

Masaru Tomita, 1987, An efficient augmented context-free parsing algorithm.
Computational Linguistics, vol. 13, No. 1, pp.31-46.

GLR 分析算法

詹卫东

<http://ccl.pku.edu.cn/doubtfire>

提 纲

1. 标准LR分析算法

1.1 从语法规则到状态转移

1.2 从状态集到分析表

1.3 标准LR分析算法示例

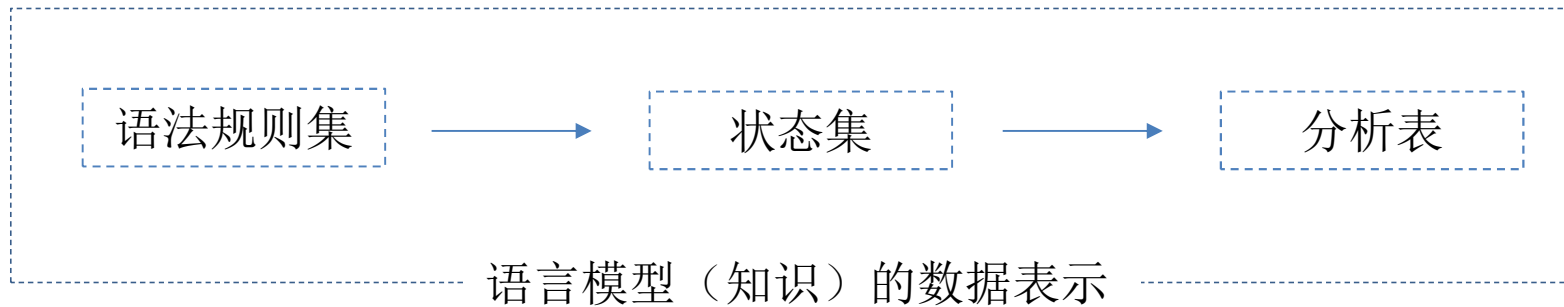
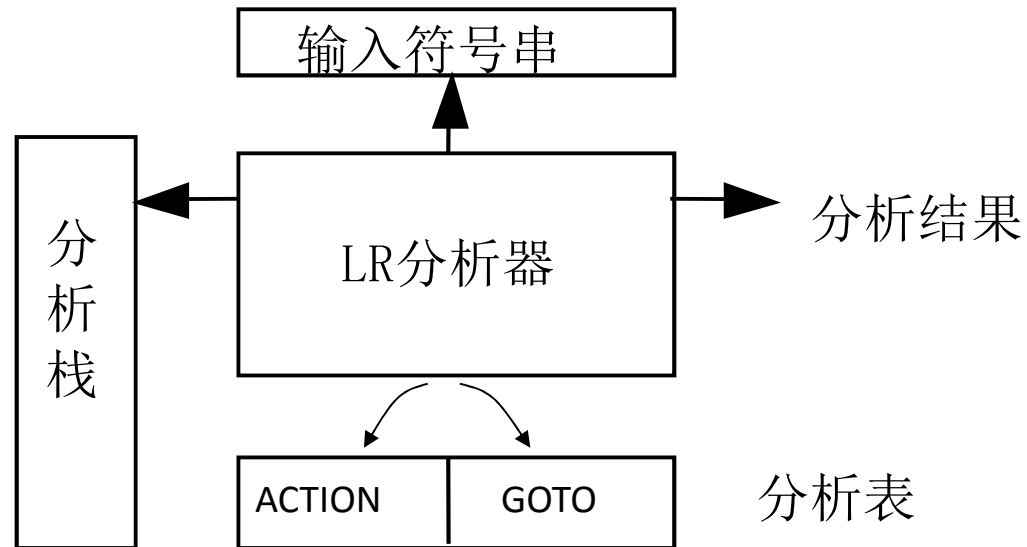
1.4 标准LR分析算法处理自然语言句子的不足

