计算语言学

第13讲 句法分析I

刘群
中国科学院计算技术研究所
liuqun@ict.ac.cn

中国科学院研究生院2002~2003学年第二学期课程讲义

内容提要 1

- 什么是句法分析
- 与形式语言句法分析的比较
- 上下文无关语法的分析策略
  - 自顶向下分析法
  - 自底向上分析法
  - 左角分析法

内容提要 2

- 上下文无关语法的分析算法
  - 移进－归约算法
  - Marcus确定性分析算法
  - CYK算法
  - Earley算法
  - Tomita算法
  - Chart算法
- 概率上下文无关语法
- 组块分析与部分分析

什么是句法分析

- 句法分析（Parsing）和句法分析器（Parser)
- 句法分析是从单词串得到句法结构的过程；
- 不同的语法形式，对应的句法分析算法也不尽相同；
- 由于短语结构语法（特别是上下文无关语法）应用得最为广泛，因此以短语结构树为目标的句法分析器研究得最为彻底；
- 很多其他形式语法对应的句法分析器都可以通过对短语结构语法的句法分析器进行简单的改造得到。
- 本讲义将主要介绍上下文无关语法的句法分析器。

与形式语言句法分析的比较

- 形式语言一般是人工构造的语言，是一种确定性的语言，即对于语言中的任何一个句子，只有唯一的一种句法结构是合理的，即使语法本身存在歧义，也往往通过人为的方式规定一种合理的解释。如程序语言中的if…then if…then…else…结构，往往都人为规定 else 子句与最接近的 if 子句配对；
- 而在自然语言中，歧义现象是天然地大量存在着的，而且这些歧义的解释往往都有可能是合理的，因此，对歧义现象的处理是自然语言句法分析器最本质的要求。
- 由于要处理大量的歧义现象，导致自然语言句法分析器的复杂程度远高于形式语言的句法分析器。

句法结构歧义的消解 1

- 人们正常交流中所使用的语言，放在特定的环境下看，一般是没有歧义的，否则人们将无法交流（某些特殊情况如幽默或双关语除外）
- 如果不考虑语言所处的环境和语言单位的上下文，将会发现语言的歧义现象无所不在；
- 结论：一般来说，语言单位的歧义现象在引入更大的上下文范围或者语言环境时总是可以被被消解的。句法分析的核心任务就是消解一个句子在句法结构上的歧义。

句法结构的歧义消解 2

- 我是县长。
  我是县长派来的。
- 咬死了猎人的狗跑了。
  就是这条狼咬死了猎人的狗。
- 小王和小李的妹妹结婚了。
  小王和小李的妹妹都结婚了。

例子－语法

- 小王和小李的妹妹结婚了

规则：

S → NP VP

NP → NP C NP

NP → N

NP → NP de N

VP → V le

词典：

小王：N

小李：N

和：C

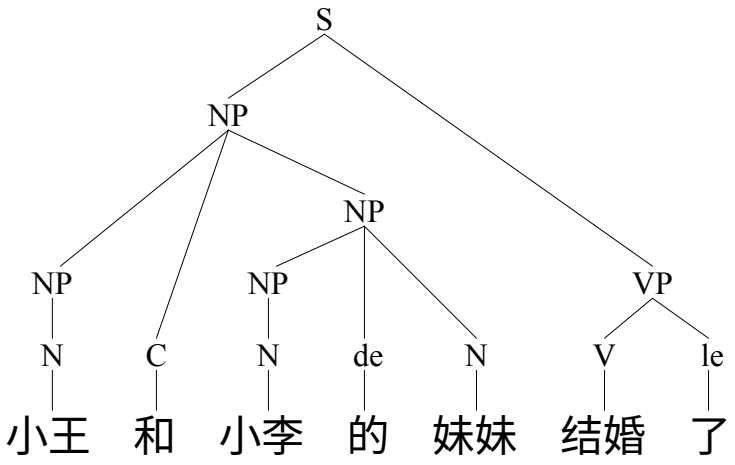
妹妹：N

结婚：V

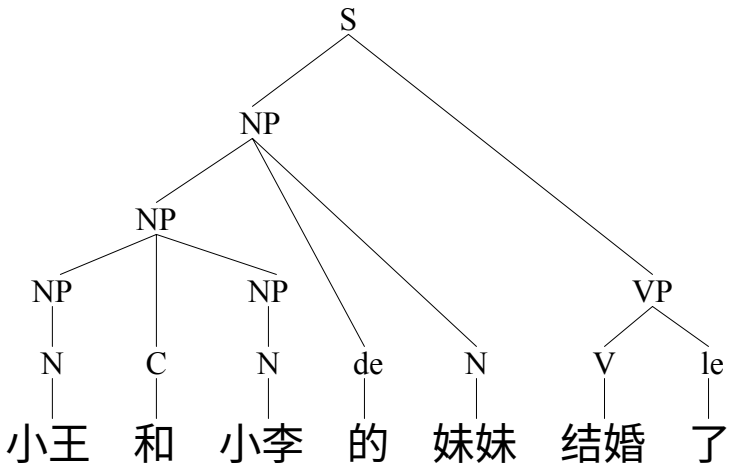
了：le

的：de

例子－分析结果之一



例子－分析结果之二



另一个例子

- 我是县长派来的

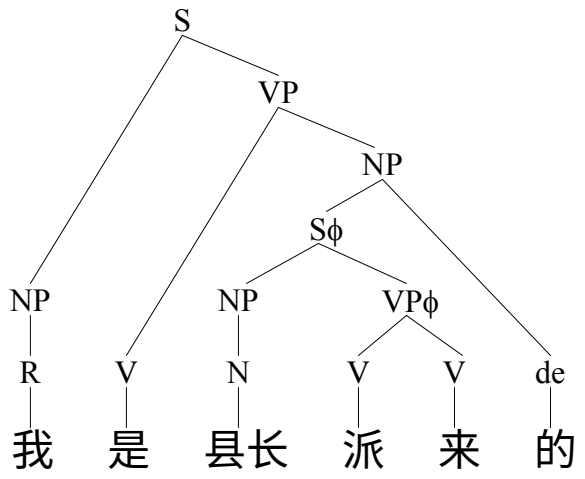
规则：

S → NP VP
NP → R
NP → N
NP → Sϕ de
VP → V NP
Sϕ → NP VPϕ
VPϕ → V V

词典：

我：R
县长：N
是：V
派：V
来：V
的：de

另一个例子－分析结果



句法分析的基本策略

句法分析通常采用的策略有：

- 自顶向下分析法；
- 自底向上分析法；
- 左角分析法；
- 其他策略。

上下文无关语法的分析算法

常见的上下文无关语法的句法分析算法：

1. CYK算法；
2. 移进－归约算法；
3. Marcus确定性分析算法；
4. Earley算法；
5. Tomita算法（GLR算法、富田算法）；
6. Chart算法（图分析算法、线图分析算法）；

自顶向下和自低向上分析法 1

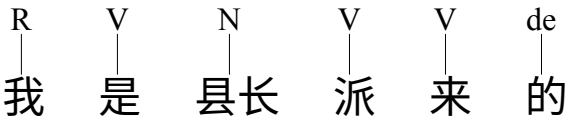
- 句法分析的过程也可以理解为句法树的构造过程
- 所谓自顶向下分析法也就是先构造句法树的根结点，再逐步向下扩展，直到叶结点；
- 所谓自底向上分析法也就是先构造句法树的叶结点，再逐步向上合并，直到根结点。

自顶向下和自低向上分析法 2

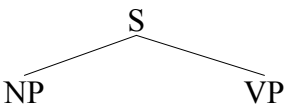
- 自顶向下的方法又称为基于预测的方法，也就是说，这种方法是先产生对后面将要出现的成分的预期，然后再通过逐步吃进待分析的字符串来验证预期。如果预期得到了证明，就说明待分析的字符串可以被分析为所预期的句法结构。如果某一个环节上预期出了差错，那就要用另外的预期来替换（即回溯）。如果所有环节上所有可能的预期都被吃进的待分析字符串所“反驳”，那就说明待分析的字符串不可能是一个合法的句子，分析失败。
- 自底向上的方法也叫基于归约的方法。就是说，这种方法是先逐步吃进待分析字符串，把它们从局部到整体层层归约为可能的成分。如果整个待分析字符串被归约为开始符号S，那么分析成功。如果在某个局部证明不可能有任何从这里把整个待分析字符串归约为句子的方案，那么就需要回溯。

