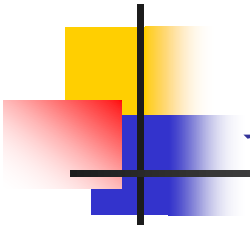
中文句法分析

刘秉权
智能技术与自然语言处理研究室
新技术楼612房间
liubq@hit.edu.cn



主要内容

- 1. 中文句法分析概述
- 2. 自底向上中文句法分析方法及示例
- 3. 自底向上中文句法分析器的实现
- 4. 总结

1. 中文句法分析概述

- 1.1 定义
- 1.2 短语结构分析
- 1.3 句法分析的意义、作用和学习目的
- 1.4 中文句法分析发展现状
- 1.5 句法结构的歧义消解
- 1.6 中文句法规则系统的构建
- 1.7 中文信息处理系统与编译系统的类比
- 1.8 中文句法分析与编译器中语法分析的类比
- 1.9 各种句法分析方法对比

1.1 定义

- 句法分析(syntactic parsing)的目的和任务就是对于给定的一个句子，识别句子的句法结构(syntactic structure)，生成一棵带有句法功能标记的短语结构树。
- 它反映了句子包含的各语言单位(包括词和短语)在结构上的组合关系和在句法功能上的聚合关系
- 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深入
- 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

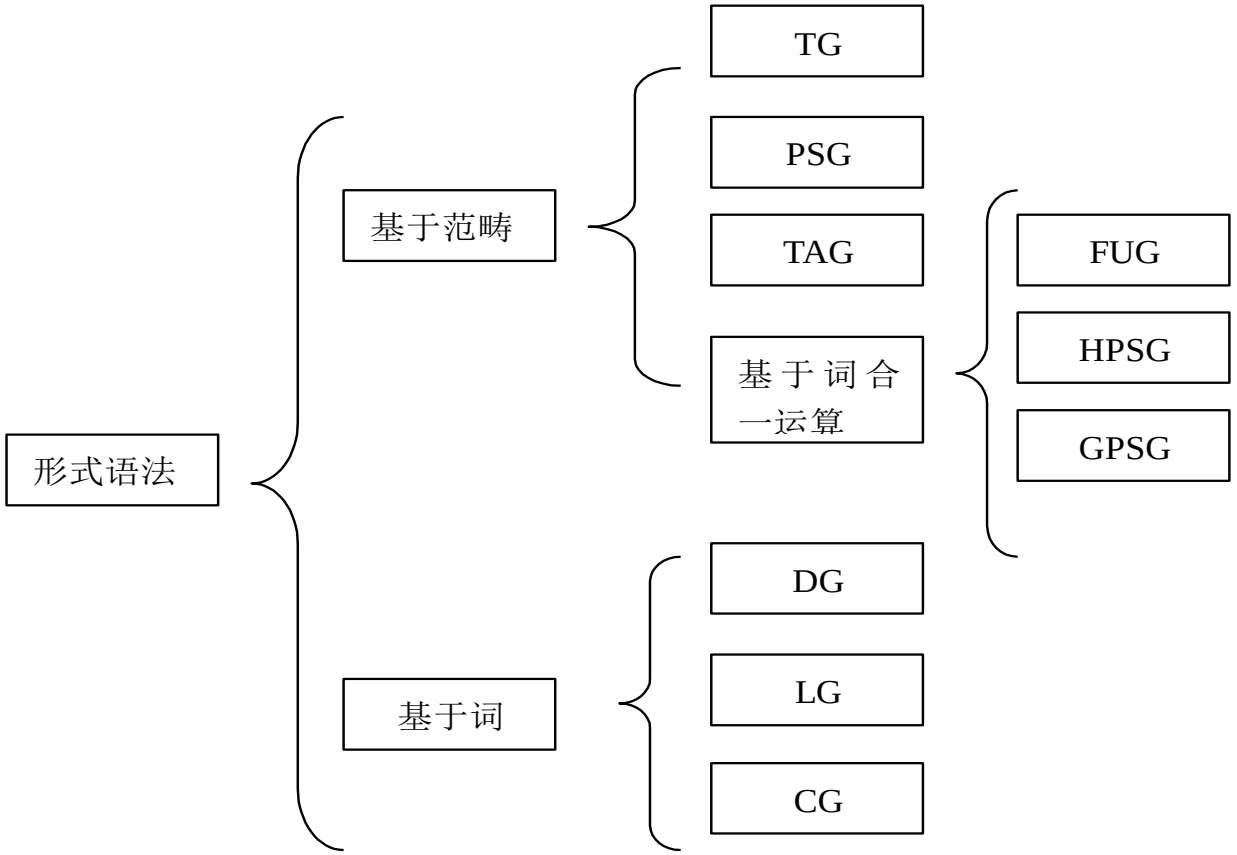
1.2 短语结构分析

- 常用的基于语言理解的语法理论
- 短语结构文法
- 四类文法分类体系
- 短语结构文法的特点
- 短语结构分析结果示例

常用的基于语言理解的语法理论

- 语言的形式化：用一定的范畴和作用在这些范畴上的规则来描述语言，范畴和规则可以通过数理符号来进行表示和推理
- 代表性的语法理论
  - 基于对语言知识表达的形式主义语法：**短语结构语法**（PSG）、**扩充转移网络**（ATN）、**支配约束理论**(GB)、**功能合一语法**(FUG)、**词汇功能语法**(LFG)、**中心词驱动的短语结构语法**(HPSG)、**广义短语结构语法**(GPSG)、**范畴语法**(CG)、**链接语法**(LG)、**树邻接语法**（TAG）
  - 基于对语言知识认知的功能主义语法：乔姆斯基**转换生成语法**（TG）、特尼埃尔的**依存语法**（配价语法）(DG)，菲尔默的**格语法**、韩礼德代表的**系统功能语法**、Langacker倡导的**认知语法**

按语法的基础分类



短语结构文法

- 短语结构文法(phrase structure grammar)以结构语言学的直接成分分析法为基础对语言进行定义,从而给予语言中的句子以有用结构的数学系统,又称Σ，F文法或乔姆斯基文法，是1957年美国语言学家N.乔姆斯基创立的语言转换生成理论的一部分。

短语结构文法

G=(T,V,P,S) :T 为终结符号集，V 为非终结符号集，是需要定义的语法范畴，P 为产生式集，S ∈ V,为文法的开始符号。

产生式规则型如：$\alpha \to \beta$，其中，$\alpha \in (T \cup V)^+$，$\beta \in (T \cup U)^*$，且$\alpha \neq \beta$，$(T \cup U)^*$表示由$(T \cup U)$ 中的符号所构成的全部符号串的集合，$(T \cup U)^+$表示$(T \cup U)^*$中除空符号以外的一切符号串的集合。

四类文法分类体系

- 对短语结构的形式体系增加某些约束，以提高区分歧义的能力
- 四类文法：0型文法、1型文法、2型文法和3型文法
- 语法的型号越高，对规则附加的限制就越多，语法的生成能力就越弱。他们之间的关系是：

0 型文法⊃ 1 型文法⊃ 2 型文法⊃ 3 型文法

无约束短语结构语法(0型文法)

若P中任一产生式都有一般的形式：

$$\alpha \to \beta \qquad \text{且}\alpha \in (T \cup V)^{+} \qquad \beta \in (T \cup U)^{*}$$

其中，对α，β 不加任何其它的限制，则称为0型文法或短语结构文法(简记为PSG)。

0 型文法所描述语言称之为 0 型语言。0 型文法是 Chomsky 文法体系中生成能力最强的一种形式。用它不足以描述自然语言，但对程序设计语言的描述而言又太一般化。所以需要一定的约束。

上下文有关语法（1型文法）

一个1型文法G中所有产生式具有如下的形式：

$$aAb \rightarrow a\beta b$$

其中，$a,b \in (T \cup U)^*$，$A \in V$，$\beta \in (T \cup U)^+$。之所以如此命名，是因为如果 A 前面出现符号 a，后面紧接着符号 b，则 A 可以重写为 β。它的重写规则依赖于上下文。

上下文无关文法（2型文法）

若在1型文法中所有的产生式有如下形式：

$$A \to \beta$$

其中，$A \in V$，$\beta \in (T \cup U)^+$，每条产生式的左侧必须是一个单独的非终结符。规则应用时不依赖于 A 所处的上下文，所以叫做上下文无关文法（CFG）。

正则语法（3型文法）

若在2型文法中仅含有如下的产生式：

$$A \to aB \quad A \to a$$

其中，A，B $\in$ V，a $\in$ T,则称为右线性文法。

若在2型文法中仅含有如下的产生式：

$$A \to Ba \quad A \to a$$

其中，A，B $\in$ V，a $\in$ T,则称为左线性文法。

一般地，把左线性文法和右线性文法统称为 3 型文法或者正则文法。一般的正则文法都可以用一个有向状态转移图（Finite-State Transition Diagram）来表示，所以它又叫做有限状态语法。它所表述的语言可以用有限自动机来识别，它一般用来描述程序设计语言的单词结构，不适宜用来描写自然语言。

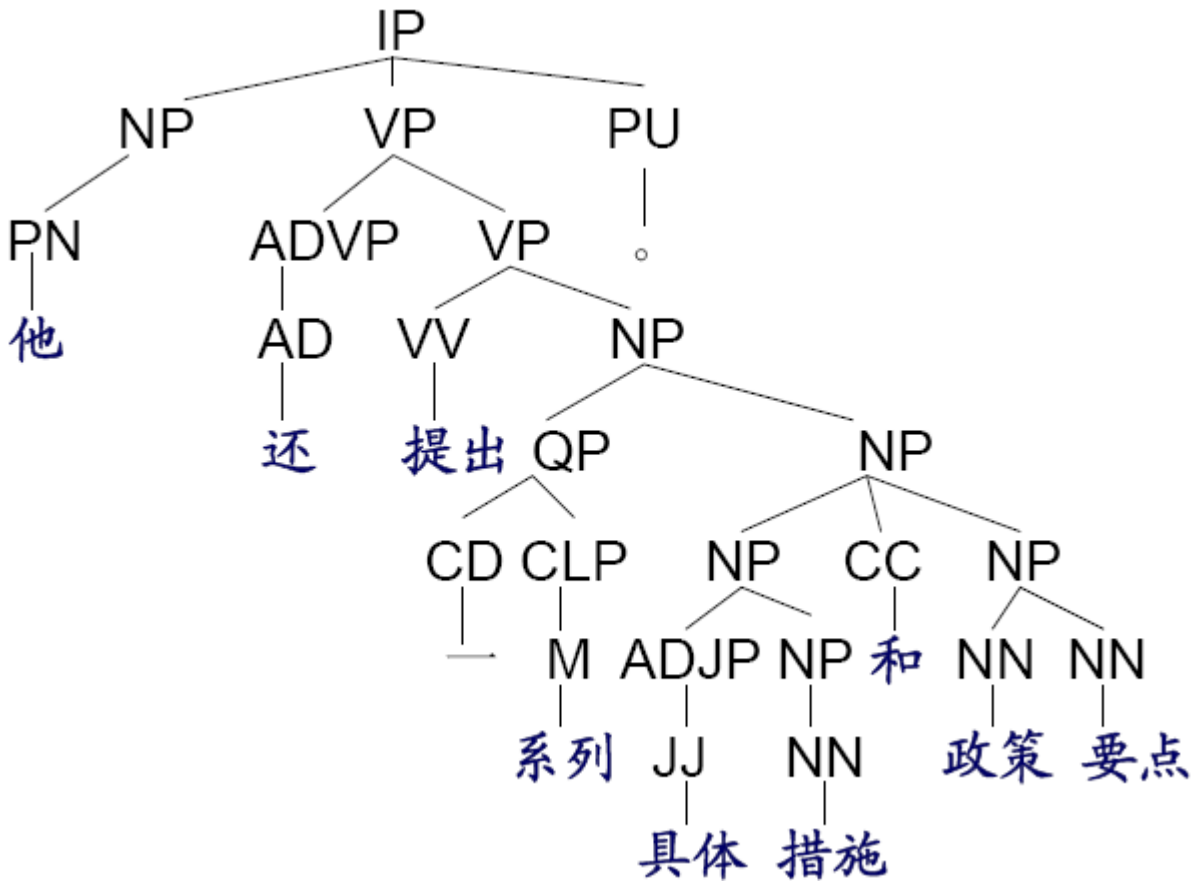
短语结构文法的特点

- 优点：可以方便地对一个句子进行结构描述
- 缺点：由于它没有引入语义的成分，在生成正确的句子的同时，也会生成错误的句子。如“牛吃草”和“草吃牛”在句法结构上都是一样的，短语结构文法没有办法加以区别。

