应现象，也就是存在着如图9所示的如下的符号串：

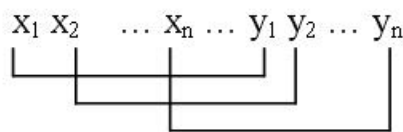


图9 词序的交叉对应

在图7的符号串中， x_1 与 y_1 对应， x_2 与 y_2 对应，...， x_n 与 y_n 对应，上下文无关语法描述不了这样的语言现象。

S. Shieber 指出，在在瑞士德语中有这样的句子：

Jan säid das mer d'chind em Hans es huus lönd hälfed aastriiche
约翰 说 我们 小孩 汉斯 房屋 让 帮助 粉刷
(约翰说，我们让小孩帮助汉斯粉刷房屋)

其中，d'chind (小孩)与动词 lönd (让)相对应，Hans (汉斯)与动词 hälfed (帮助)相对应，huus (房屋)与动词 aastriiche (粉刷)相对应。瑞士德语中这样的语言现象是不能用上下文无关语法来描述的。这样看来，自然语言并不是上下文无关的。

吴道平举出例子说明汉语中存在着非上下文无关的现象。其实，在汉语中，这样的非上下文无关的现象还不少。我们再举两个例子：

- (1) 胡锦涛、温家宝、吴邦国、贾庆林分别担任国家主席、国务院总理、人大委员长、政协主席。
- (2) 昆明、成都、长沙、长春、沈阳、哈尔滨、杭州分别是云南、四川、湖南、吉林、辽宁、黑龙江、浙江的省会。

在句子(1)中，“胡锦涛”与“国家主席”对应，“温家宝”与“国务院总理”对应，“吴邦国”与“人大委员长”对应，“贾庆林”与“政协主席”对应，上下文无关文法不能描述这样

¹¹陆丙甫，无限递归的条件和有限切分，《汉语学习》，Vol.3，1983。

¹²陆丙甫，“组块”与语言结构难度，《世界汉语教学》，No.1。

¹³ Shieber, Stuart M.(1985) “Evidence Against the Context-Freeness of Natural Language”, *Linguistic and Philosophy*, 8: 333-343, 1985.

的对应关系。

在句子(20)中,“昆明”与“云南”对应,“成都”与“四川”对应,“长沙”与“湖南”对应,“长春”与“吉林”对应,“沈阳”与“辽宁”对应,“哈尔滨”与“黑龙江”对应,“杭州”与“浙江”对应,上下文无关语法也不能描述这样的对应关系。

尽管自然语言的大部现象可以使用上下文无关语法来描述,上下文无关语法是生成语法的基础部分;但是,从总体上看,自然语言还不能算上下文无关的,自然语言的性质似乎介于上下文无关与上下文有关之间。Chomsky 在《规则与表达》¹⁴中指出,自然语言可能比上下文有关语言还要复杂,它是 Chomsky 层级上最复杂 0 型语言,这是一种递归可枚举语言 (recursive numerable language)。

自然语言的这种性质反映了它的“计算复杂性”(computational complexity)。关于自然语言的计算复杂性的讨论是语言学理论中一个重要而饶有趣味的问题,我们应当关注这个问题。

参考文献

1. N. Chomsky, 1957, *Syntactic Structures*, Mouton.
2. C. C. Fries, 1964, 英语结构 - 英语句子结构导论 (中译本), 264 页, 商务印书馆。
3. E. A. Nida, 1949, *Morphology*, University of Michigan Press.
4. Kayne, Richard S., 1984, *Connectedness and Binary Branching*, Foris.
5. Shieber, Stuart M., 1985, Evidence Against the Context-Freeness of Natural Language, *Linguistic and Philosophy*, 8: 333-343.
6. 冯志伟, 1983, 汉语句子的多叉多标记树形图分析法,《人工智能学报》,第 2 期。
7. 冯志伟, 2010, 自然语言处理的形式模型, 中国科学技术大学出版社。
8. 陆丙甫, 1983, 无限递归的条件和有限切分,《汉语学习》,第 3 期。
9. 陆丙甫, 2009, “组块”与语言结构难度,《世界汉语教学》,第 1 期。
10. 吴道平, 2009, “两分法”浅析,《东方语言学》,总第 3 期。

¹⁴ N. Chomsky, *Rules and Representations*, Columbia University Press, 1980/2005.