自顶向下分析法－示例1

查词典

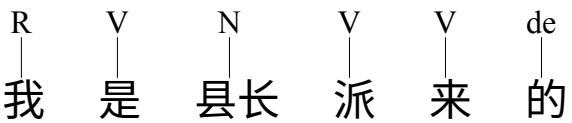


自顶向下分析法－示例2

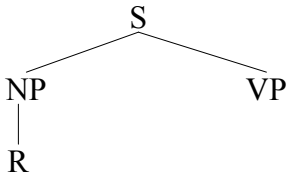


使用规则：

S → NP VP

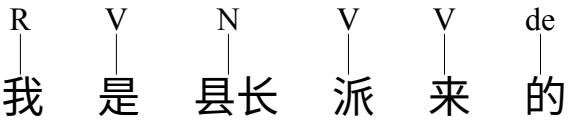


自顶向下分析法－示例3

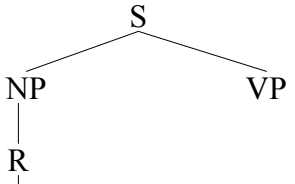


使用规则：

NP→R

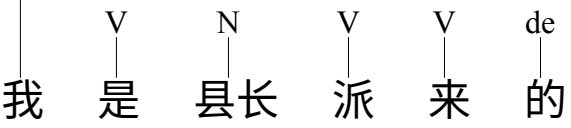


自顶向下分析法－示例4

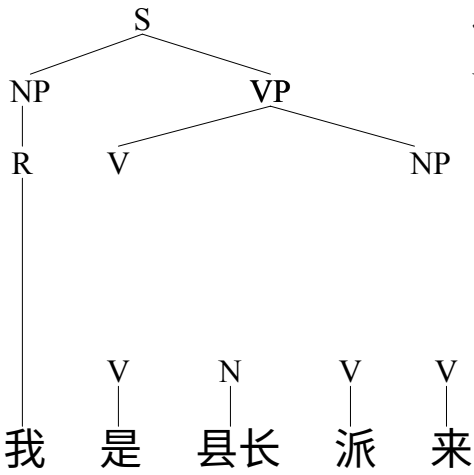


词典匹配成功：

我：R



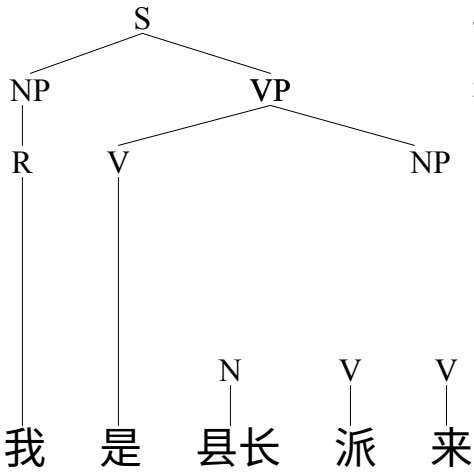
自顶向下分析法－示例5



使用规则：

VP→V NP

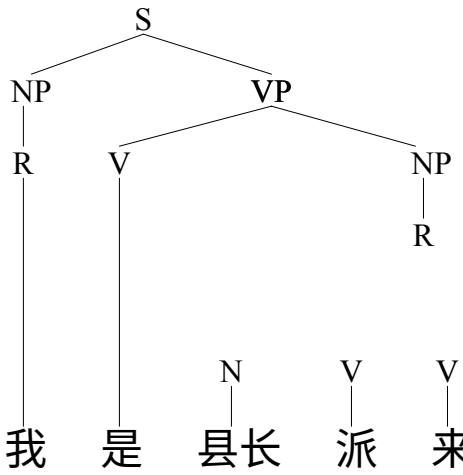
自顶向下分析法－示例6



词典匹配成功：

是：V

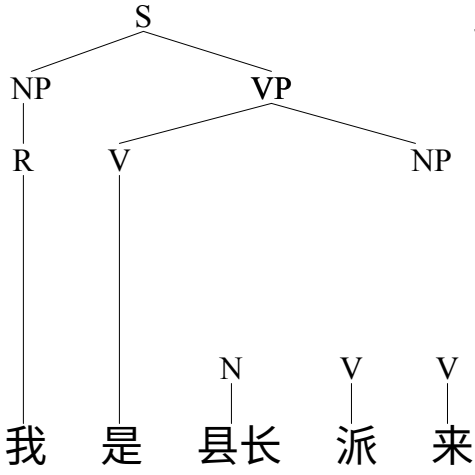
自顶向下分析法－示例7



使用规则：

NP→R

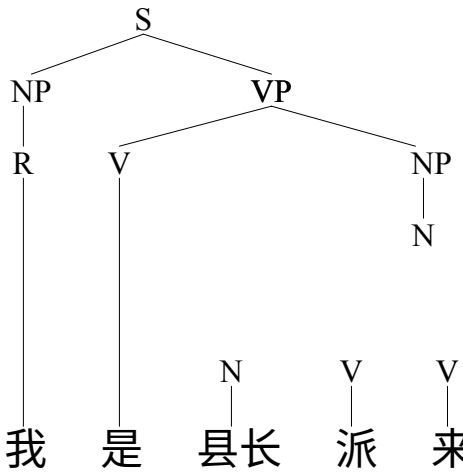
自顶向下分析法－示例8



与词典不匹配

回溯

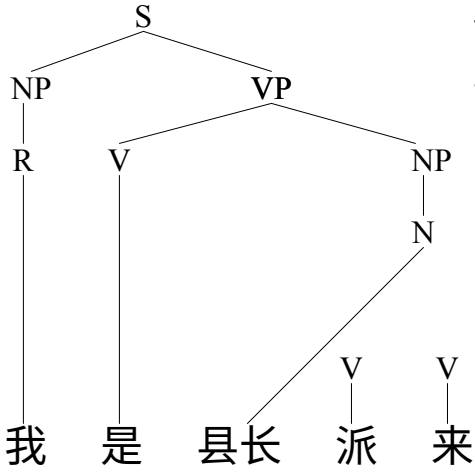
自顶向下分析法－示例9



使用下一条规则：

NP→N

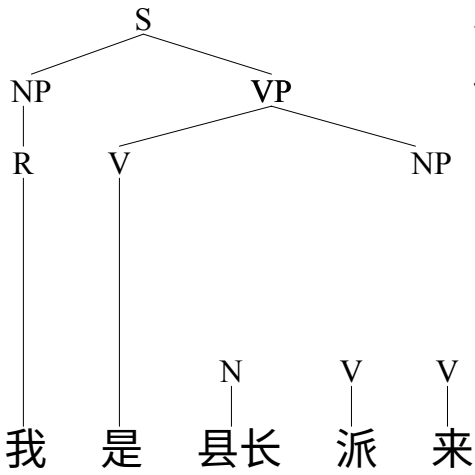
自顶向下分析法－示例10



词典匹配成功：

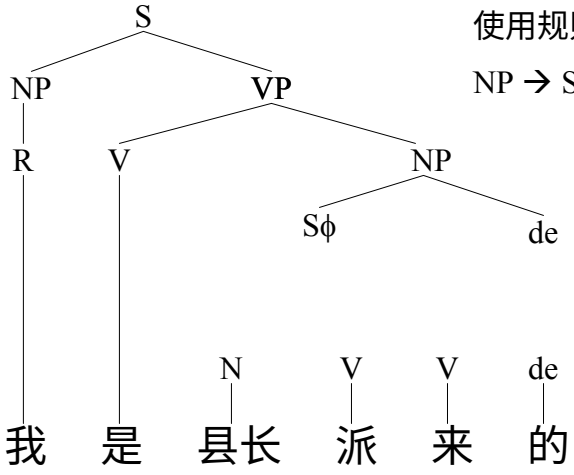
县长：N

自顶向下分析法－示例11



句法树扩展完毕，
但句子没有完全
匹配成功，回溯

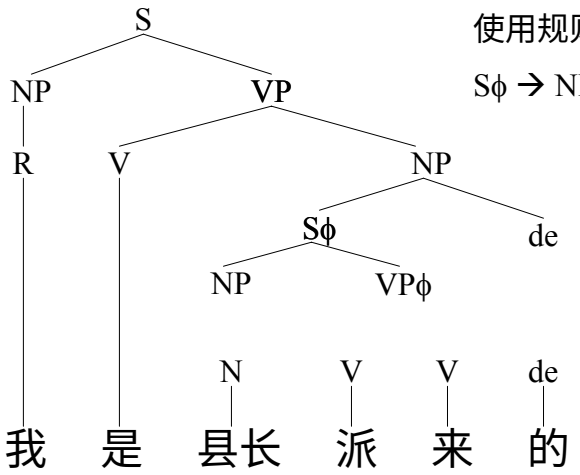
自顶向下分析法－示例12



使用规则：

NP → Sϕ de

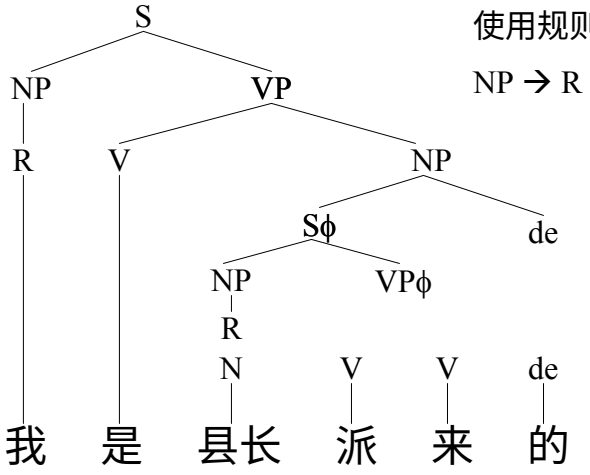
自顶向下分析法－示例13



使用规则：

Sϕ → NP Vϕ

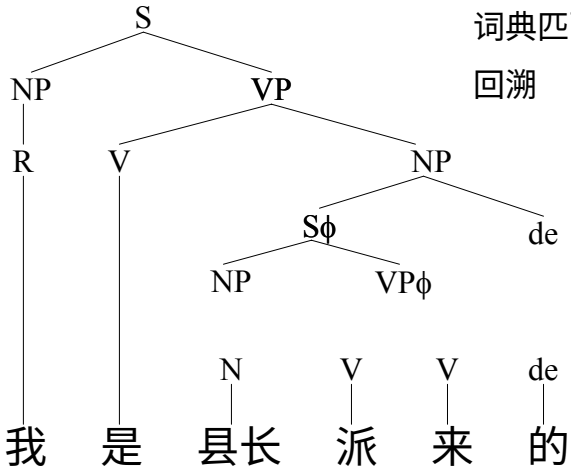
自顶向下分析法－示例14



使用规则：

NP → R

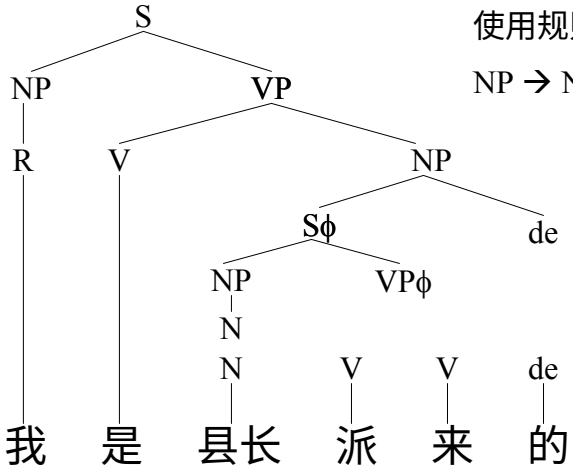
自顶向下分析法－示例15



词典匹配失败

回溯