短语结构分析结果示例

```
( IP (NP-SBJ (PN 他 ))
     (VP (ADVP ( AD 还 ))
          (VP (VV 提出 ))
               (NP-OBJ(QP (CD 一)
                          (CLP ( M 系列 )))
                     (NP (NP(ADJP ( JJ 具体)
                                (NP (NN 措施)))
                          ( CC 和)
                          ( NP ( NN 政策 )
                               ( NN 要点 ))))))
     ( PU 。 ))
```

树状表示


IP
NP
VP
PU
PN
ADVP
VP
。
他
AD
VV
NP
还
提出
QP
NP
CD
CLP
NP
CC
NP
一
M
ADJP
NP
和
NN
NN
系列
JJ
NN
政策
要点
具体
措施

1.3 句法分析的意义、作用和学习目的

- 自然语言处理（中文信息处理）的主要任务、重要环节和关键技术，是语言学、计算机科学、数学与认知学的交叉点
- 是很多重要文字处理应用的基础
  - 机器翻译
  - 文本摘要
  - 问答系统
  - 关系抽取与挖掘
- 学习目的
  - 了解中文句法分析的基本概念、基本思路和方法
  - 为深度中文信息处理和应用打下必要的基础

1.4 中文句法分析发展现状

- 短语结构文法技术相对成熟、稳步发展
- 中文处理跟踪、借鉴英文句法分析的方法
- 通过结合句法歧义消解和统计技术改进中文句法分析算法（统计句法分析）
- 部分句法分析、结合语义的句法分析、依存分析等是研究热点，并取得较好的应用效果

1.5 句法结构的歧义消解

- 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

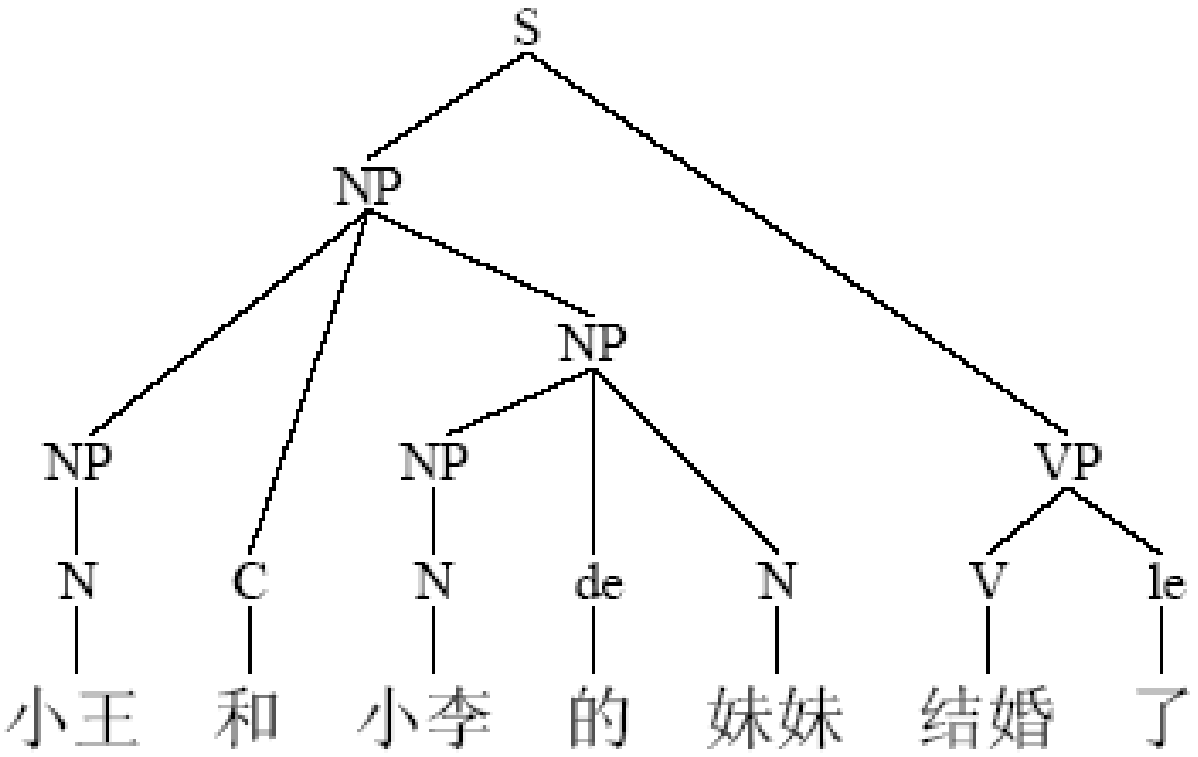
妹妹：N

结婚：V

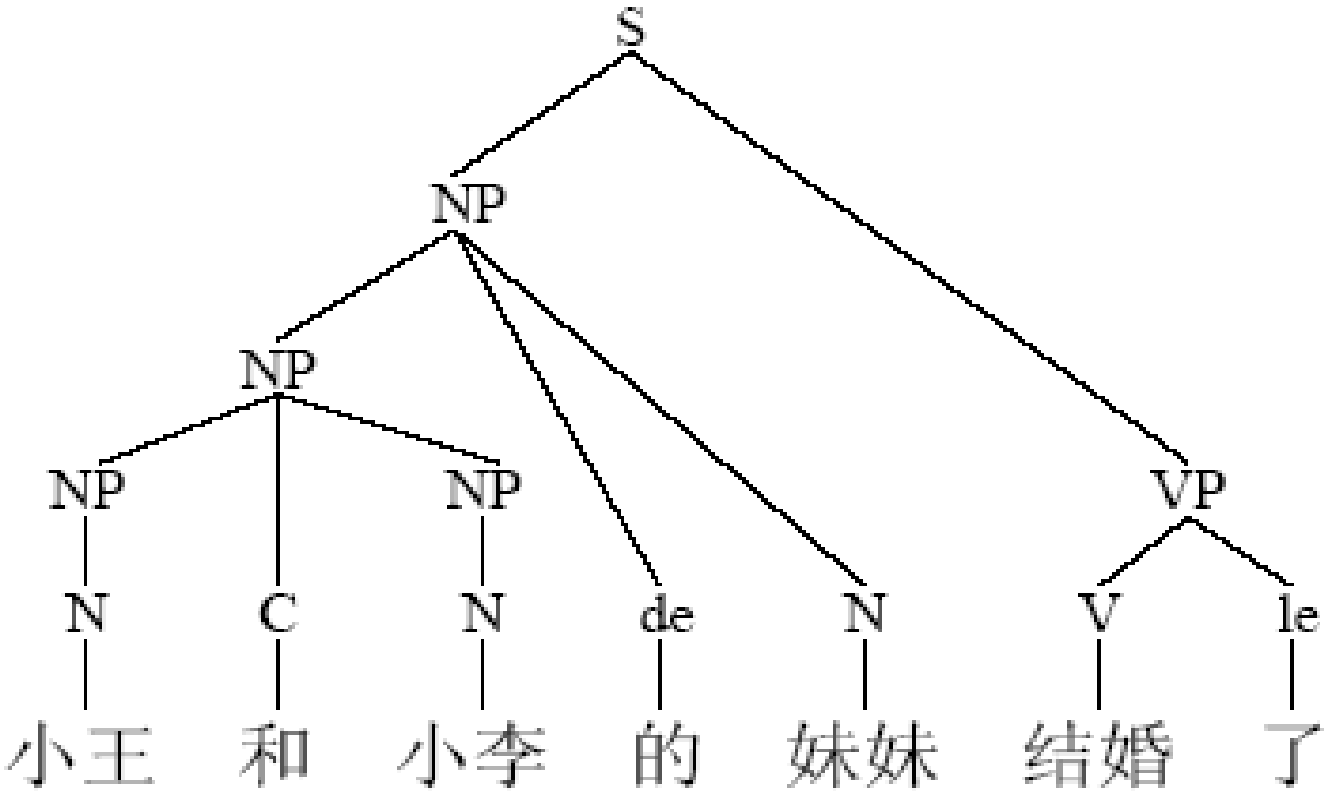
了：le

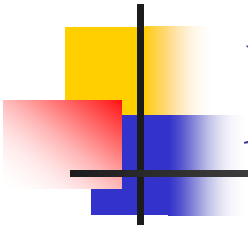
的：de

例子一分析结果之一



例子一分析结果之二



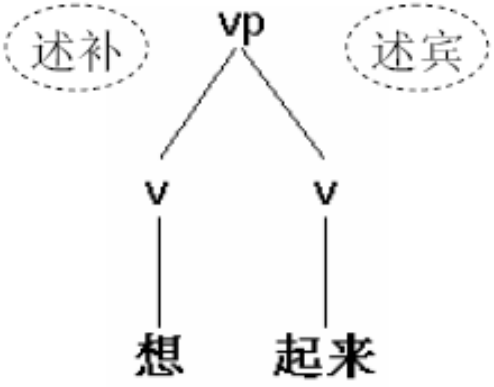


更多例子

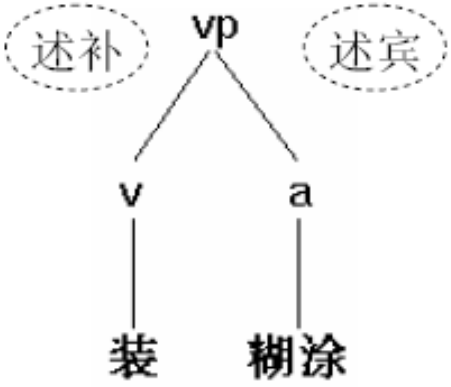
- 发现了敌人的哨兵
- 怀疑张三的老师
- 骑了三年的自行车
- 没有买票的人
- 支持罢课的学生
- 擦洗干净的桌子