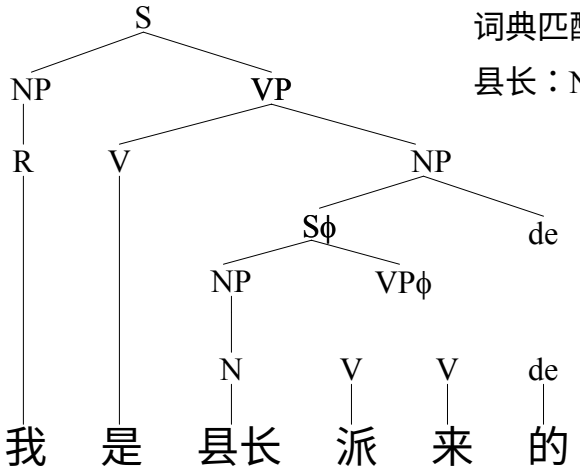
自顶向下分析法－示例16



使用规则：

NP → N

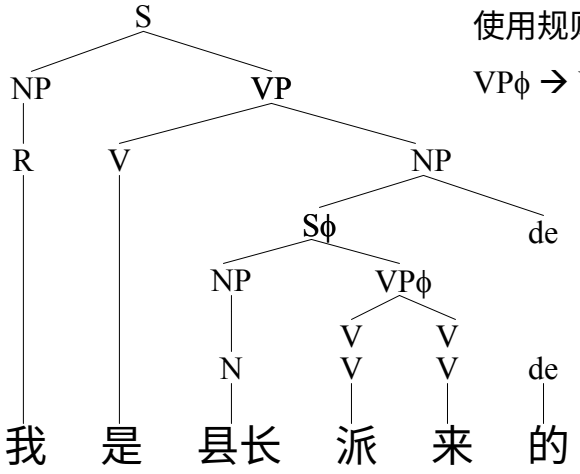
自顶向下分析法－示例17



词典匹配成功：

县长：N

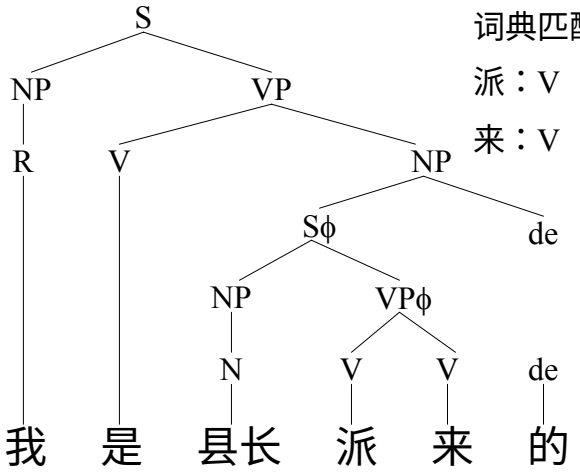
自顶向下分析法－示例18



使用规则：

VPϕ → V V

自顶向下分析法－示例19

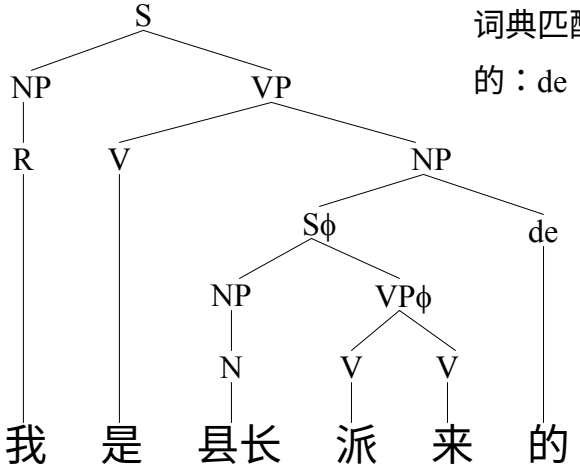


词典匹配成功：

派：V

来：V

自顶向下分析法－示例20



词典匹配成功：

的：de

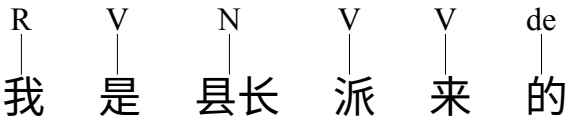
句法树扩展完毕

恰好句子匹配完成

分析成功

自底向上分析法－示例1

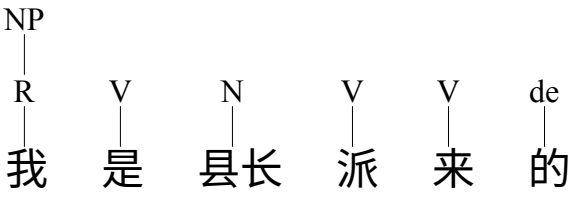
查词典



自底向上分析法－示例2

使用规则：

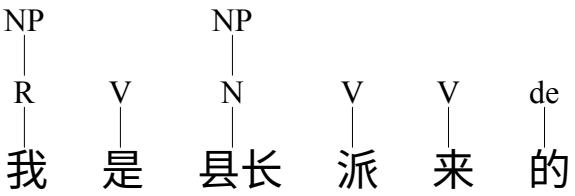
NP→R



自底向上分析法－示例3

使用规则：

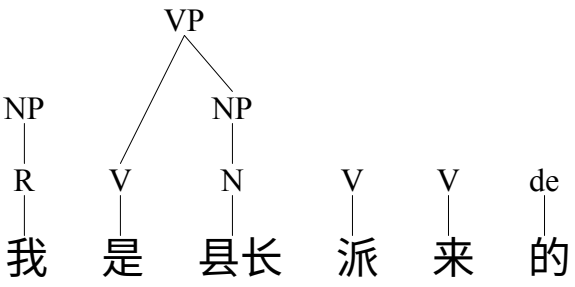
NP→N



自底向上分析法－示例4

使用规则：

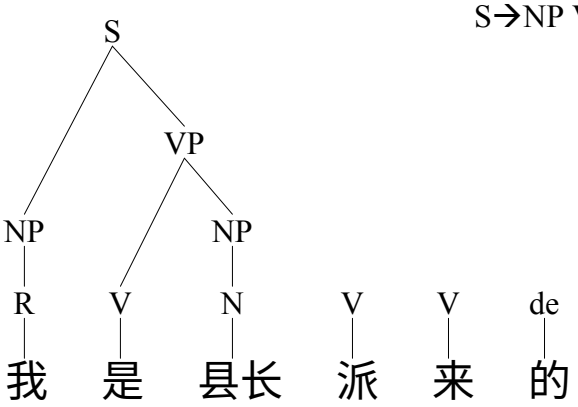
VP→V NP



自底向上分析法－示例5

使用规则：

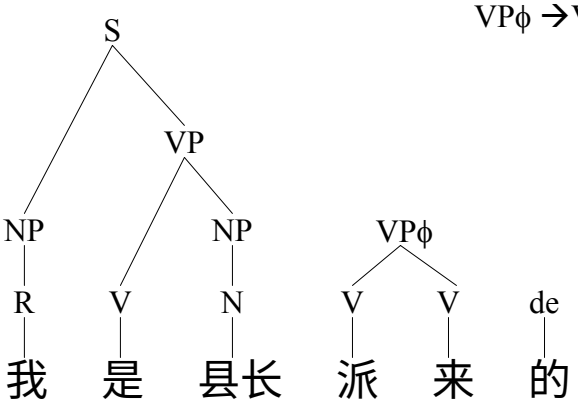
S→NP VP



自底向上分析法－示例6

使用规则：

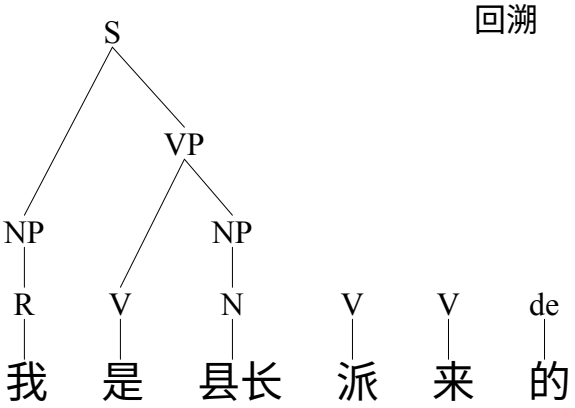
VPϕ →V V



自底向上分析法－示例7

无规则可用

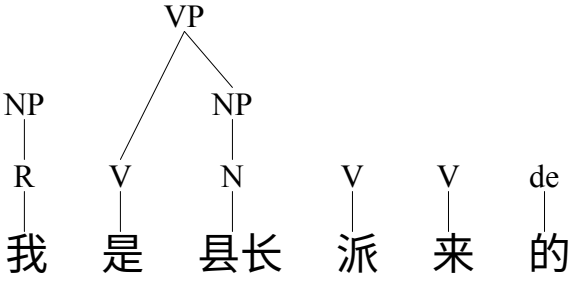
回溯



自底向上分析法－示例8

无规则可用，

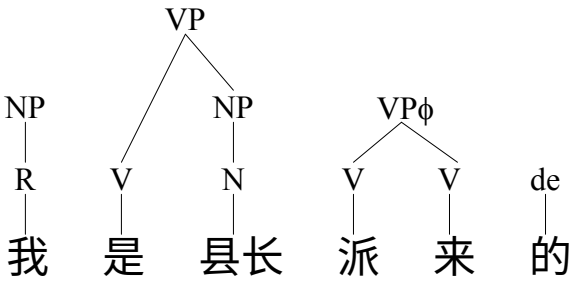
回溯



自底向上分析法－示例9

使用规则：

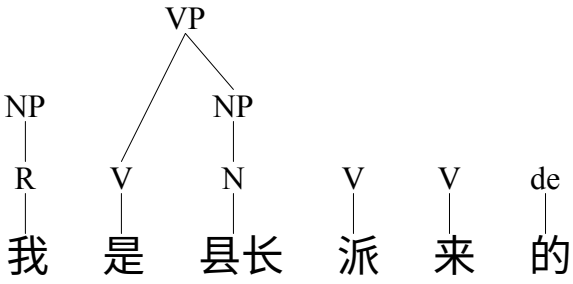
VPϕ →V V



自底向上分析法－示例10

无规则可用，

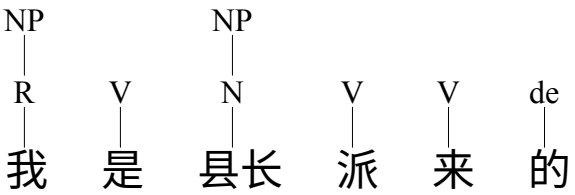
回溯



自底向上分析法－示例11

无规则可用，

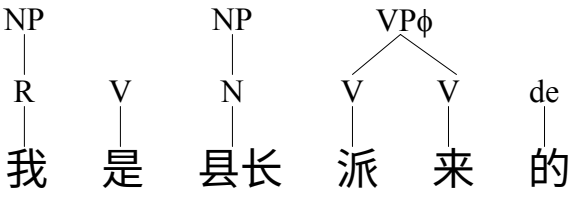
回溯



自底向上分析法－示例12

使用规则：

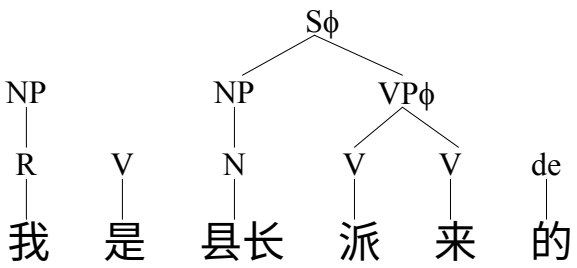
VPϕ →V V



自底向上分析法－示例13

使用规则：

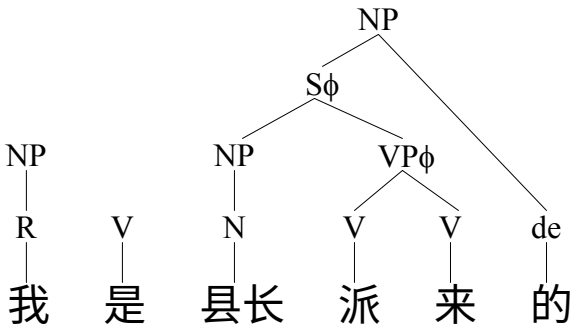
Sϕ →NP VPϕ



自底向上分析法－示例14

使用规则：

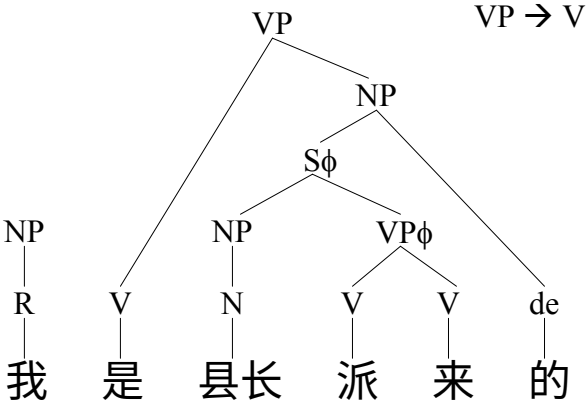
NP → Sϕ de



自底向上分析法－示例15

使用规则：

VP → V NP

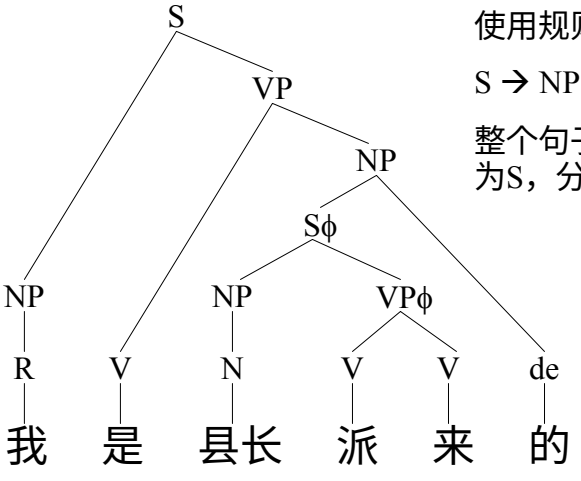


自底向上分析法－示例16

使用规则：

S → NP VP

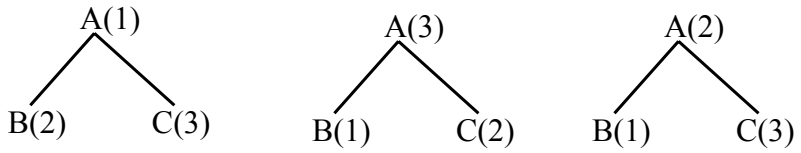
整个句子归结为S，分析成功



左角分析法－概述

- 左角分析法是一种自顶向下和自底向上相结合的方法
- 所谓“左角(Left Corner)”是指任何一个句法子树中左下角的那个符号
- 比较：

自顶向下分析法　自底向上分析法　左角分析法



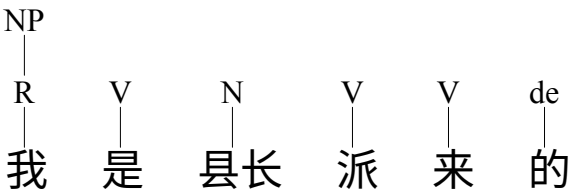
左角分析法－示例1

查词典

左角分析法－示例2

使用规则：

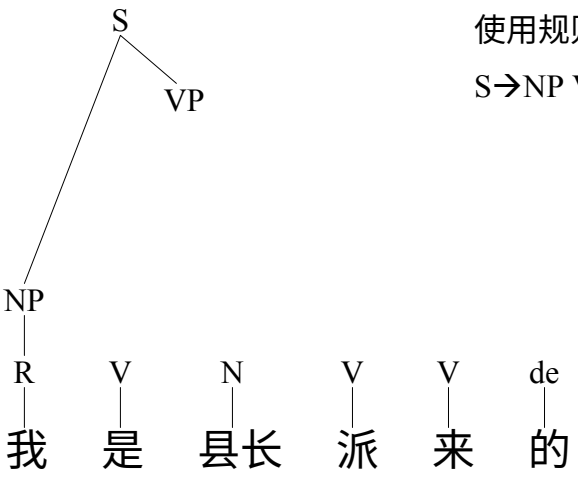
NP→R



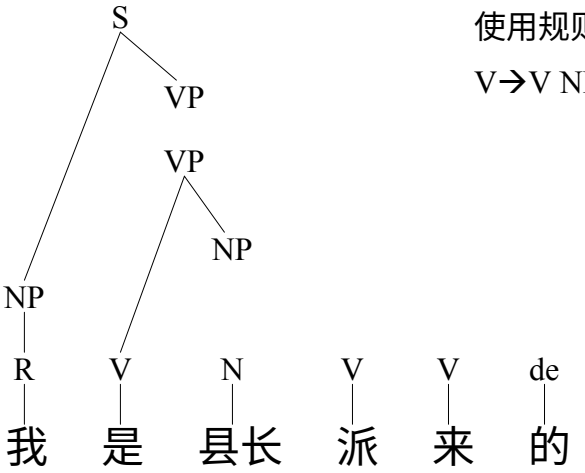
左角分析法－示例3

使用规则：

S→NP VP



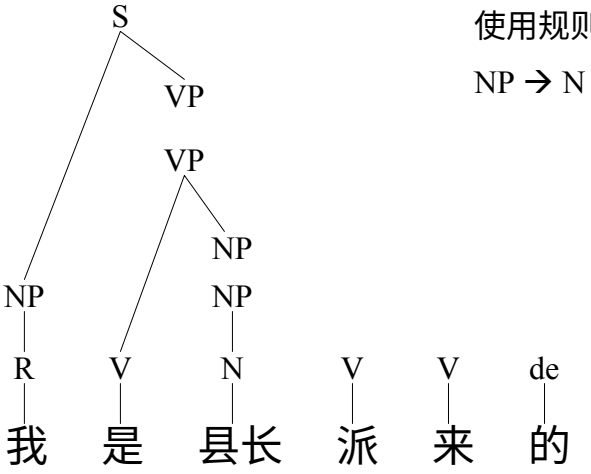
左角分析法－示例4



使用规则：

V→V NP

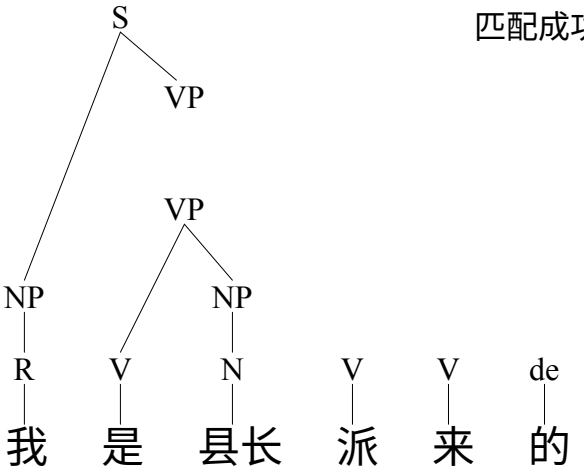
左角分析法－示例5



使用规则：

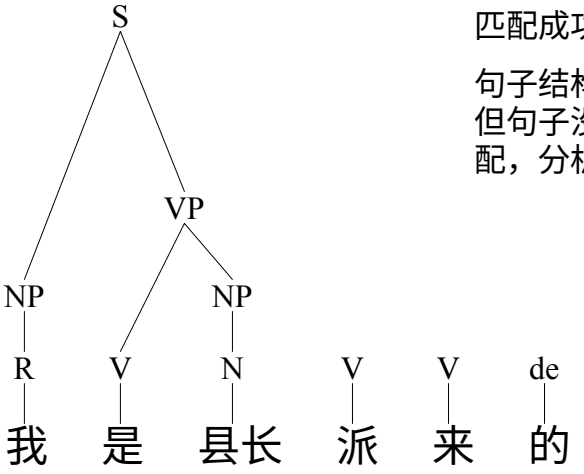
NP → N

左角分析法－示例6



匹配成功

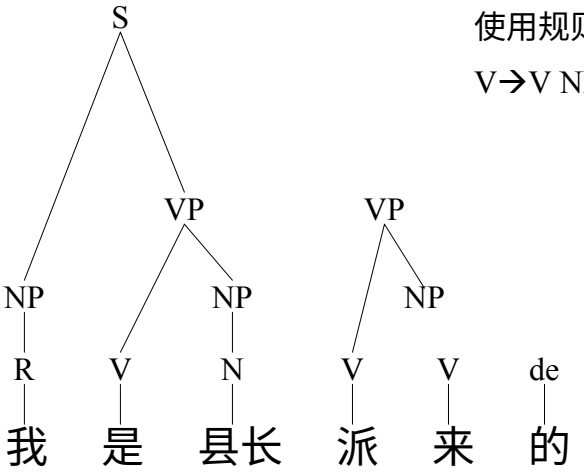
左角分析法－示例7



匹配成功

句子结构生成完毕，但句子没有全部匹配，分析失败

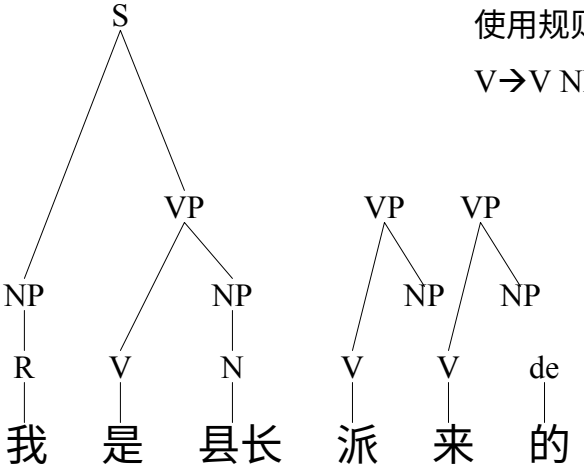
左角分析法－示例8



使用规则：

V→V NP

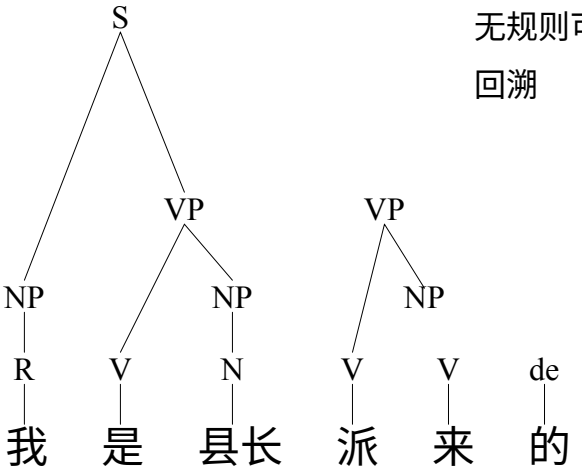
左角分析法－示例9



使用规则：

V→V NP

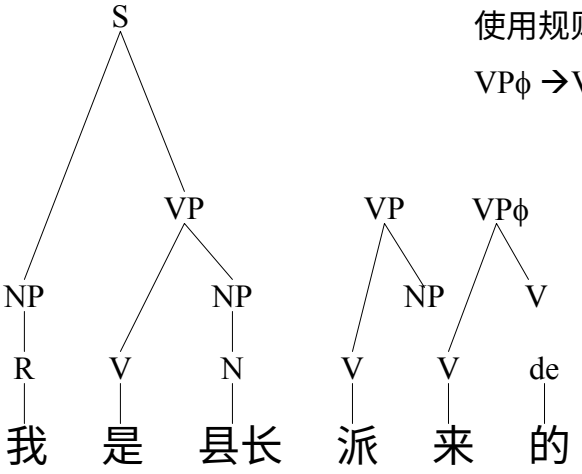
左角分析法－示例10



无规则可用

回溯

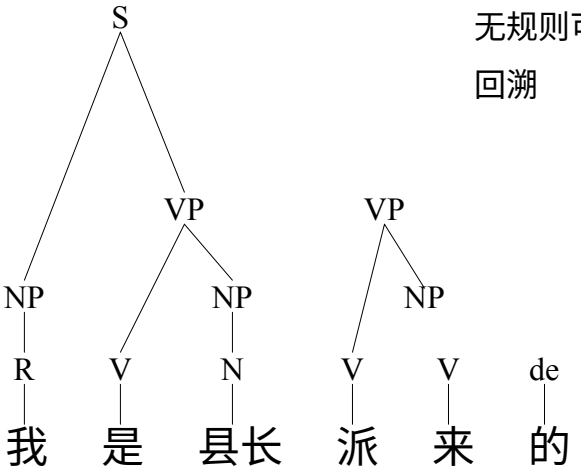