更多例子

想起来



装糊涂



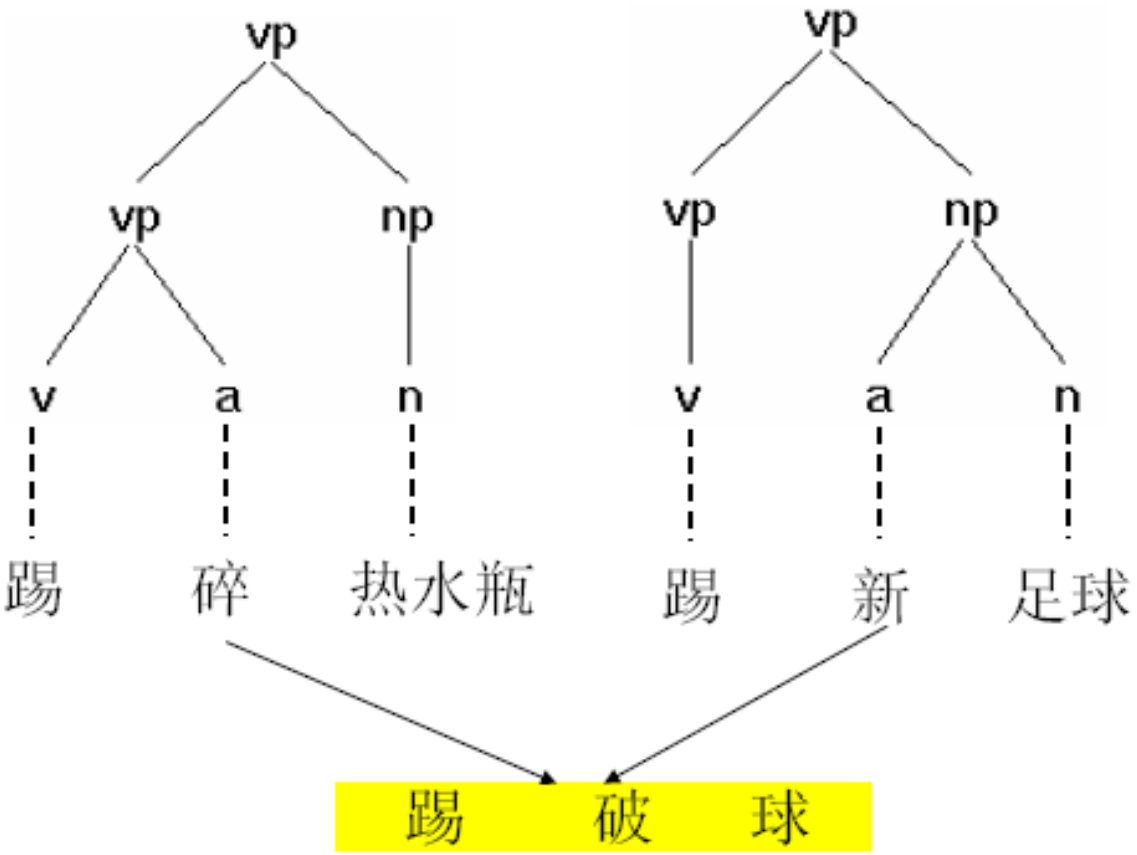
我终于想起来那天发生的事情了

奶奶躺了一整天，现在想起来了。

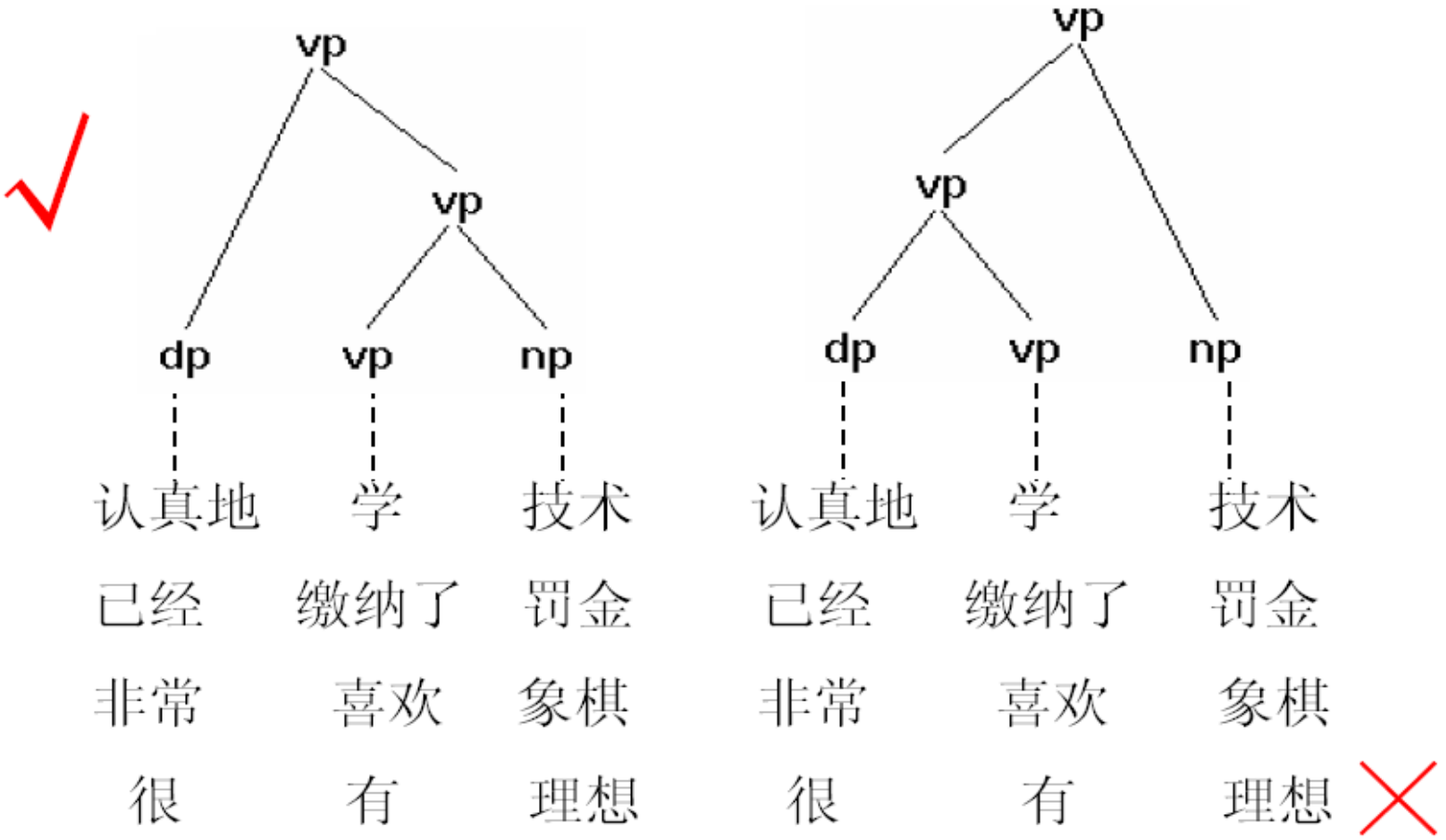
他就会装糊涂，其实他心理比谁都清楚

装了一上午家具，我都装糊涂了

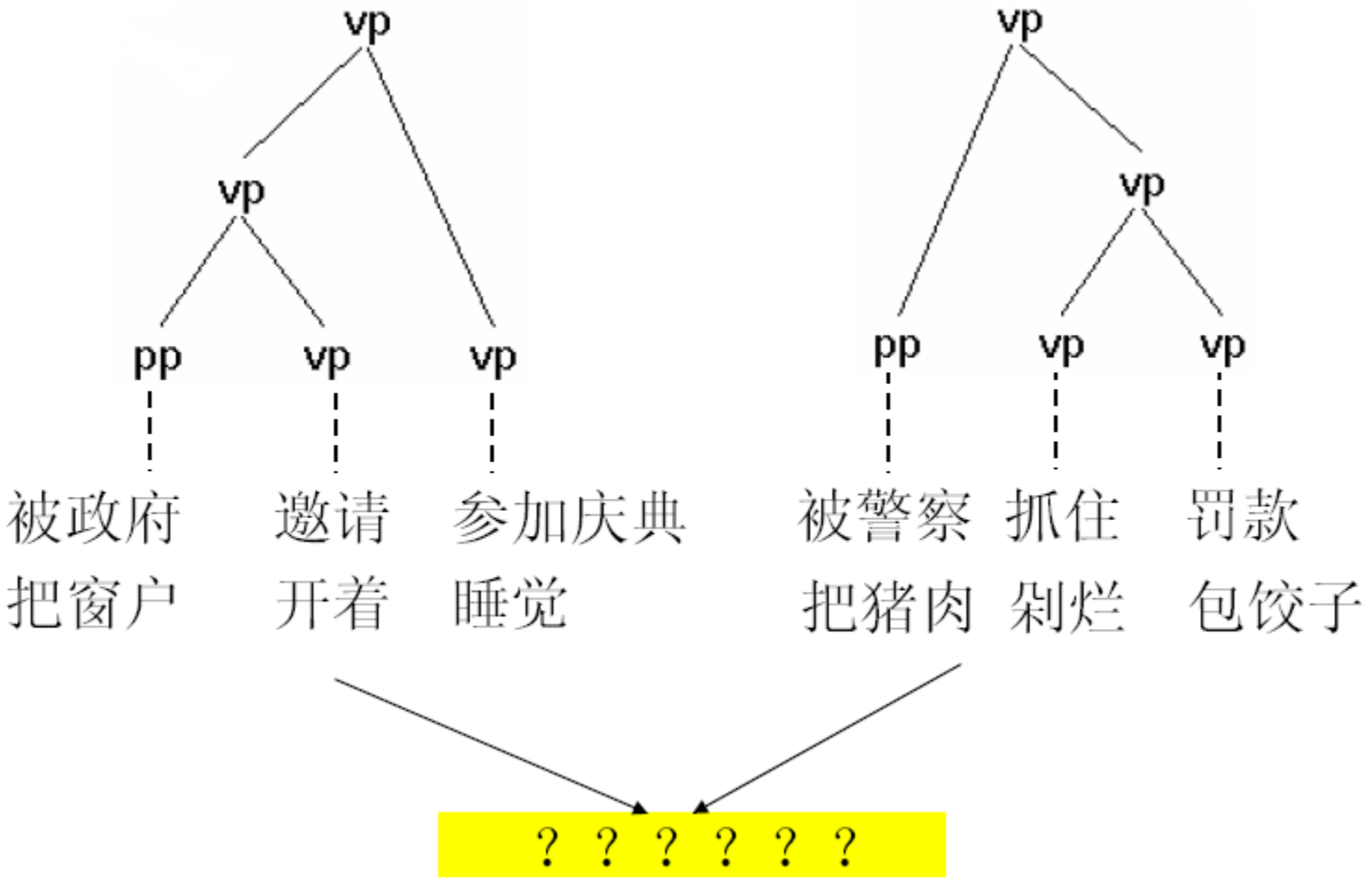
真歧义

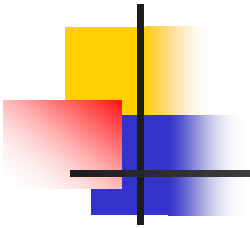


伪歧义


vp
vp
dp
vp
np
认真地
学
技术
已经
缴纳了
罚金
非常
喜欢
象棋
很
有
理想
vp
vp
dp
vp
np
认真地
学
技术
已经
缴纳了
罚金
非常
喜欢
象棋
很
有
理想

准歧义


vp
vp
pp
vp
vp
被政府
把窗户
邀请
开着
参加庆典
睡觉
vp
vp
pp
vp
vp
被警察
把猪肉
抓住
剁烂
罚款
包饺子
? ? ? ? ? ?



问题

- 歧义的消解对改善句法分析有什么影响？
- 如何消解中文的句法歧义？句法分析方法对消歧有什么影响？

1.6 中文句法规则系统的构建

- 文法体系的确定
- 词表的构建（终结符）
- 句法标注体系的确立（非终结符）
- 规则系统的构建（规则获取）
- 构建理论和方法的研究

北京大学基于功能分类方法的句法标注集

标记	名称	实例	标记	名称	实例
NP	名词短语	漂亮的帽子	SP	处所词短语	村子里
NBAR	名词准短语	工人们	MP	数量短语	这群
VP	动词短语	看电影	MBAR	数词准短语	一千三百
VBAR	动词准短语	学过	DJ	单句句型	她态度和蔼
AP	形容词短语	特别安静	ZJ	整句	你去不去？
ABAR	形容词准短语	高兴高兴	JQ	句群	救命啊！救命啊！
DP	副词短语	虚心地	DLC	独立成分	
PP	介词短语	在北京	YJ	直接引语	
BP	区别词短语	大型中型	FJ	复句	如果他去,我就去。
TP	时间词短语	战争时期			

1.7 中文信息处理系统与编译系统的类比

编译系统			中文信息处理系统（MT）		
过程	主要任务	方法	过程	主要任务	方法
词法分析	标识符、常数、运算符、关键字、界符识别	正则表达式方法	词法分析	词表查询、分词、实体识别，词性标注	规则与统计方法
语法分析	分析判别程序的语法结构是否符合语法规则	自顶向下、自底向上分析方法	句法分析	判定自然语言句子（子句）的正确性	继承编译系统方法；统计句法分析；语法词汇化等
语义分析	分析语法范畴的含义（变量是否定义、类型是否正确）	属性文法、语法制导	语义分析	语言单位的语义消歧、语义关系抽取等	统计方法、基于词典的方法、本体方法等
中间代码生成	生成独立于目标机器的中间代码	---	中间语言生成	生成人工定义的中间语言	---
目标代码生成	生成机器相关的可执行代码	---	目标语言生成	生成目标语言	---

1.8 中文句法分析与编译器中语法分析的类比

		人工语言语法分析	自然语言句法分析
相同点和联系		短语结构分析主要都采用上下文无关文法（CFG）， 自然语言分析继承了编译理论中的基本思想和方法	
区别	开放性	词表为小规模封闭集合； 有限规则集； 语法现象有限	词表为大规模开放集合； 大规模可扩充规则集合 语法现象复杂多样
	确定性	给出确定结果	给出句子符合语法的可能性
	主要问题	自顶向下方法中的回溯、消除左递归等问题，自底向上方法中可归约串的判定问题； 解决办法：对语法形式进行约束、规范，算符优先分析，数据结构的改造等	大规模应用中的效率问题，复杂语言歧义的消解问题； 解决：概率方法、剪枝、引入词义信息、语法词汇化
	具体方法	自顶向下：LL(1)分析法、递归下降分析方法、预测分析方法 自底向上：算符优先分析、LR 分析法	Chart 算法、CYK 算法、 Earley 算法、LR 算法、GLR 算法、带回溯的深度（广度）优先算法