左角分析法－示例11



使用规则

VPϕ →V V

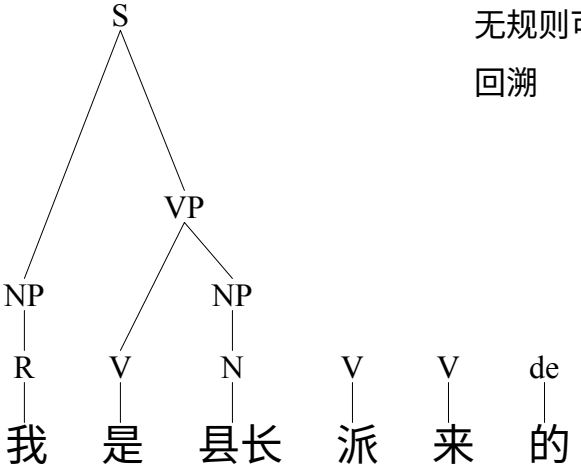
左角分析法－示例12



无规则可用

回溯

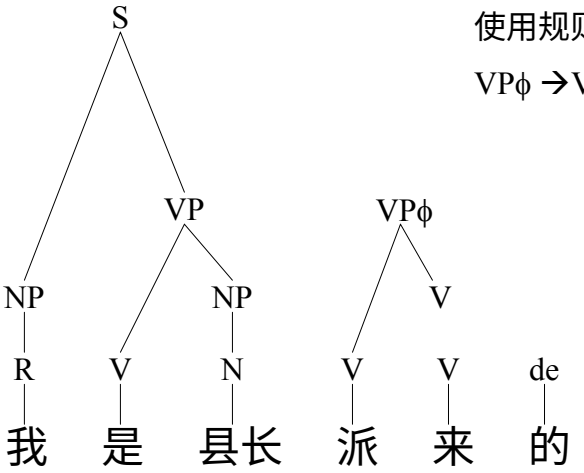
左角分析法－示例13



无规则可用

回溯

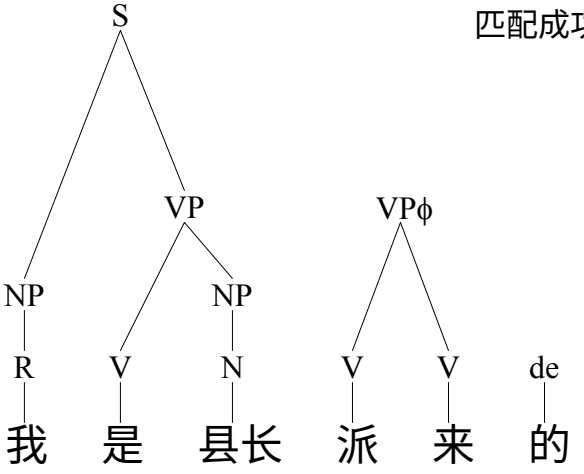
左角分析法－示例14



使用规则

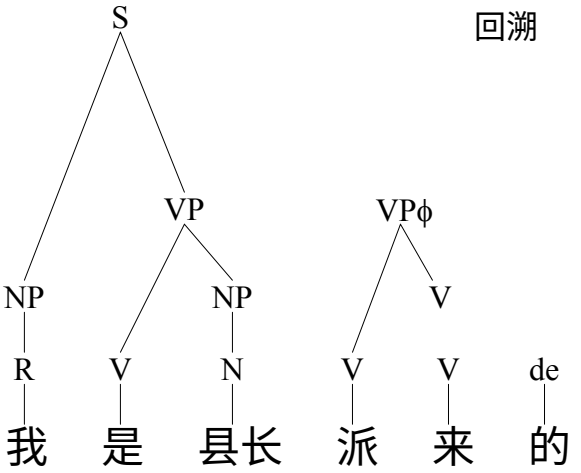
VPϕ →V V

左角分析法－示例15



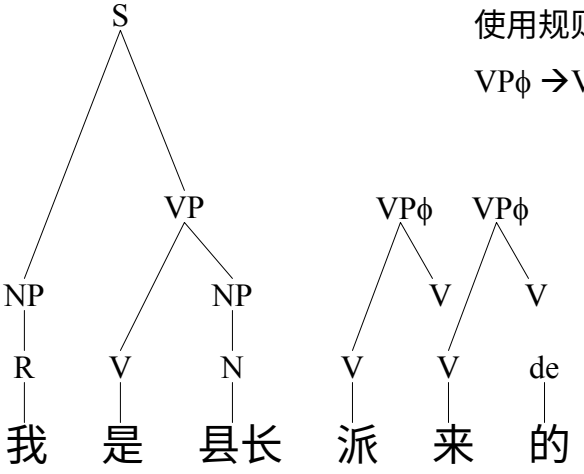
匹配成功

左角分析法－示例16



回溯

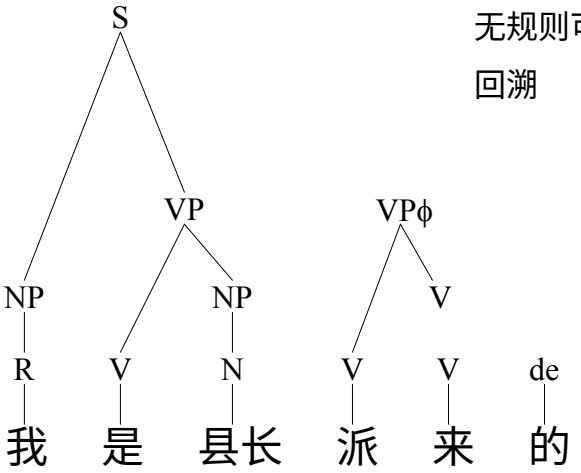
左角分析法－示例17



使用规则

VPϕ →V V

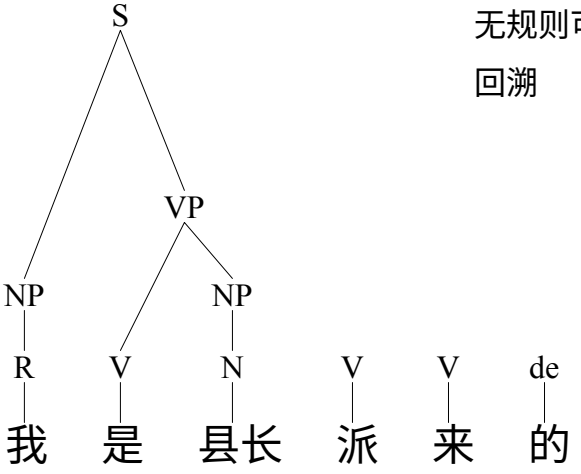
左角分析法－示例18



无规则可用

回溯

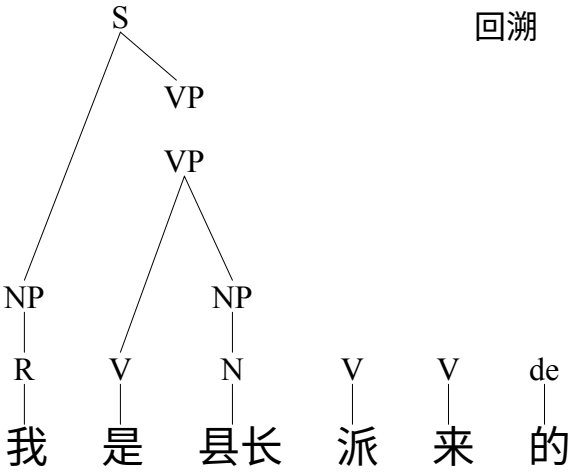
左角分析法－示例19



无规则可用

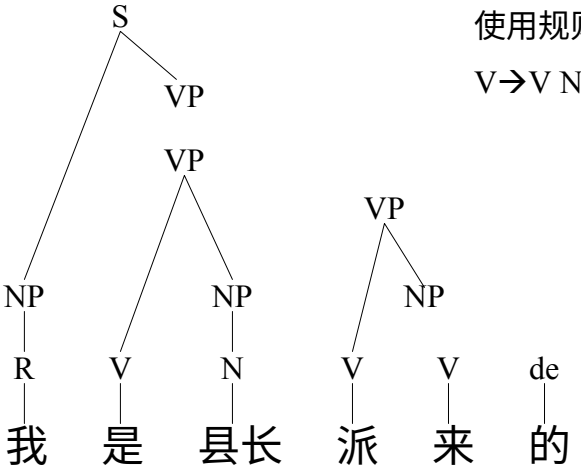
回溯

左角分析法－示例20



回溯

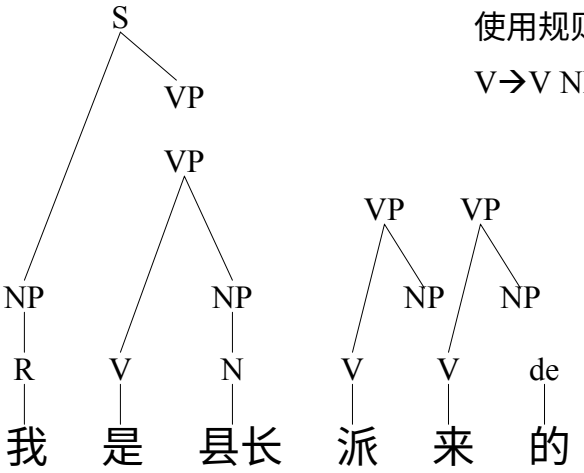
左角分析法－示例21



使用规则：

V→V NP

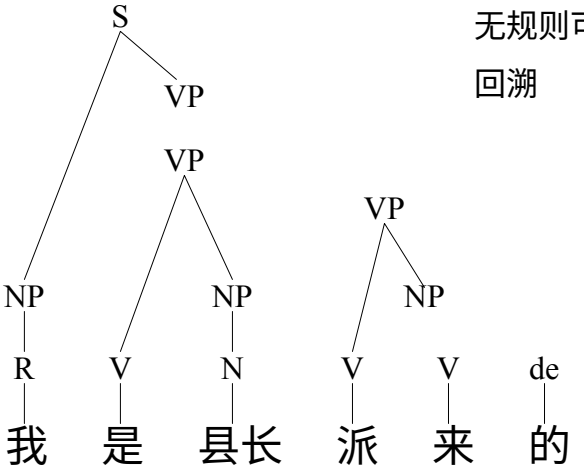
左角分析法－示例22



使用规则：

V→V NP

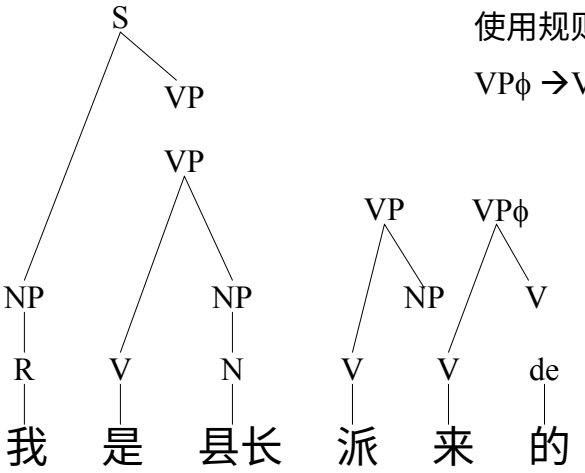
左角分析法－示例23



无规则可用

回溯

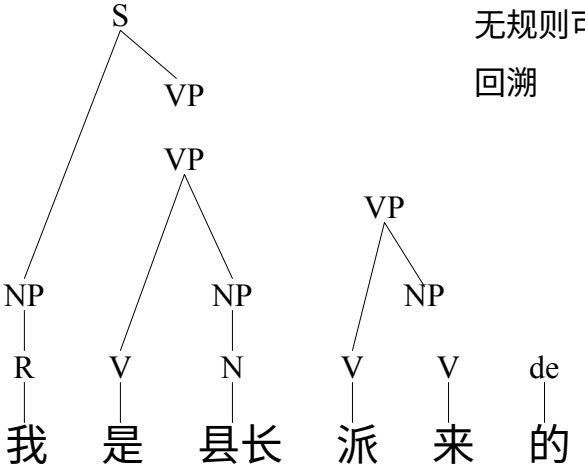
左角分析法－示例24



使用规则

VPϕ →V V

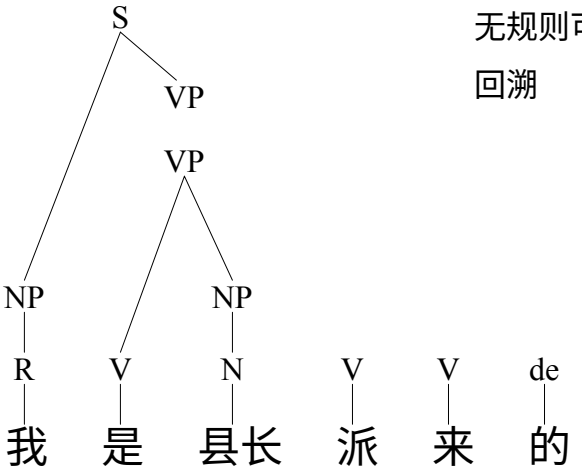
左角分析法－示例25



无规则可用

回溯

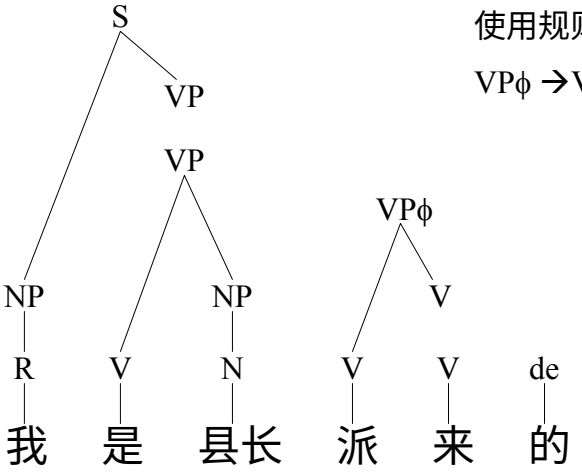
左角分析法－示例26



无规则可用

回溯

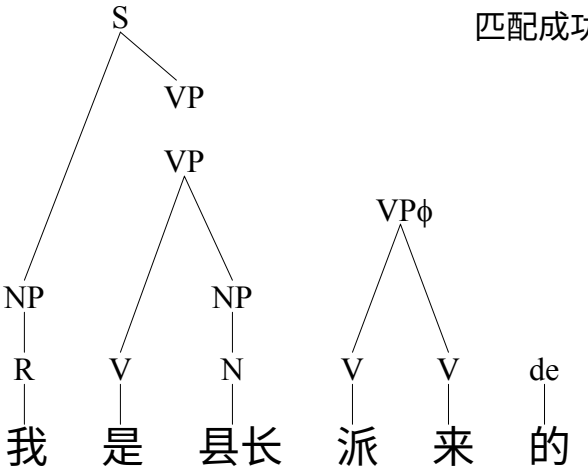
左角分析法－示例27



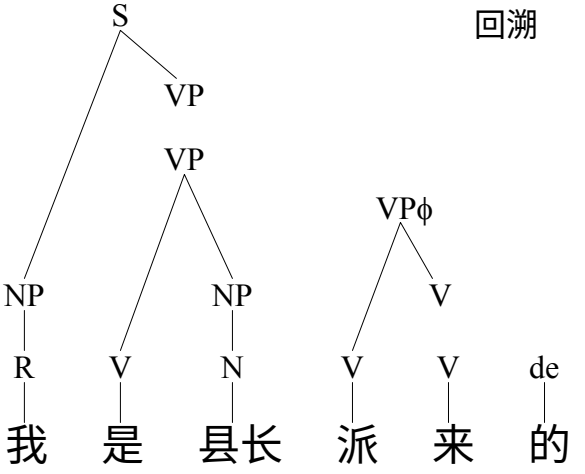
使用规则

VPϕ →V V

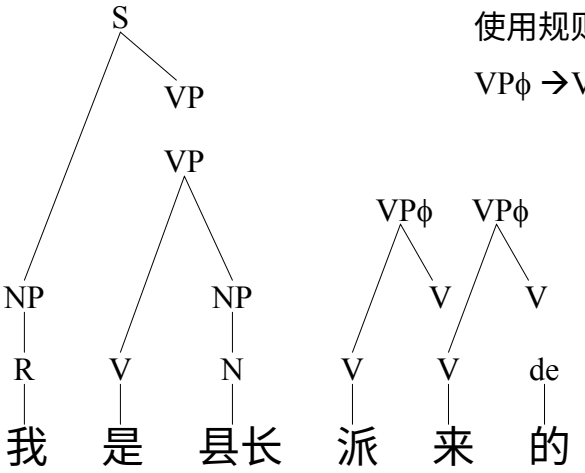
左角分析法－示例28



左角分析法－示例29



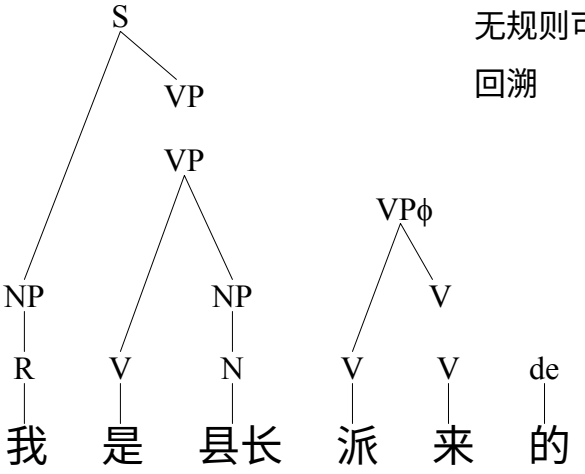
左角分析法－示例30



使用规则

VPϕ →V V

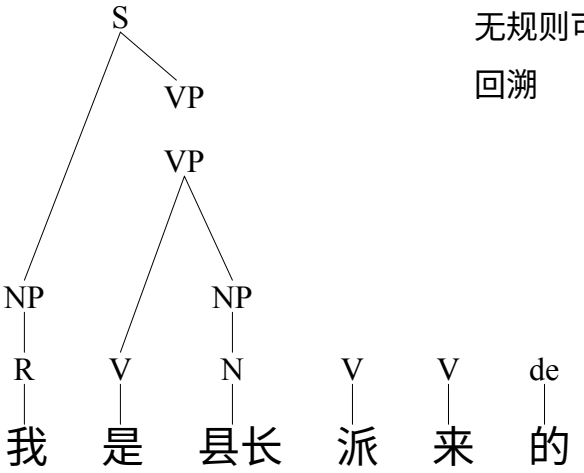
左角分析法－示例31



无规则可用

回溯

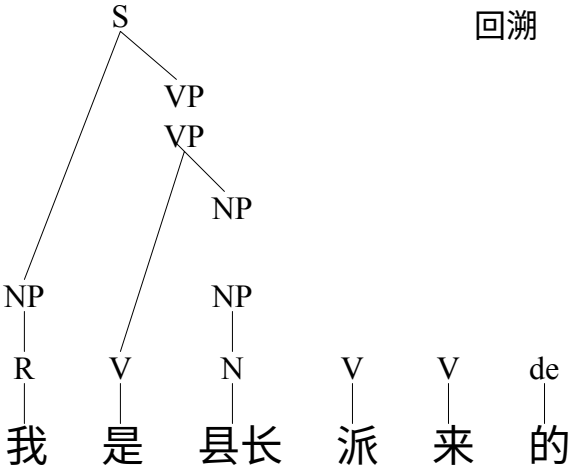
左角分析法－示例32



无规则可用

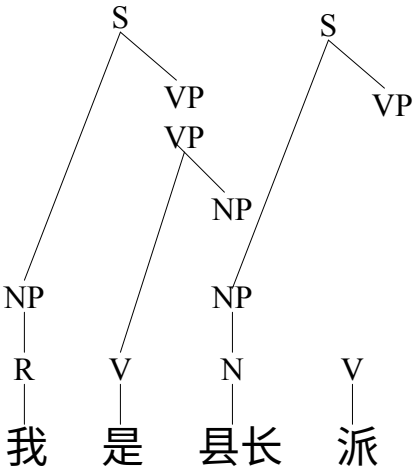
回溯

左角分析法－示例33



回溯

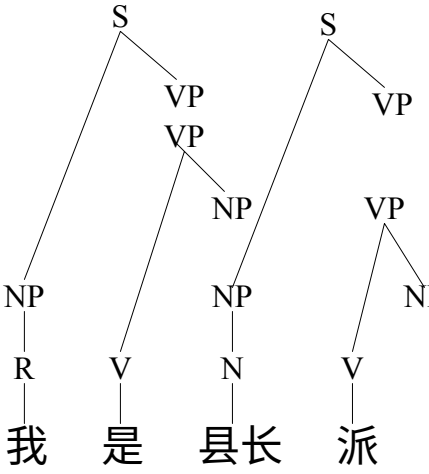
左角分析法－示例34



使用规则

S→NP VP

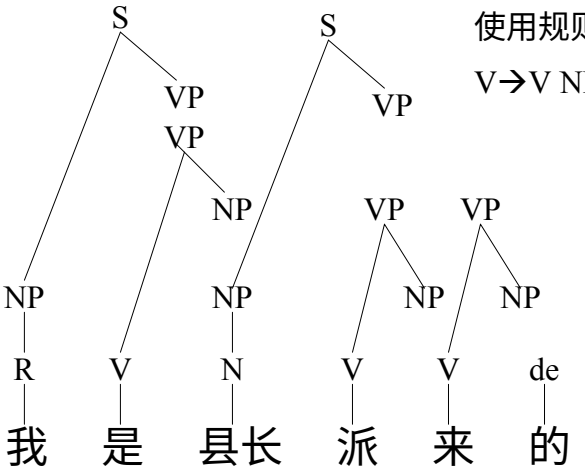
左角分析法－示例35



使用规则：

V→V NP

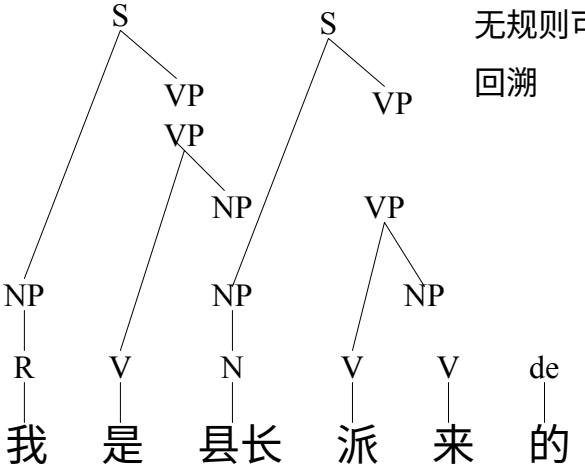
左角分析法－示例36



使用规则：

V→V NP

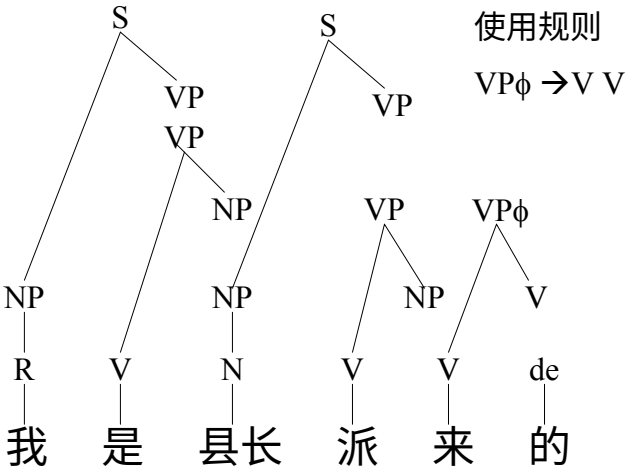
左角分析法－示例37



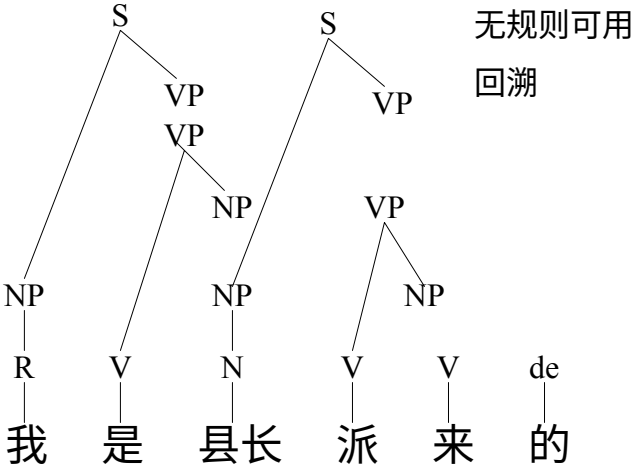
无规则可用

回溯

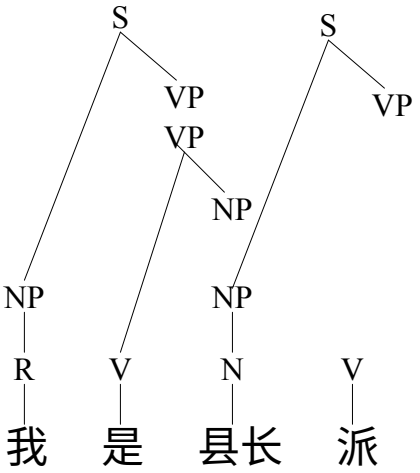
左角分析法－示例38



左角分析法－示例39



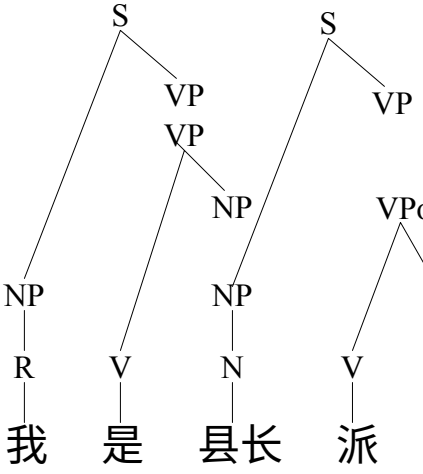
左角分析法－示例40



无规则可用

回溯

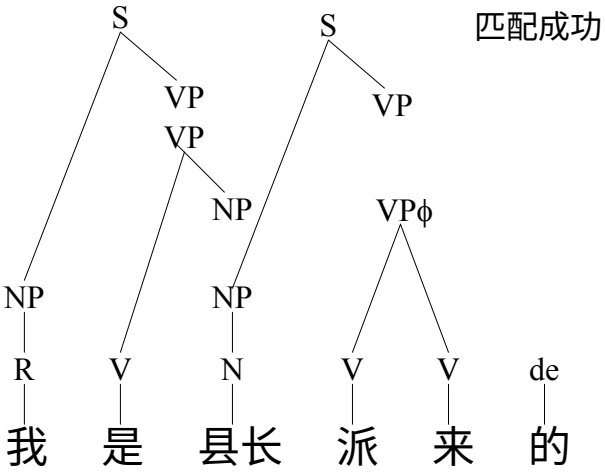
左角分析法－示例41



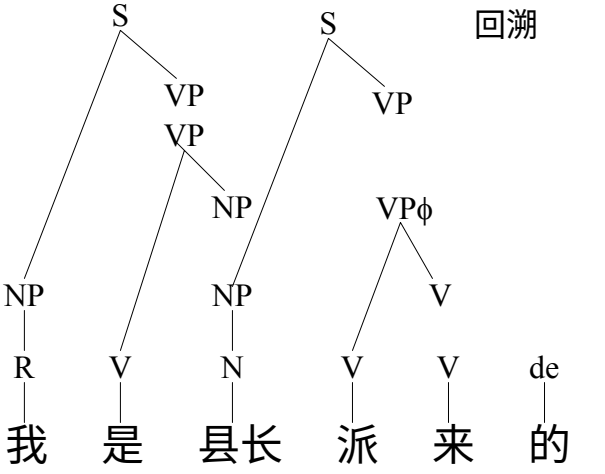
使用规则

VPϕ →V V

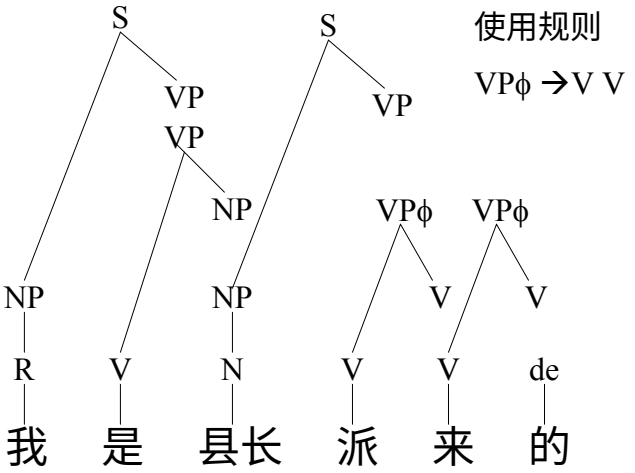
左角分析法－示例42



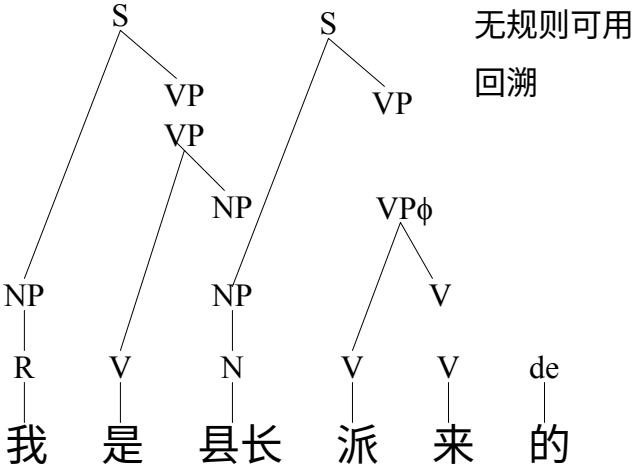
左角分析法－示例43



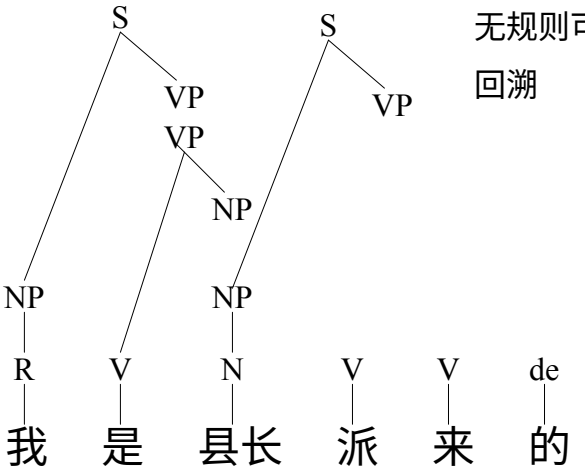
左角分析法－示例44



左角分析法－示例45



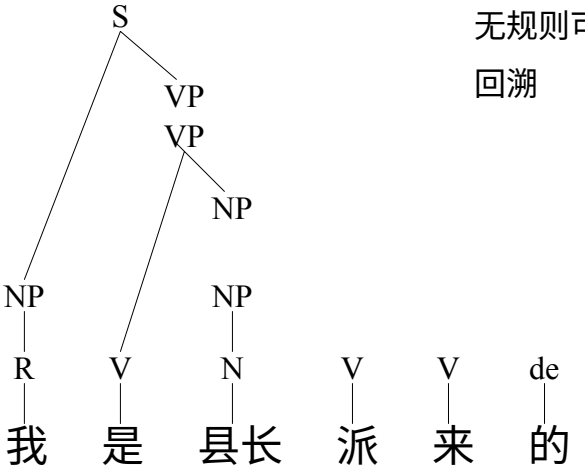
左角分析法－示例46



无规则可用

回溯

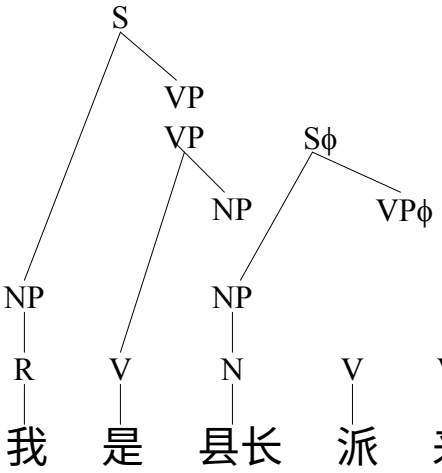
左角分析法－示例47