1.9 各种句法分析方法对比

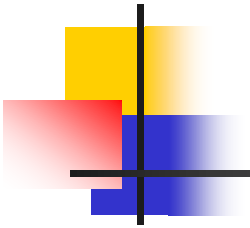
搜索策略 / 复杂度、方向	算法	
	自顶向下	自底向上
一般性的无方向分析	Chart 算法	CYK 算法 Chart 算法
一般性的有方向分析	带回溯的递归下降广度优先分析	Earley 算法 移近/规约算法 带回溯的深度优先分析
线性的有方向分析	LL（k）分析算法	LR 算法 有界上下文分析 优先分析
接近线性的一般性的有方向分析	（尚无研究结果）	GLR（Tomita）算法

复杂性说明

- CYK算法、Earley算法、Chart算法的时间复杂度通常认为是$O(n^3)$，其中Earley方法对于无歧义语法可以达到$O(n^2)$的复杂度
- 通过提供一个“适当构造的子串表(well formed substring table，WFST)”，每一个带回溯的分析方法都可以具有多项式的时间复杂度
- 上表中所有的确定性方法(包括LL(k)分析、优先分析、有界上下文分析、LR算法等)都是线性时间复杂度的O(n)
- GLR算法理论上最差情况下的时间复杂度为$O(C^n)$，然而在实际应用中，由于所用分析表的不确定性已尽可能减低到最小的程度，以及诸多选择性分析策略的使用，单纯从算法执行的角度来讲，在平均情况下可以达到$O(n^3)$，在最好情况下可以达到接近于线性的时间复杂度

2. 自底向上中文句法分析方法及示例

- 也称基于规约的方法。
- 这种方法是先逐步输入待分析字符串，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。



回溯

- 回溯法也称试探法。
- 基本思想：从问题的某一种状态（初始状态）出发，搜索从这种状态出发所能达到的所有“状态”；当一条路走到“尽头”的时候（不能再前进），再后退一步或若干步，从另一种可能“状态”出发，继续搜索，直到所有的“路径”（状态）都试探过。
- 这种不断“前进”、不断“回溯”寻找解的方法，就称作“回溯法”。

自底向上分析法一示例

- 例：我是县长派来的

- 其中Sφ、VPφ分别表示带空位的S和VP，把Sφ和VPφ分别简单看成两个独立的短语类型即可。

词典中的词条有：

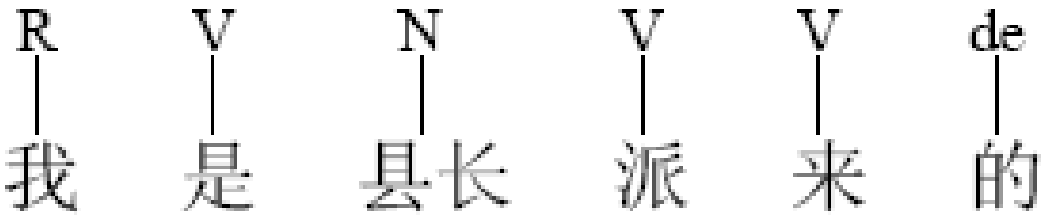
(1) R → 我
(2) N → 县长
(3) V → 是 | 派 | 来

所使用的规则为：

(1) S → NP VP
(2) NP → R
(3) NP → N
(4) NP → Sϕ 的
(5) VP → V NP
(6) Sϕ → NP VPϕ
(7) VPϕ → V V

自底向上分析法—示例1

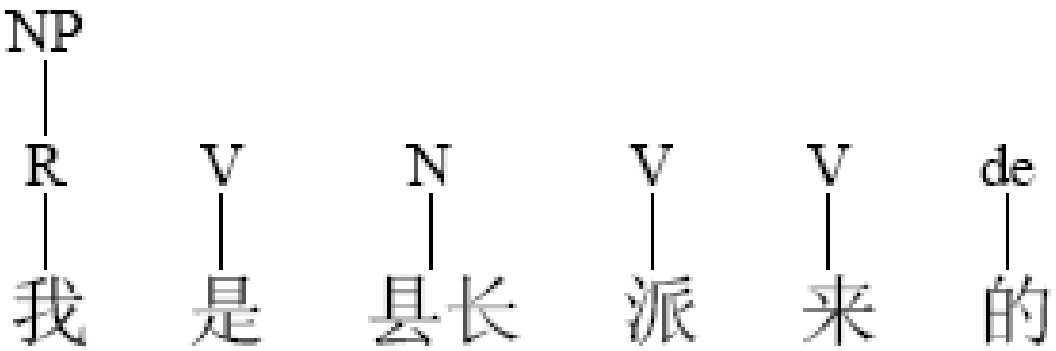
查词典



自底向上分析法—示例2

使用规则：

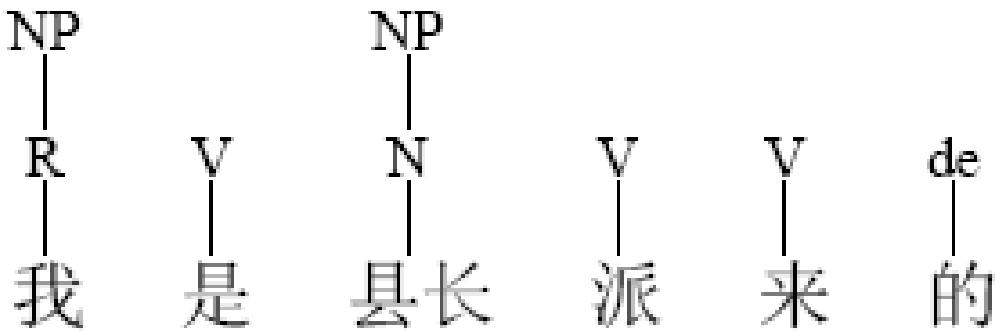
NP→R



自底向上分析法—示例3

使用规则：

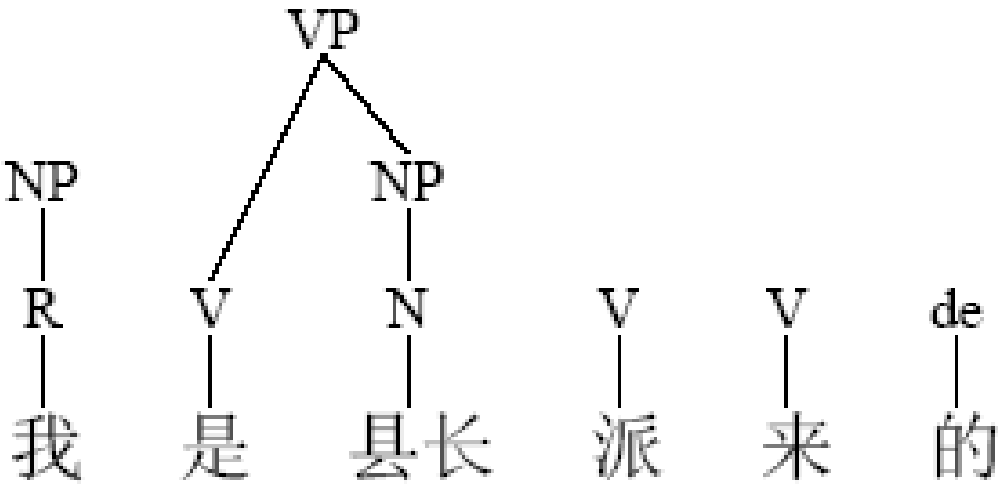
NP→N



自底向上分析法—示例4

使用规则：

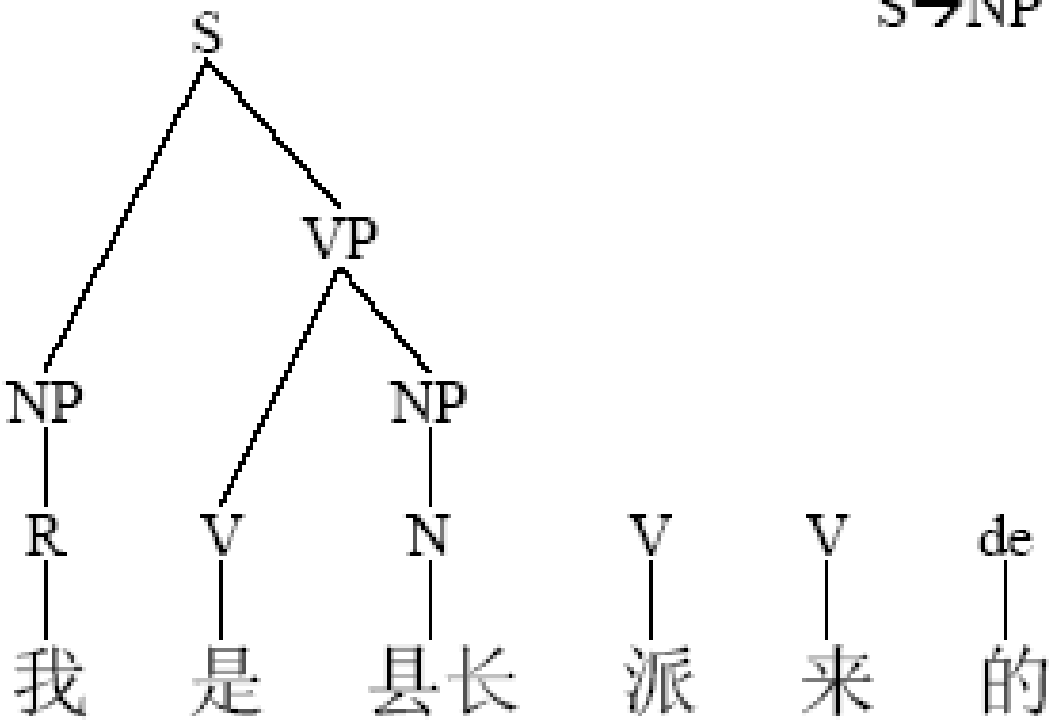
VP→V NP



自底向上分析法—示例5

使用规则：

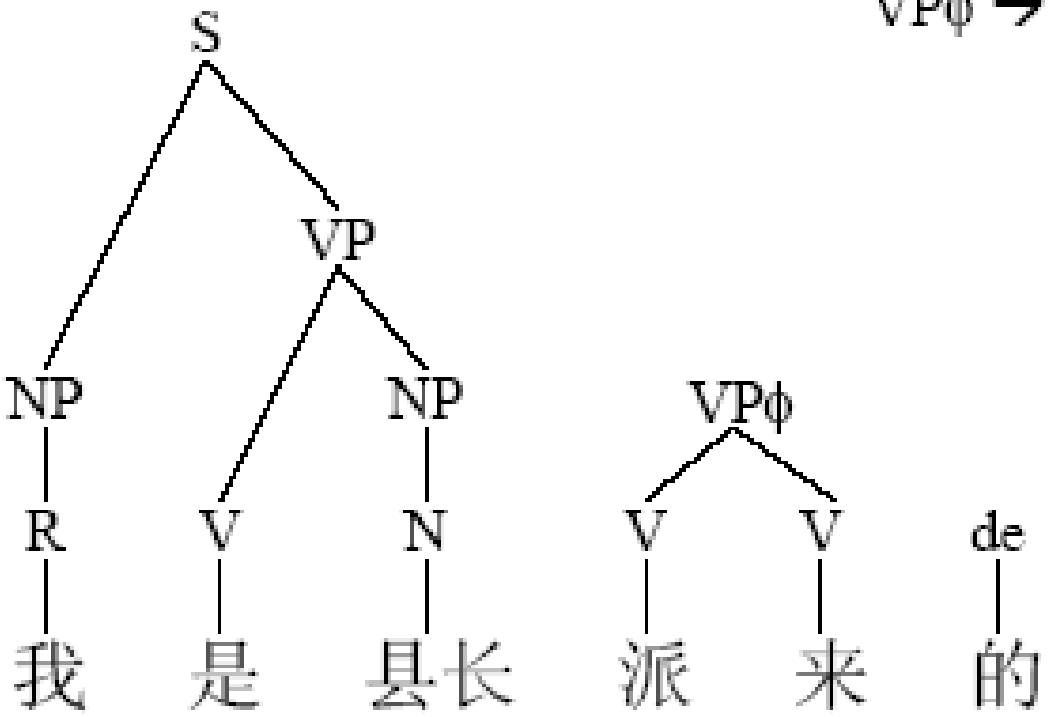
S→NP VP



自底向上分析法—示例6

使用规则：

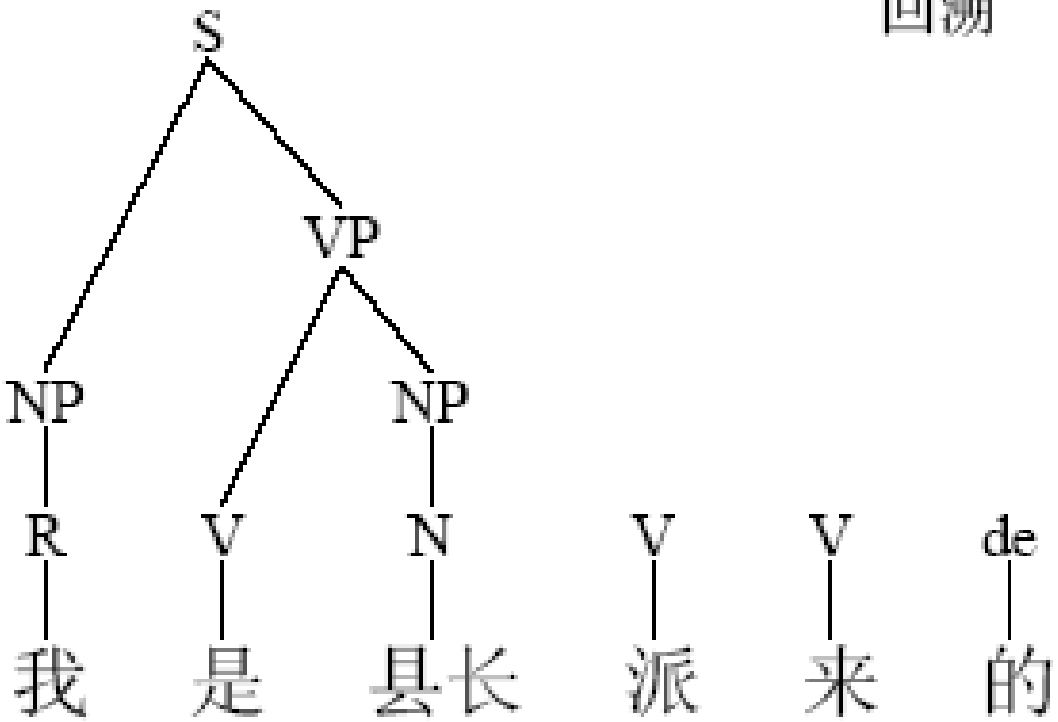
VPϕ →V V



自底向上分析法—示例7

无规则可用

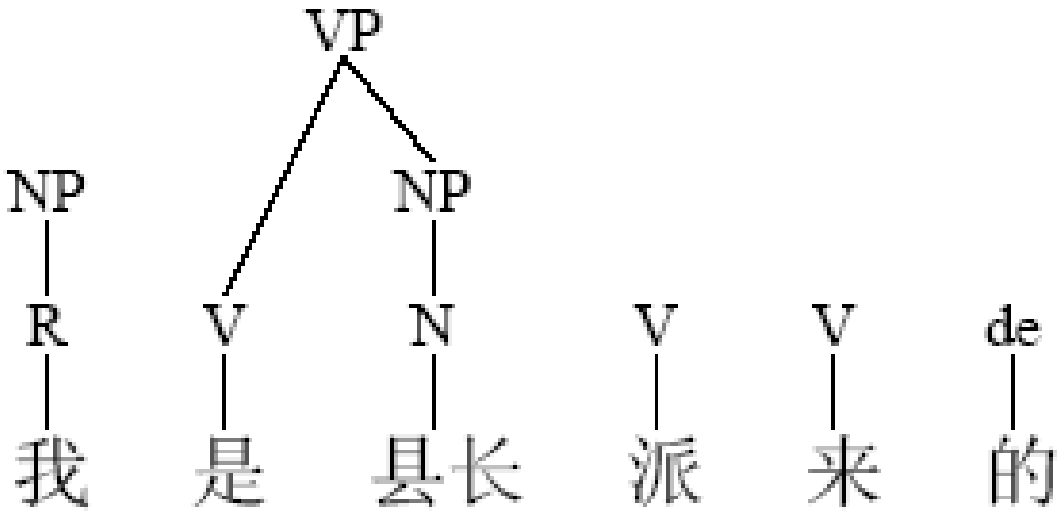
回溯



自底向上分析法—示例8

无规则可用，

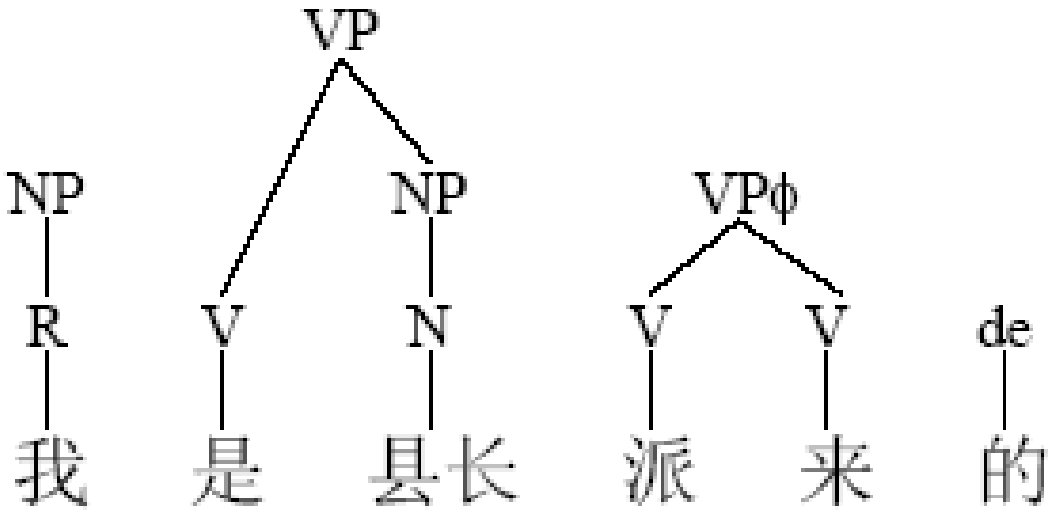
回溯



自底向上分析法—示例9

使用规则：

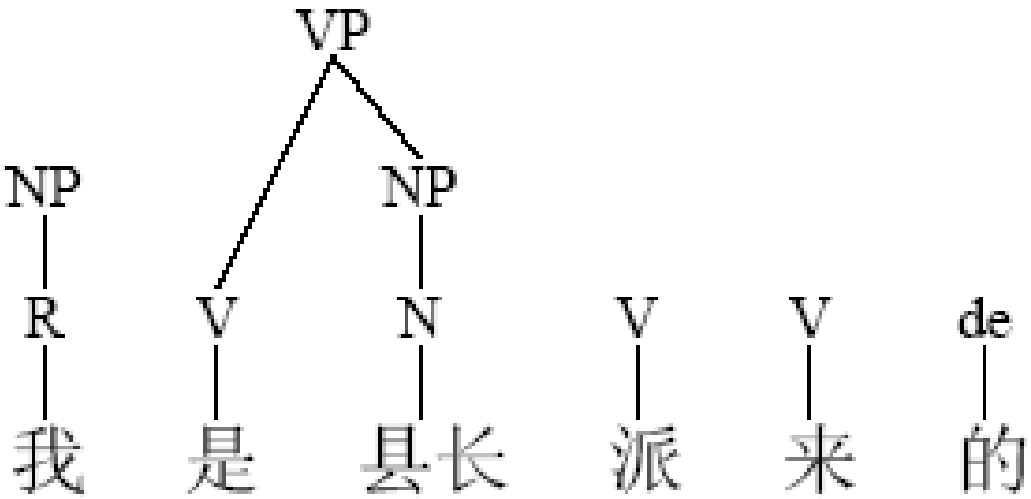
VPϕ →V V



自底向上分析法—示例10

无规则可用，

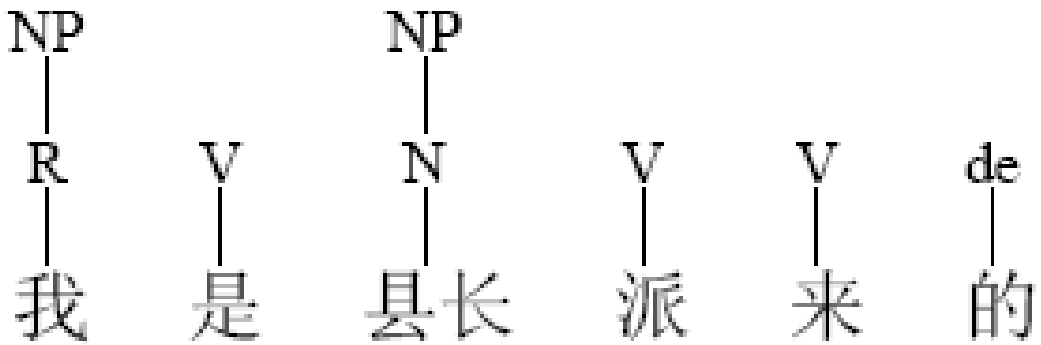
回溯



自底向上分析法—示例11

无规则可用，

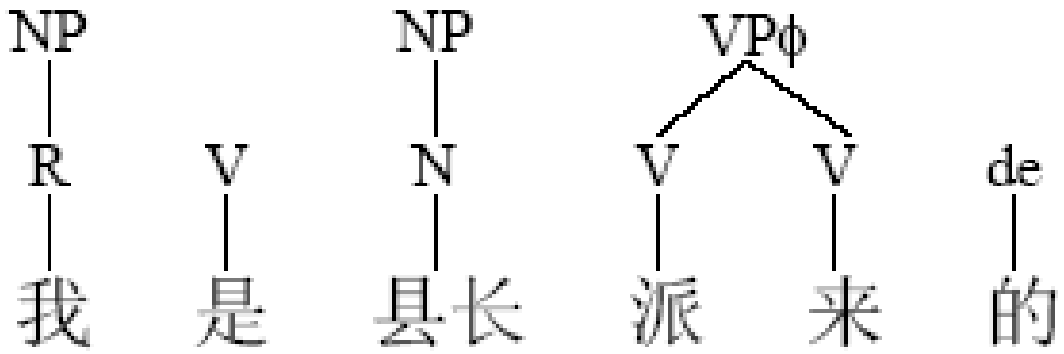
回溯



自底向上分析法—示例12

使用规则：