无规则可用

回溯

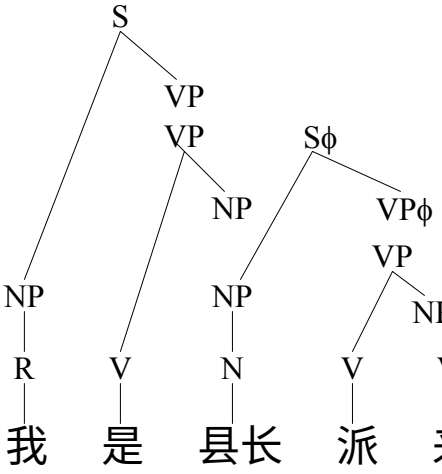
左角分析法－示例48



使用规则：

Sϕ →NP VPϕ

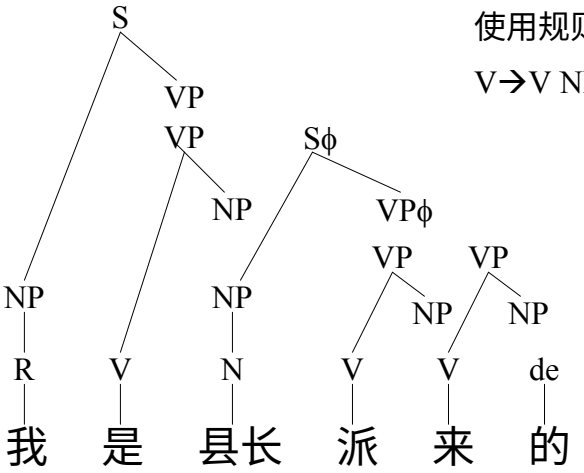
左角分析法－示例49



使用规则：

V→V NP

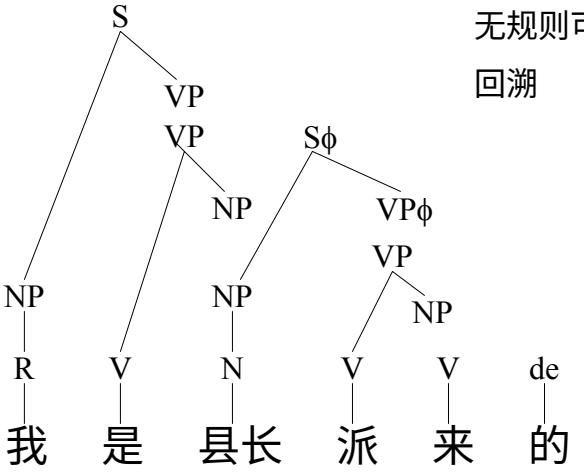
左角分析法－示例50



使用规则：

V→V NP

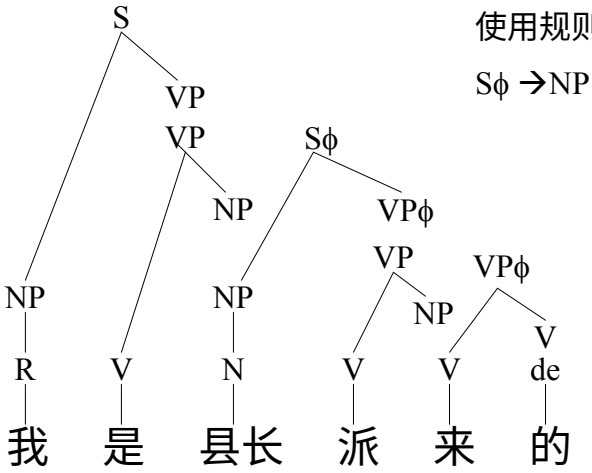
左角分析法－示例51



无规则可用

回溯

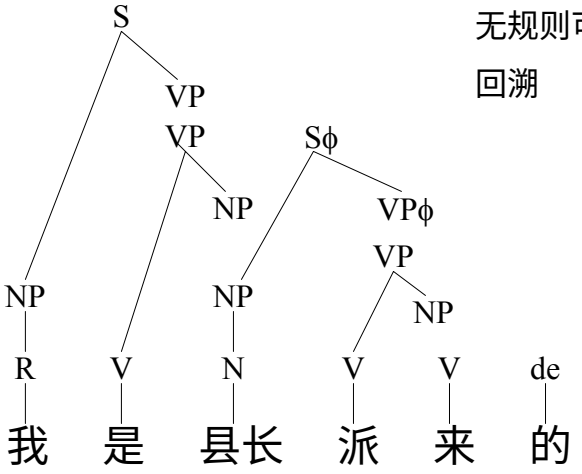
左角分析法－示例52



使用规则：

Sϕ →NP VPϕ

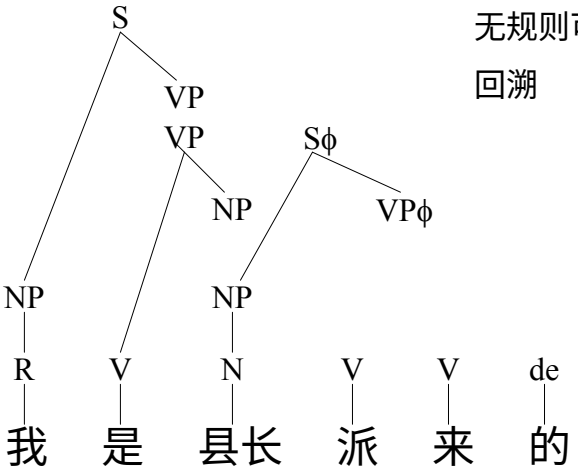
左角分析法－示例53



无规则可用

回溯

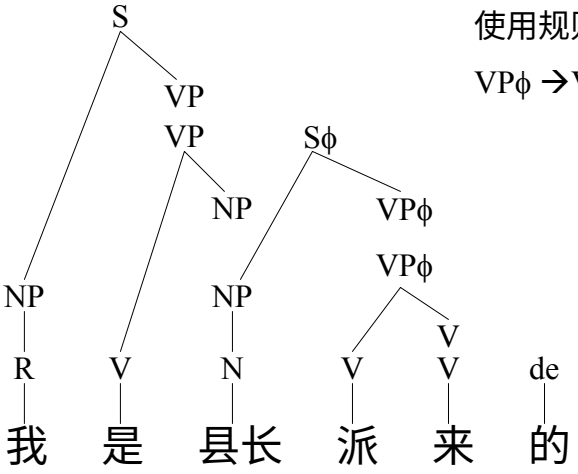
左角分析法－示例54



无规则可用

回溯

左角分析法－示例55

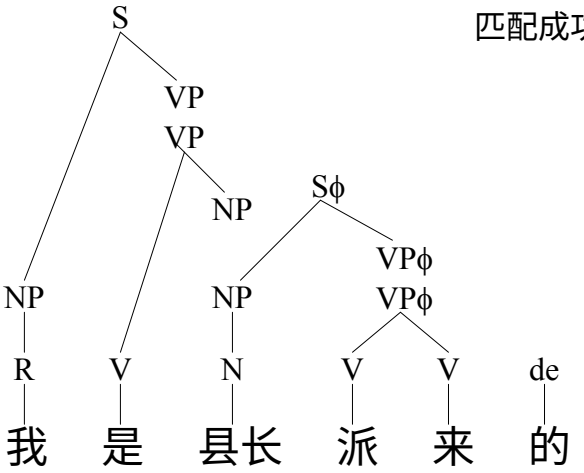


使用规则：

VPϕ →V V

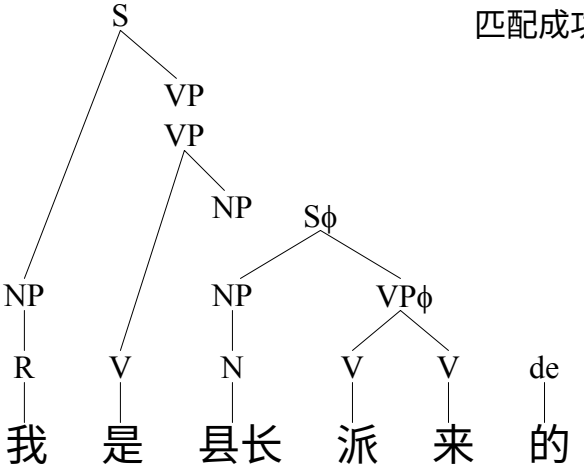
左角分析法－示例56

匹配成功

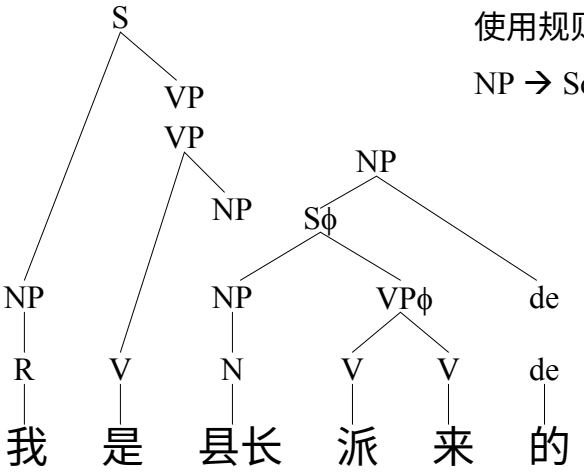


左角分析法－示例57

匹配成功



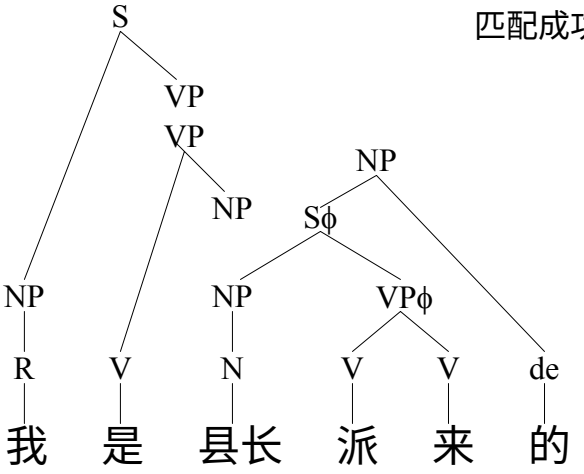
左角分析法－示例58



使用规则：

NP → Sϕ de

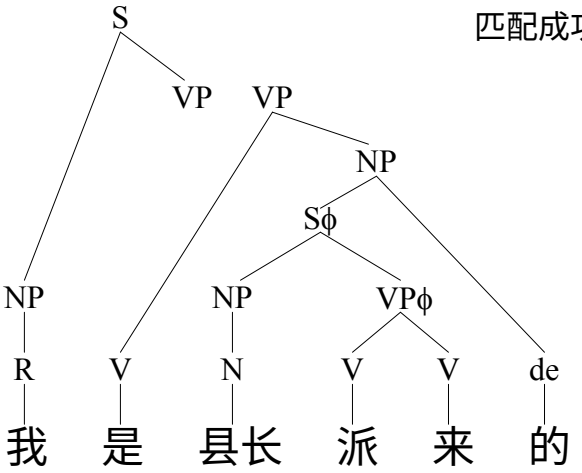
左角分析法－示例59



匹配成功

左角分析法－示例60

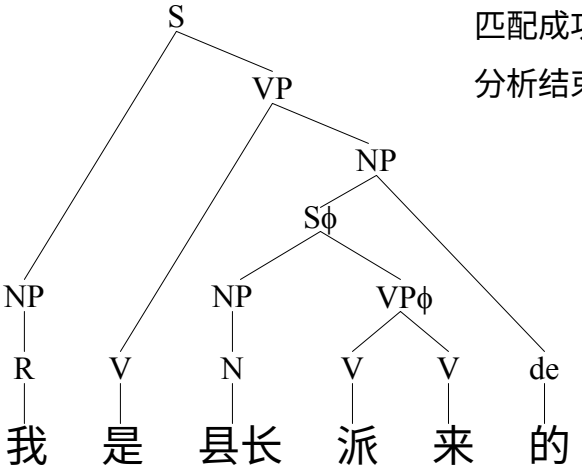
匹配成功



左角分析法－示例61

匹配成功

分析结束



移进－归约算法：概述

- 移进－归约算法：Shift-Reduce Algorithm
- 移进－归约算法类似于下推自动机的LR分析算法
- 移进－归约算法的基本数据结构是堆栈
- 移进－归约算法的四种操作：
  - 移进：从句子左端将一个终结符移到栈顶
  - 归约：根据规则，将栈顶的若干个符号替换成一个符号
  - 接受：句子中所有词语都已移进到栈中，且栈中只剩下一个符号S，分析成功，结束
  - 拒绝：句子中所有词语都已移进栈中，栈中并非只有一个符号S，也无法进行任何归约操作，分析失败，结束