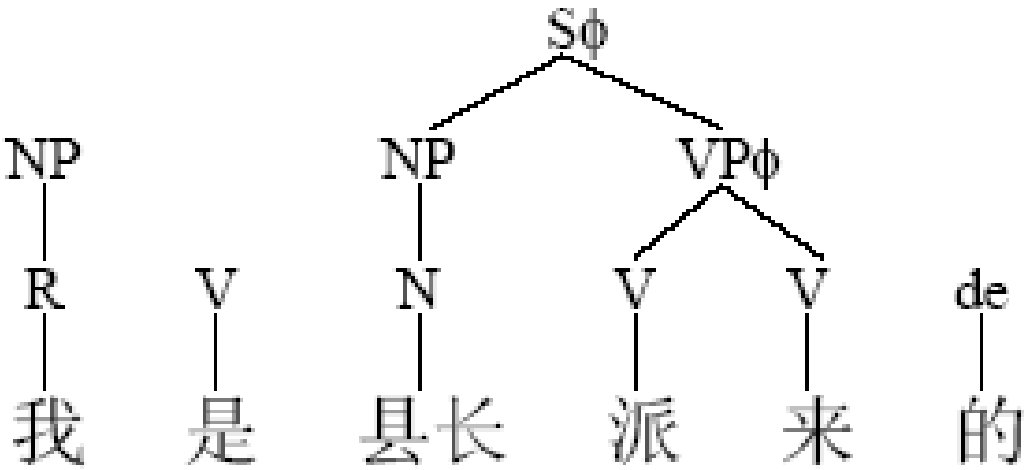
VPϕ →V V



自底向上分析法—示例13

使用规则：

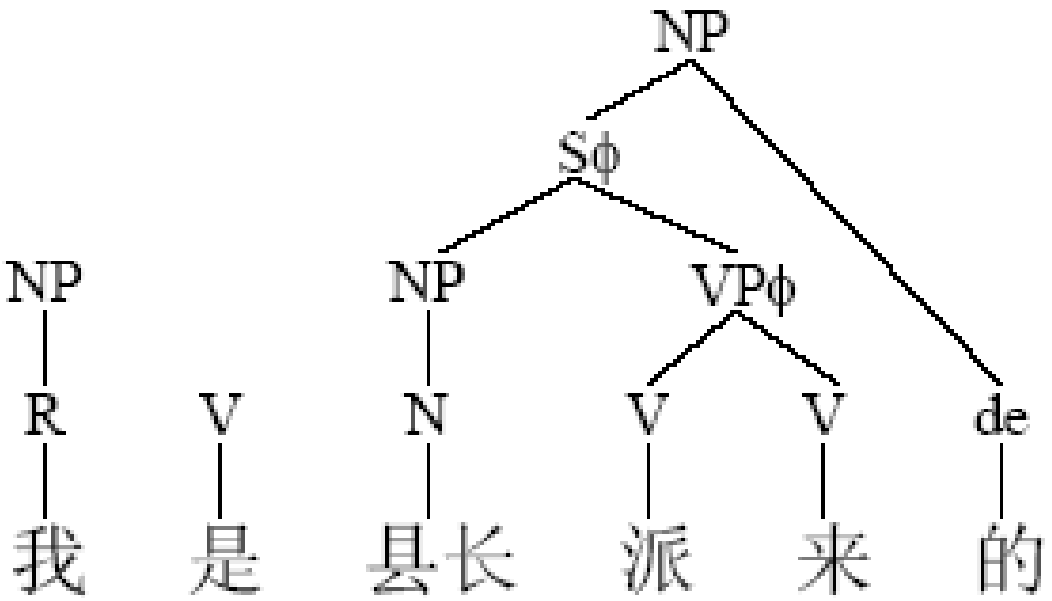
Sϕ →NP VPϕ



自底向上分析法—示例14

使用规则：

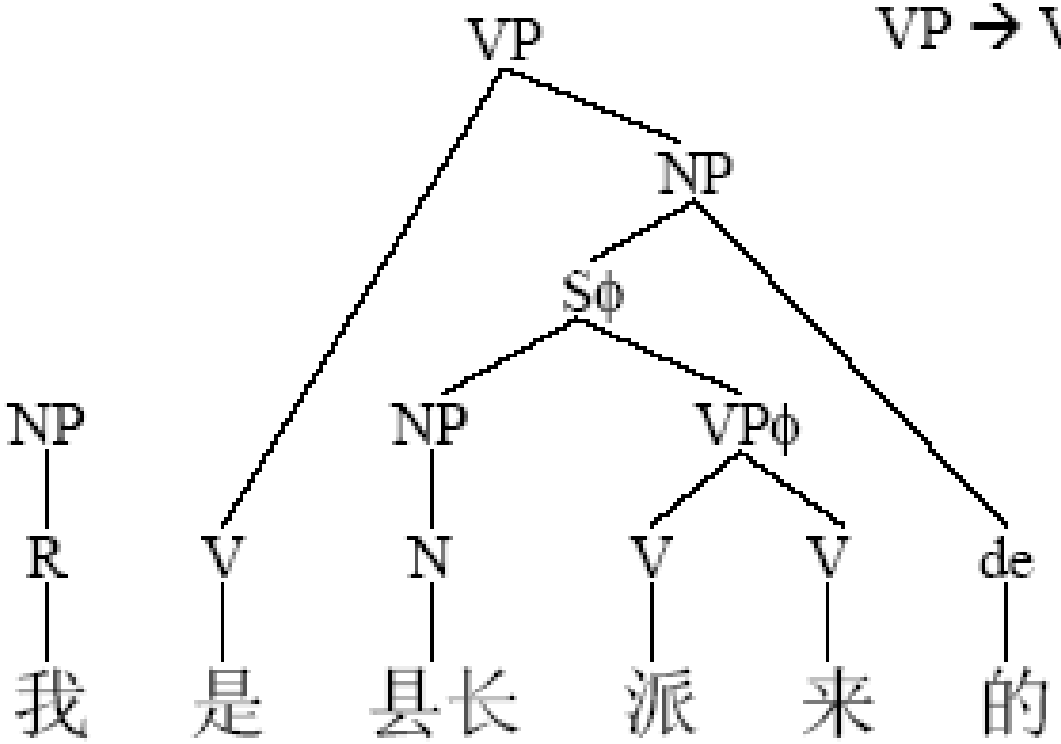
NP → Sϕ de



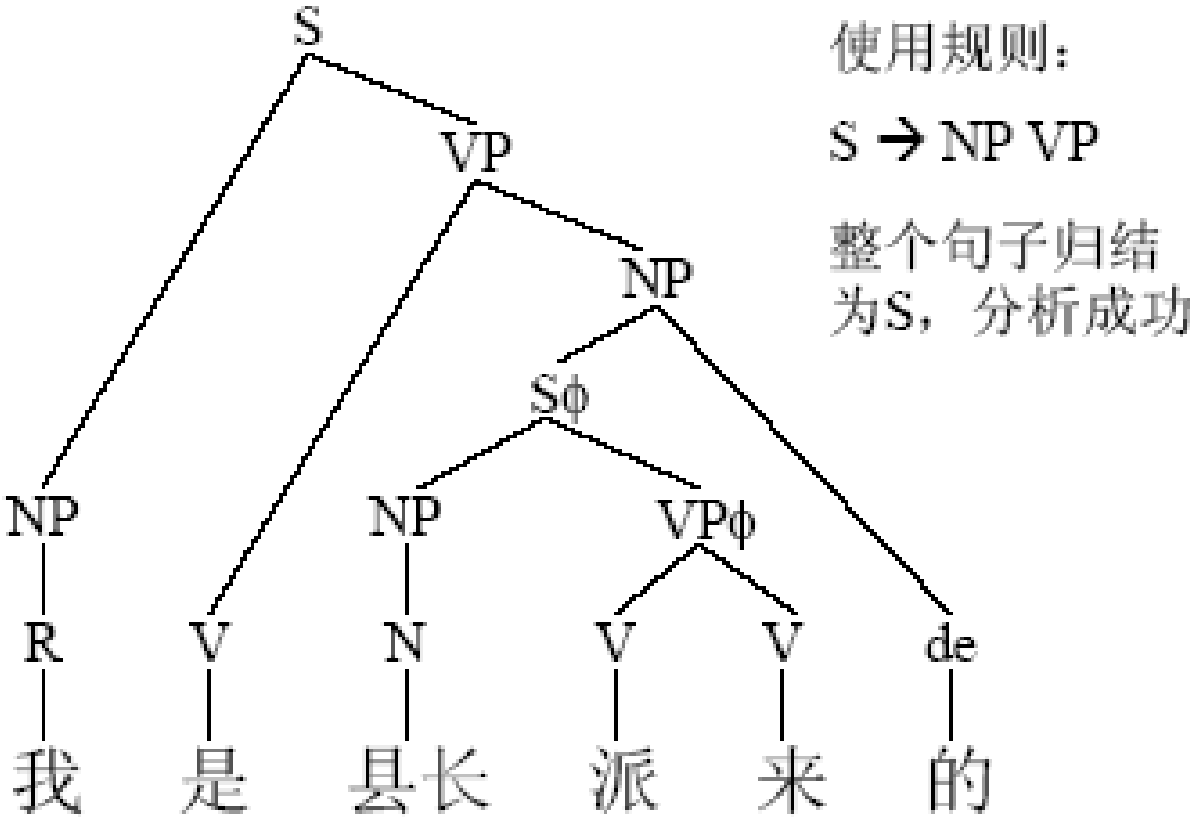
自底向上分析法—示例15

使用规则：

VP → V NP



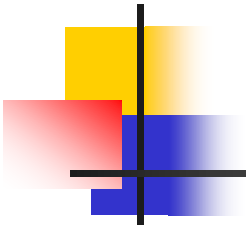
自底向上分析法—示例16



使用规则：

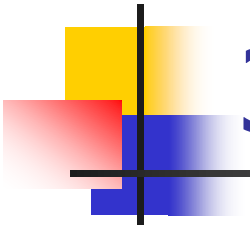
S → NP VP

整个句子归结为S，分析成功



问题

- 用什么方法实现以上句法分析过程？
- 解决回溯问题有哪些策略？
- 能否避免回溯问题的发生？



3. 自底向上句法分析算法

- 移进－归约算法
- CYK算法
- Chart算法

3.1 移进－归约算法：概述

- 移进－归约算法：Shift-Reduce Algorithm
- 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	R→ 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
- 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

- 冲突解决策略：
  - 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	(10) VP→V NP
10	# NP (7) VP (10)	派 来 的	归约	(3) S→NP VP
11	# S (3)	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	
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	

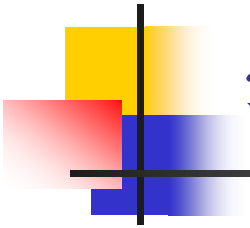
说明：为简洁起见，这个例子中没有给出堆栈中被替换结点信息，但必须注意到这些信息是必需的，否则归约操作将无法回溯。

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



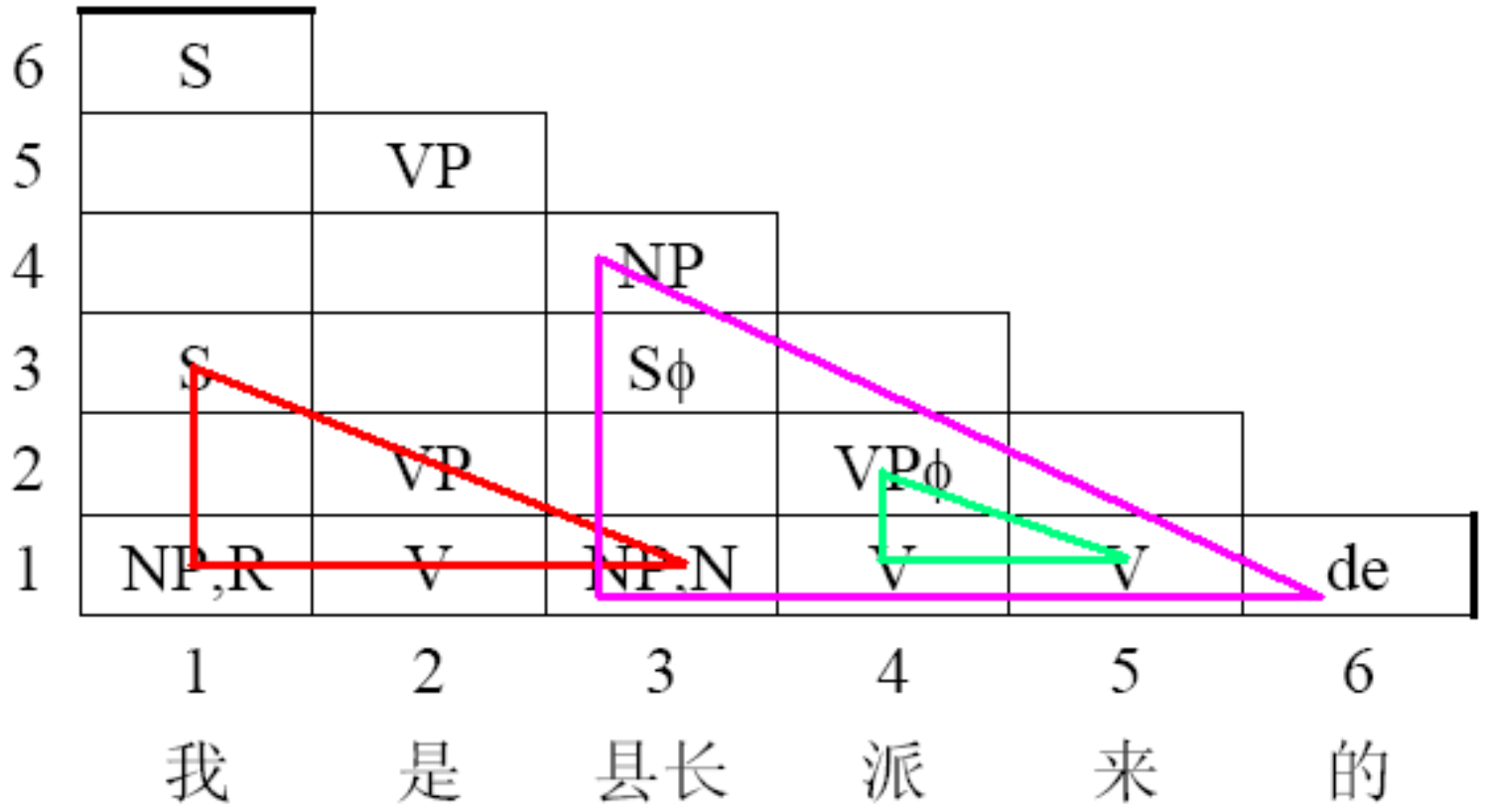
尝试

- 请验证53步移进-规约过程是否准确。
- 试着给出被替换结点的具体信息。
- 思考或尝试实现给出的各项改进措施。
- 能否避免回溯问题？

3.2 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



CYK算法—数据结构2

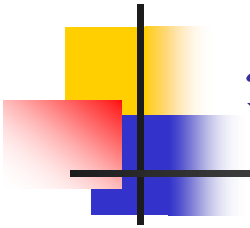
- 一个二维矩阵：{ P(i , j) }
  - 每一个元素P(i , j)对应于输入句子中某一个跨度（Span）上所有可能形成的短语的非终结符的集合
  - 横坐标i：该跨度左侧第一个词的位置
  - 纵坐标j：为该跨度包含的词数
- 上图中P(3,1)={NP,N}表示“县长”可以归约成N和NP，P(3,3)={Sφ}表示“县长派来”可以规约成Sφ

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

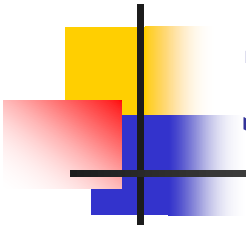
CYK算法：特点

- 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，没有回溯带来的冗余操作；
- 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，难以采用启发式策略进行改进。



尝试

- 用多进程或多线程程序实现该算法。



3.3 Chart(线图)算法

- 文法实例
- Chart数据结构
- 活跃边与非活跃边
- 日程表(Agenda)
- Chart算法的基本流程
- 日程表的初始化策略
- 基本策略
- 规则调用策略
- 日程表组织策略
- 其他细节处理
- 评价

文法实例

- 例：我是县长派来的

- 其中Sφ、VPφ分别表示带空位的S和VP，把Sφ和VPφ分别简单看成两个独立的短语类型即可。

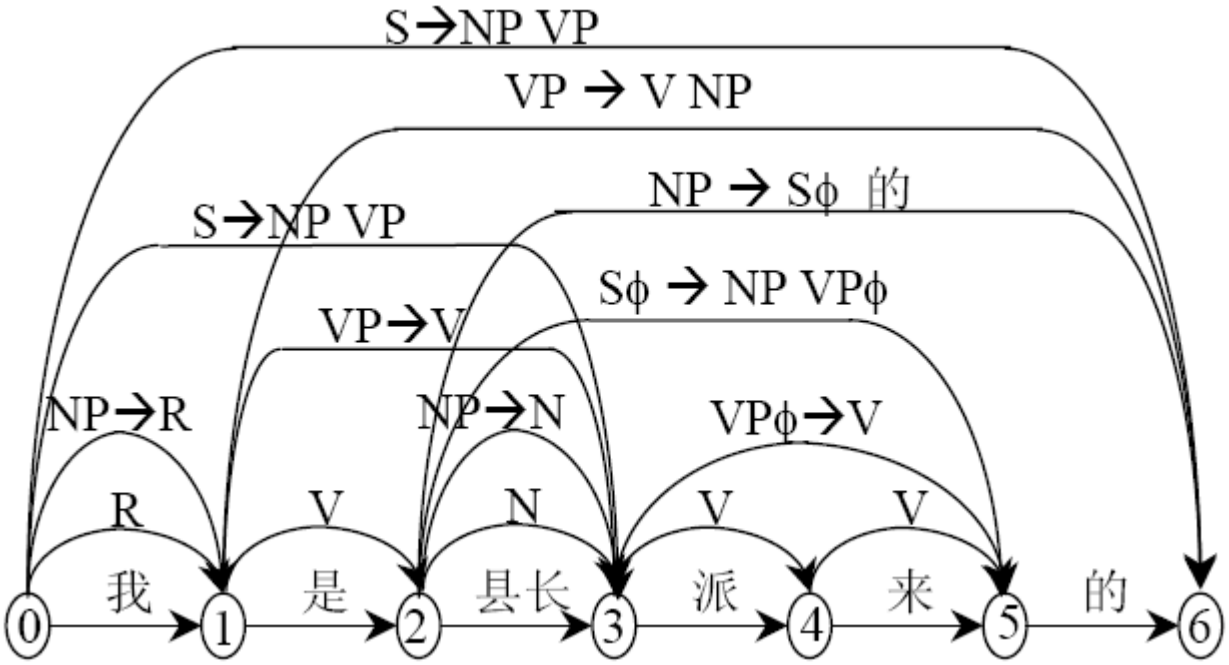
词典中的词条有：

(1) R → 我
(2) N → 县长
(3) V → 是 | 派 | 来

所使用的规则为：

(1) S → NP VP
(2) NP → R
(3) NP → N
(4) NP → Sϕ 的
(5) VP → V NP
(6) Sϕ → NP VPϕ
(7) VPϕ → V V

Chart数据结构



- 线图是把词与词之间的间隔作为结点，把词和短语当作连接结点的边。不仅标出了每条边的标记，还标出了产生该边的规则。

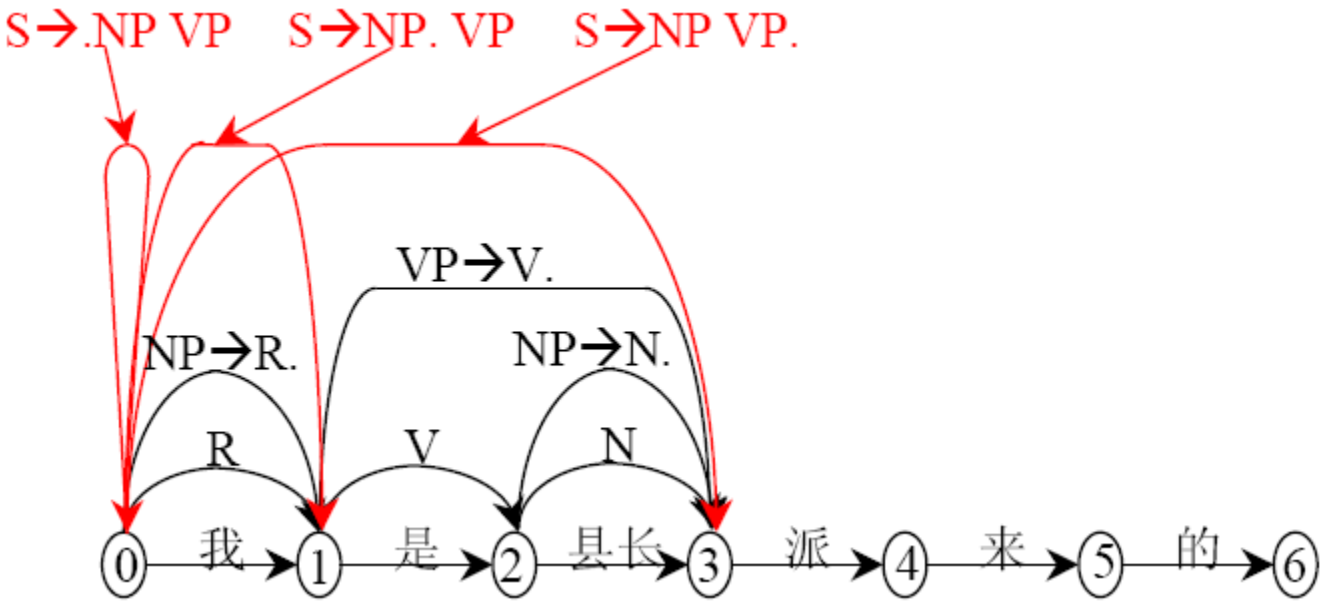
活跃边与非活跃边

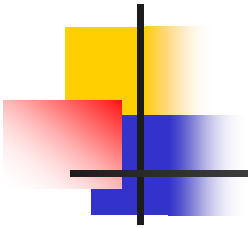
- 定义边：<i,j,A → BC>
- 非活跃边：记录一条规则被完全匹配的边
- 活跃边：记录一条规则部分被匹配的边
- 例：

记录方式	边状态	匹配程度	起点	终点	对应词串
<0,0, S→.NP VP>	活跃	S → .NP VP	0	0	
<0,1, S→NP. VP>	活跃	S → NP. VP	0	1	我
<0,3, S→NP VP.>	非活跃	S → NP VP.	0	3	我是县长

活跃边与非活跃边

- “匹配程度”用规则中加入句点来表示，其中句点的位置表示规则已经匹配成功的位置（从左边开始）





日程表(Agenda)

- Chart分析的过程就是一个不断产生新的边的过程。
- 但是每一条新产生的边并不能立即加入到Chart中，而是要放到日程表中。
- 日程表（Agenda）是一个边的集合，用于存放已经产生，但是还没有加入到Chart中的边。
- 日程表中边的排序和存取方式，是Chart算法执行策略的一个重要方面。
- Chart算法就是一个由日程表驱动的不断循环的过程

Chart算法的基本流程

- (1) 按照初始化策略初始化日程表（Agenda）；
- (2) 如果日程表（Agenda）为空，那么分析失败；
- (3) 每次按照日程表组织策略从日程表（Agenda）中取出一条边；
- (4) 如果取出的边是一条非活跃边，而且覆盖整个句子，那么返回成功；
- (5) 将取出的边加入到Chart中，执行基本策略和规则调用策略，将产生的新边又加入到日程表（Agenda）中；
- (6) 返回第(2)步。

日程表的初始化策略

- 自底向上分析：
  - (2) 将所有单词（含词性）边加入到日程表（Agenda）中。
- *自顶向下分析：*
  - *(1) 将所有单词（含词性）边加入到日程表（Agenda）中；*
  - *(2) 对于所有形式为：S→W的规则，产生一条形式为<0, 0, S →.W>的边，并加入到日程表（Agenda）中。*

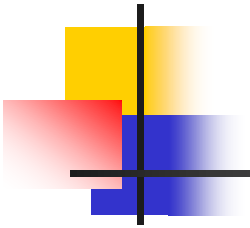
基本策略

- 如果新加入一条活跃边形式为：<i, j, A → W1. B W2>
  - 那么对于Chart中所有形式为：<j, k, B → W3>的非活跃边，生成一条形式为<i, k, A → W1 B. W2>的新边，并加入到日程表（Agenda）中；
- 如果新加入一条非活跃边形式为：<j, k, B → W3>
  - 那么对于Chart中所有形式为：<i, j'(k>j'>=j), A → W1. B W2>的活跃边，生成一条形式为<i, k, A → W1 B. W2>的新边，并加入到日程表（Agenda）。

其中A、B为非终结符，W1、W2、W3为终结符和非终结符组成的串，其中W1、W2允许为空，W3不允许为空。

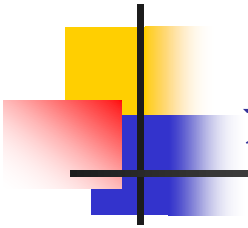
规则调用策略

- 自底向上分析：
  - 如果要加入一条形式为<i, j, C → W1.>的边到Chart中， 那么对于所有形式为B → C W2的规则，产生一条形式为<i, i, B → .C W2>的边加入到日程表（Agenda）中。
- *自顶向下分析：*
  - *如果要加入一条形式为<i, j, C → W1.B W2>的边到Chart中，那么对于所有形式为B → W的规则，产生一条形式为<j, j, B → .W>的边，并加入到日程表（Agenda）中。*