移进－归约算法：举例

步骤	栈	输入	操作	规则
1	#	我 是 县长	移进	
2	# 我	是 县长	归约	N→ 我
3	# R	是 县长	归约	NP→R
4	# NP	是 县长	移进	
5	# NP 是	县长	归约	V→是
6	# NP V	县长	移进	
7	# NP V 县长		归约	N→县长
8	# NP V N		归约	NP→N
9	# NP V NP		归约	VP→V NP
10	# NP VP		归约	S→NP VP
11	# S		接受	

移进－归约算法：冲突

- 移进－归约算法中有两种形式的冲突：
  - 移进－归约冲突：既可以移进，又可以归约
  - 归约－归约冲突：可以使用不同的规则归约
- 冲突解决方法：回溯
- 回溯导致的问题：
  - 回溯策略：对于互相冲突的各项操作，给出一个选择顺序
  - 断点信息：除了在堆栈中除了保存非终结符外，还需要保存断点信息，使得回溯到该断点时，能够恢复堆栈的原貌，并知道还可以有哪些可选的操作

移进－归约算法：示例 1

- 回溯策略：
  - 移进－归约冲突：先归约，后移进
  - 归约－归约冲突：规则事先排序，先执行排在前面的规则
- 断点信息：
  - 当前规则：标记当前归约操作所使用的规则序号
  - 候选规则：记录在当前位置还有哪些规则没有使用（由于这里规则是排序的，所以这一条可以省略）
  - 被替换结点：归约时被替换的结点，以便回溯时恢复

移进－归约算法：示例 2

- 给规则排序并加上编号：

规则：

(7) NP → R
(8) NP → N
(9) NP → Sϕ de
(10) VPϕ → V V
(11) VP → V NP
(12) Sϕ → NP VPϕ
(13) S → NP VP

词典：

(1) R→我
(2) N →县长
(3) V →是
(4) V →派
(5) V →来
(6) de →的

移进－归约算法：示例 3

步骤	栈	输入	操作	规则
1	#	我 是 县长 派 来 的	移进	
2	# 我	是 县长 派 来 的	归约	(1) R→ 我
3	# R (1)	是 县长 派 来 的	归约	(7) NP→R
4	# NP (7)	是 县长 派 来 的	移进	
5	# NP (7) 是	县长 派 来 的	归约	(3) V→是
6	# NP (7) V (3)	县长 派 来 的	移进	
7	# NP (7) V (3)县长	派 来 的	归约	(2) N→县长
8	# NP (7) V (3) N (2)	派 来 的	归约	(8) NP→N
9	# NP (7) V (3) NP (8)	派 来 的	归约	(11) VP→V NP
10	# NP (7) VP (11)	派 来 的	归约	(13) S→NP VP
11	# S (13)	派 来 的	移进	

移进－归约算法：示例 4

步骤	栈	输入	操作	规则
12	# S (13) 派	来 的	归约	(4) V→派
13	# S (13) V(4)	来 的	移进	
14	# S (13) V(4) 来	的	归约	(5) V→来
15	# S (13) V(4) V(5)	的	归约	(10) VPϕ → V V
16	# S (13) VPϕ (10)	的	移进	
17	# S (13) VPϕ (10) 的		归约	(6) de→的
18	# S (13) VPϕ (10) de(6)		回溯	
19	# S (13) VPϕ (10) 的		回溯	
20	# S (13) VPϕ (10)	的	回溯	
21	# S (13) V(4) V (5)	的	回溯	
22	# S (13) V(4) 来	的	回溯	

移进－归约算法：示例 5

步骤	栈	输入	操作	规则
23	# S (13) V(4)	来 的	回溯	
24	# S (13) 派	来 的	回溯	
25	# S (13)	派 来 的	回溯	
26	# NP (7) VP (11)	派 来 的	移进	(13) S→NP VP
27	# NP (7) VP (11) 派	来 的	归约	(4) V→派
28	# NP (7) VP (11) V(4)	来 的	移进	
29	# NP (7) VP (11) V(4) 来	的	归约	(5) V→来
30	# NP (7) VP (11) V(4) V(5)	的	归约	(10) VPϕ → V V
31	# NP (7) VP (11) VPϕ (10)	的	移进	
32	# NP (7) VP (11) VPϕ (10) 的		归约	(6) de→的
33	# NP (7) VP (11) VPϕ (10) de(6)		回溯	

移进－归约算法：示例 6

步骤	栈	输入	操作	规则
34	# NP (7) VP (11) VPϕ (10) 的		回溯	
35	# NP (7) VP (11) VPϕ (10)	的	回溯	
36	# NP (7) VP (11) V(4) V(5)	的	回溯	
37	# NP (7) VP (11) V(4) 来	的	回溯	
38	# NP (7) VP (11) V(4)	来 的	回溯	
39	# NP (7) VP (11) 派	来 的	回溯	
40	# NP (7) VP (11)	派 来 的	回溯	
41	# NP (7) VP (11)	派 来 的	回溯	
42	# NP (7) V (3) NP (8)	派 来 的	移进	
43	# NP (7) V (3) NP (8) 派	来 的	归约	(4) V→派
44	# NP (7) V (3) NP (8) V(4)	来 的	移进	

移进－归约算法：示例 7

步骤	栈	输入	操作	规则
45	# NP (7) V (3) NP (8) V(4)来	的	归约	(5) V→来
46	# NP (7) V (3) NP (8) V(4) V(5)	的	归约	(10) VPϕ → V V
47	# NP (7) V (3) NP (8) VPϕ (10)	的	归约	(12) Sϕ → NP VPϕ
48	# NP (7) V (3) Sϕ (12)	的	移进	
49	# NP (7) V (3) Sϕ (12) 的		归约	(6) de→的
50	# NP (7) V (3) Sϕ (12) de (6)		归约	(9) NP → Sϕ de
51	# NP (7) V (3) NP (9)		归约	(11) VP→V NP
52	# NP (7) VP (11)		归约	(13) S→NP VP
53	# S (13)		接受	

说明：为简洁起见，这个例子中没有给出堆栈中被替换结点信息，但必须注意到这些信息是必需的，否则归约操作将无法回溯。

移进－归约算法：特点

- 移进－归约算法是一种自底向上的分析算法
- 为了得到所有可能的分析结果，可以在每次分析成功时都强制性回溯，直到分析失败
- 可以看到，采用回溯算法将导致大量的冗余操作，效率非常低

移进－归约算法的改进

- 如果在出现冲突（移进－归约冲突和归约－归约冲突）时能够减少错误的判断，将大大提高分析的效率
- 引入规则：通过规则，给出在特定条件（栈顶若干个符号和待移进的单词）应该采取的动作
- 引入上下文：考虑更多的栈顶元素和更多的待移进单词来写规则
- 引入缓冲区（Marcus算法）：是一种确定性的算法，没有回溯，但通过引入缓冲区，可以延迟作出决定的时间

CYK算法－概述

- CYK算法：Cocke-Younger-Kasami算法
- CYK算法是一种并行算法，不需要回溯；
- CYK算法建立在Chomsky范式的基础上
  - Chomsky范式的规则只有两种形式：A→BC A→x 这里A,B,C是非终结符，x是终结符
  - 由于后一种形式实际上就是词典信息，在句法分析之前已经进行了替换，所以在分析中我们只考虑形如A→BC形式的规则
  - 由于任何一个上下文无关语法都可以转化成符合Chomsky范式的语法，因此CYK算法可以应用于任何一个上下文无关语法

CYK算法－数据结构 1

6	S					
5		VP				
4			NP			
3	S		Sϕ			
2		VP		VPϕ		
1	NP,R	V	NP,N	V	V	de
	1	2	3	4	5	6
	我	是	县长	派	来	的

CYK算法－数据结构 2

- 一个二维矩阵：{ P(i , j) }
  - 每一个元素P(i , j)对应于输入句子中某一个跨度（Span）上所有可能形成的短语的非终结符的集合
  - 横坐标 i：该跨度左侧第一个词的位置
  - 纵坐标 j：该跨度包含的词数
- 上图中P(3,1)={NP,N}表示“县长”可以归约成N和NP，P(3,3)={Sϕ}表示“县长 派 来”可以规约成Sϕ

CYK算法：算法描述

1. 对i=1…n，j=1（填写第一行，长度为1）
   对于每一条规则A→Wi，
   将非终结符A加入集合P(i,j)；
2. 对j=2…n（填写其他各行，长度为j）
   对i=1…n-j+1（对于所有起点i）
   对k=1…j-1（对于所有左子结点长度k）
   对每一条规则A→BC，
   如果B ∈ P(i,k)且C∈P(i+k,j-k)
   那么将非终结符A加入集合P(i,j)
3. 如果S∈P(1,n)，那么分析成功，否则分析失败

CYK算法：特点

- 本质上是一种自底向上分析法；
- 采用广度优先的搜索策略；
- 采用并行算法，不需要回溯，没有冗余的操作；
- 时间复杂度$O(n^3)$；
- 由于采用广度优先搜索，在歧义较多时，必须分析到最后才知道结果，无法采用启发式策略进行改进。

Earley算法－概述

- Earley算法也是一种并行算法，不需要回溯；
- 类似于CYK算法，Earley算法中也通过一个二维矩阵来存放已经分析过的结果；
- Earley算法的一个重要贡献是引入了点规则，进一步减少了规则匹配中的冗余操作；
- Earley算法是一种自顶向下的分析算法。

Earley算法：点规则

- 所谓点规则，是在规则的右部的终结符或非终结符之间的某一个位置上加上一个圆点，表示规则右部被匹配的程度
- 例子：
  - VP→ · V NP 表示这条规则还没有被匹配
  - VP→ V · NP 表示这条规则右部的V已经匹配成功，而NP还没有被匹配
  - VP→ V NP · 表示这条已被完全匹配，并形成了一个短语VP

Earley算法：数据结构

- 数据结构：二维矩阵{E(i,j)}，其中每个元素是一个点规则的集合，用来存放句子中单词i到单词j这个跨度上所分析得到的所有点规则
- 还是以“我 是 县长 派 来 的”为例：

  E(0,0)={S→ · NP VP，NP→ · N，NP → · R，
  NP→ · Sϕ de，Sϕ→ · NP VPϕ ……}

  E(0,1)={N→我 · ，NP→N · ，S→NP · VP ,
  VP→ · V V, VP → · V NP …… }
- Earley算法就是从左到右逐步填充这个二维矩阵的过程

Earley算法：算法描述

- 初始化：
  - 对于规则集中，所有左端为初始符S的规则S→ α，把S→ · α加入到E(0,0)中
  - 如果B→ · A β在E(0,0)中，那么对于所有左端为符号A的规则A→ α，把A→ · α加入到E(0,0)中
- 循环执行以下步骤，直到分析成功或失败：
  - 如果A→α · x_j β在E(i,j-1)中，那么把A→α x_j · β加入到E(i,j)中
  - 如果A→ α · B β在E(i,j)中，那么对所有左端为符号B的规则B→γ，把B→ · γ加入到E(j,j)中
  - 如果B→γ在E(i,j)中，且在E(k,i-1)存在 A→ α · B β，那么把A→ α B · β加入到E(k,j)中

复习思考题

- “小王和小李的妹妹结婚了”，从语义上说，有哪些种可能的解释？
- 在以上介绍的分析算法中，对于句法歧义我们采用了两种消解策略：回溯和并行。请解释那种算法采用了回溯策略，那种算法采用了并行策略，并比较回溯和并行这两种策略的优缺点。
- 移进－归约算法是一种采用堆栈作为数据结构的自底向上分析算法。试构造一种采用堆栈作为数据结构的自顶向下分析算法。
- 试比较移进－归约算法和编译原理中LR(k)算法的异同。