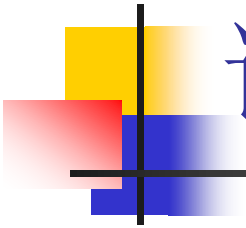
日程表组织策略

- 深度优先：
  - 将日程表设计成堆栈形式，每次从日程表中取出最后加入的结点；
- 广度优先：
  - 将日程表设计成队列形式，每次从日程表中取出最早加入的结点。



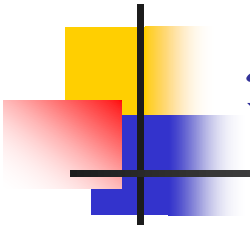
其他细节处理

- (1) 考虑到可能通过多种途径生成一条完全相同的边，所以每次从日程表（Agenda）中取出一条新边加入Chart时，要先检查一下Chart中是否已经有相同的边，如果有，那么删除这条边，直接进入下一个循环；
- (2) 为了生成最后的句法结构树，每一条边中还应该记录其子句法成分所对应的边。



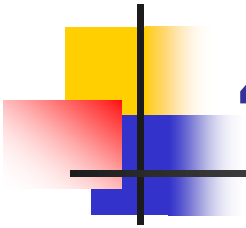
评价

- Chart分析算法是一种非常灵活的分析算法，通过修改分析过程中的一些具体策略，Chart分析算法可以模拟很多种其他句法分析算法。
- 很多研究工作基于此展开
- 缺点
  - 时间复杂度为 Kn^3
  - 需要高质量的规则，分析结果与规则质量密切相关;
  - 难以区分歧义结构



尝试

- 实现一个基于Chart算法的中文句法分析器
- 模拟在不同初始化策略、Agenda组织策略下分析器的分析过程
- 模拟不同初始化顺序、不同规则产生顺序下的分析结果，并分析其原因
- 分析Chart算法的时间复杂性



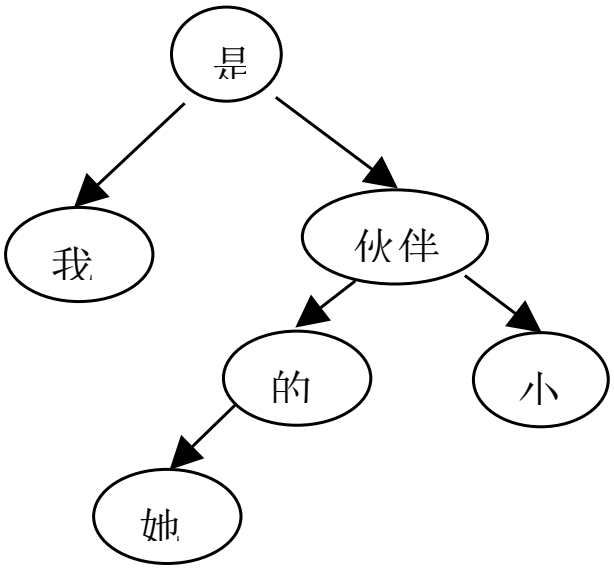
4. 总结

- 了解了中文句法分析的概况
- 通过示例，了解了自底向上方法的基本思想
- 熟悉了三种自底向上中文句法分析的基本算法及其主要特点
- 这些算法本身无法解决复杂的语言歧义问题
- Chart算法经过了深入研究，很多改进工作在其上完成，是后续内容的基础

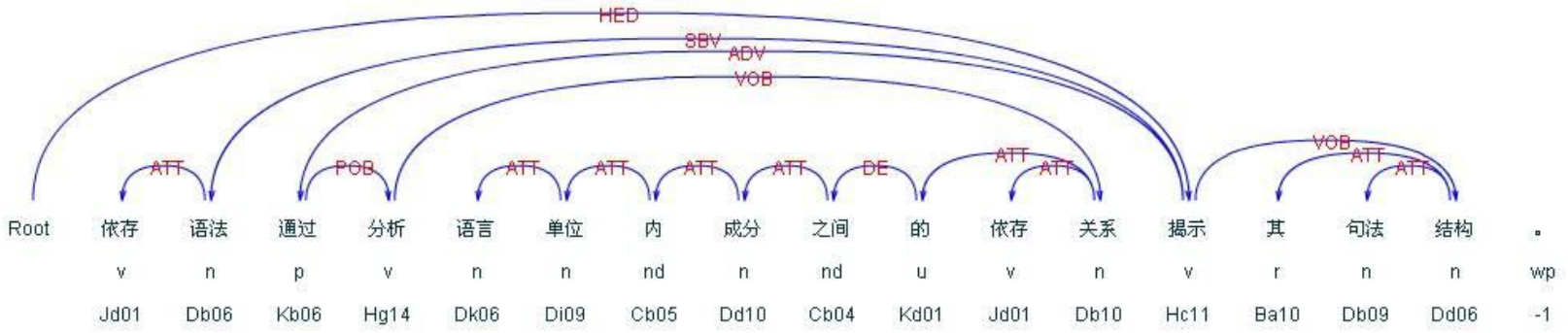
依存语法(法国语言学家Tesniere . L1959年提出)

- 不管是哪个学派的语法学说，都不得不正视句子中词与词之间的句法、语义联系
- 反映动词对名词性成分的支配能力的概念，是哪一种语法理论也回避不了的
- 依存语法表述词与词之间一种最基本的联系
- 在一般应用的时候，都是以动词作为一个句子的中心，支配其他的成分。所有受支配成分都以某种依存关系从属于支配成分

句子的依存语法表示



另一个示例


HED
SBV
ADV
VOB
ATT
POB
ATT
ATT
ATT
ATT
DE
ATT
ATT
VOB
ATT
ATT
Root 依存 语法 通过 分析 语言 单位 内 成分 之间 的 依存 关系 揭示 其 句法 结构 。
v n p v n n nd n nd u v n v r n n wp
Jd01 Db06 Kb06 Hg14 Dk06 Di09 Cb05 Dd10 Cb04 Kd01 Jd01 Db10 Hc11 Ba10 Db09 Dd06 -1

依存关系的四大公理(Robinson J.J, 1970年)

- 一个句子中只有一个成分是独立的，不受任何成分支配
- 其他成分直接依存于另外的某一个成分
- 任何一个成分都不能直接依存于两个或两个以上的其他成分
- 如果A成分直接依存于B成分，而C成分在句中位于A成分和B成分之间的话，那么，C或者直接依存于A，或者直接依存于B，或者直接依存于A和B之间的某一成分

依存语法的主要特点

- 表示方法简洁，易懂
- 依存语法生成的句法树不含非终结符，一个具有N个词的句子，生成的依存句法树只有N个结点和(N-1)条边。而利用短语结构语法得到的句法树由于含有非终结符，结点数大大超过N,大大超过(N-1)条边
- 注重词与词的语义的依存关系,而不是在句中出现的顺序关系

概率上下文无关文法 (Probabilistic/ Stochastic Context-Free Grammar, PCFG/SCFG)

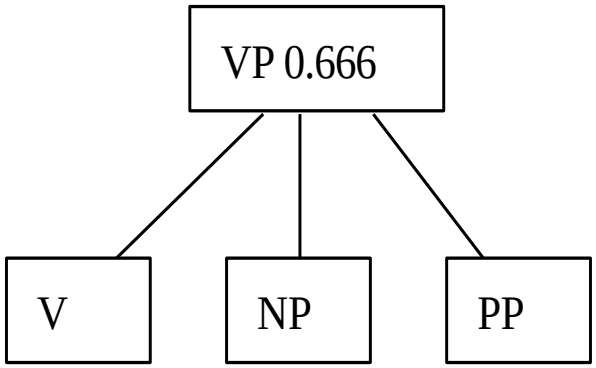
一个四元组 $G=\langle \Sigma_N^+, \Sigma_T^+, S, R\rangle$，

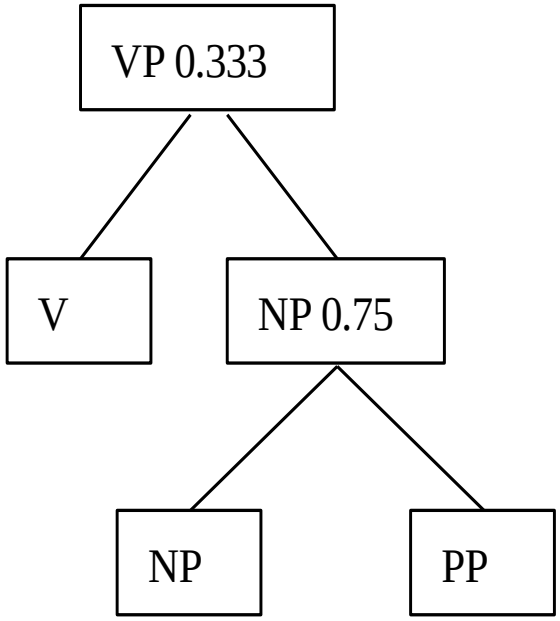
其中，$\Sigma_N^+ = \{N^1, N^2, \cdots, N^{|\Sigma_N^+|}\}$ 是一个非终结符集合；$\Sigma_T^+ = \{t^1, t^2, \cdots, t^{|\Sigma_T^+|}\}$ 是一个终结符集合；S 是一个起始符号，$S \in \Sigma_N^+$；$R = \{r^1, r^2, \cdots, r^{|R|}\}$ 是一组产生式，对于每个产生式 $r^i \in R$ 有 $(r^i : N^j \to \xi^k, P_r(r^i))$，其中，$\xi^k \in (\Sigma_T^+ \cup \Sigma_N^+)^*$，产生式概率 $P_r(r^i) = P_r(N^j \to \xi^k \mid N^j)$，表示句子推导过程中非终结符 N^j 被符号串 ξ^k 所替换的概率，它一般满足：$P_r(N^j \to \xi^k \mid N^j) > 0$；而且，$\forall N^j \in \Sigma_N^+$，$\sum_{\xi^k} P_r(N^j \to \xi^k \mid N^j) = 1$。

PCFG是CFG和概率的组合

设某个句子 S 有 λ 个可能的句法分析树，以 $\pi(S)=\{\pi_1,\pi_2,\cdots,\pi_\lambda\}$ 表示，则 S 的第 $i(1\le i\le\lambda)$ 棵分析树 π_i 的概率为该分析树所用到的所有规则的概率的乘积，即：$P_r(\pi_i)=\prod_j P_r(r_j)$；而句子 S 的概率则为它所包含的所有可能的分析树的概率之和，即：$P_r(S)=\sum_{i=1}^{\lambda}P_r(\pi_i)$；句法分析的目的就是从输入句子的所有可能的分析树中寻找最佳的分析树 $\hat{\pi}=\arg\max_{1\le i\le\lambda}P_r(\pi_i)$。实际上，概率 $P_r(\pi_i)$ 和 $P_r(S)$ 是句法分析的两个重要方面。$P_r(\pi_i)$ 反映了句子 S 的第 $i(1\le i\le\lambda)$ 棵候选分析树 π_i 的可能性；$P_r(S)$ 则反映句子 S 合乎语法的程度。

对V NP PP分析的结果



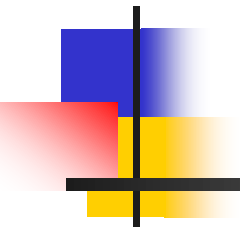


概率值

- 第一棵句法树的概率是P(T1) = 0.666
- 第二颗句法树的概率是P(T2) = 0.333*0.75 = 0.25
- P(V NP PP)=0.666+0.25=0.916

浅层句法分析(shallow parsing)

- 亦称部分句法分析(partial parsing)或语块分析(chunk parsing)
- 近年来自然语言处理领域出现的一种新的语言处理策略
- 不要求得到完全的句法分析树，只要求识别其中的某些结构相对简单的成分，如非递归的名词短语、动词短语等
- 识别出来的结构通常被称作语块(chunk)，和短语概念通常可以换用
- 浅层句法分析将句法分析分解为两个子任务：（1）语块的识别和分析；（2）语块之间的依附关系(attachment)分析
- 使句法分析的任务简化，同时也利于句法分析技术在大规模真实文本处理系统中迅速应用



谢谢！