2. GLR分析算法：针对自然语言句法分析做的改进

- 分析表允许多重入口
- 分析栈由线性栈（单顶部）结构改为多重顶部栈结构
- 共享子树
- 局部歧义压缩

标准LR分析算法

(Left-to-right **R**educe)



LR分析算法的基本思想和基本概念

- 利用预读字符（Lookahead）和当前状态来决定下一步分析动作
- 状态由若干个二元组（项目）构成：<点规则, 规则完成后的后续字符>
- 分析动作：
 - 移进（shift）
 - 归约（reduce）
 - 成功（accept）
 - 报错（error）

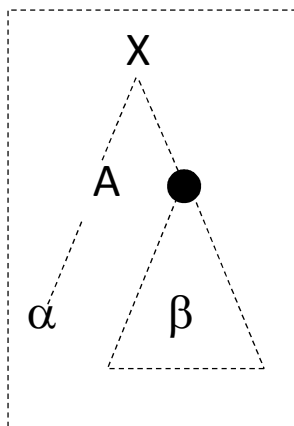
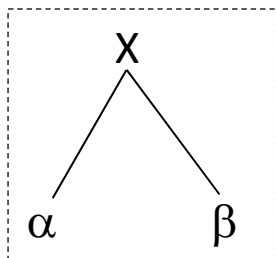
(1) $S \rightarrow NP VP$
(2) $NP \rightarrow N$
(3) $NP \rightarrow CS$ 的
(4) $VP \rightarrow V NP$
(5) $CS \rightarrow NP V'$
(6) $V' \rightarrow V V$

根据规则集中规则之间的相互制约关系，来判断当前规则的使用是否合理。比如：根据示例规则（5），CS后面一定是“的”；根据规则（1）（2）（5），NP后面要么是V，要么是\$（结束符）

First(x)函数 —— 实现Lookahead

$$\text{First}(x) = \{\alpha \mid x \overset{*}{\Rightarrow} \alpha \beta, \alpha \in V_T, \beta \in V_N \cup V_T\}$$

$\text{First}(x) = \{x\}$ 若 $x \in V_T$ 终结符的first集是它自身



示例:

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$



$\text{First}(S) = \{N\}$
 $\text{First}(NP) = \{N\}$
 $\text{First}(N) = \{N\}$
 $\text{First}(CS) = \{N\}$
 $\text{First}(VP) = \{V\}$
 $\text{First}(V') = \{V\}$

LR分析算法之[状态构造算法]

- 1) 首先在规则集中添加一条新规则 $S' \rightarrow S$, S' 为新的文法开始符号;
- 2) 定义0为起始状态, 项目 $\langle S' \rightarrow \cdot S, \$ \rangle$ 属于状态0, $\$$ 是输入串结束标志;
- 3) 对当前状态 i 内的项目进行扩展:
如果项目 $\langle x \rightarrow \alpha \cdot \gamma \beta, t \rangle$ 属于状态 i , 并且 $\gamma \rightarrow \gamma$ 是规则集中的一条产生式规则, 那么项目 $\langle \gamma \rightarrow \cdot \gamma, t' \rangle$ 也属于状态 i , 其中, 若 β 不为空, $t' \in \text{first}(\beta)$, 若 β 为空, 则 $t' = t$;
- 4) 从当前状态 i 转移到新的状态 j :
状态 j 是状态 i 遇到字符 γ (终结符或非终结符) 时的后继状态,
状态 i 中每个形如 $\langle x \rightarrow \alpha \cdot \gamma \beta, t \rangle$ 的项目, 变换为
状态 j 中的项目 $\langle x \rightarrow \alpha \gamma \cdot \beta, t \rangle$ (如 j 已存在同形式 k , 则 j 不加入状态集, 而是把 k 作为 i 的后继状态)
- 5) 重复上面3、4两步, 直至最终状态中项目均形如 $\langle x \rightarrow \alpha \gamma \beta \cdot, t \rangle$

注[1]: 状态 i 的后继状态 j 有 0 到多个, 具体数量为 i 中项目 $\langle x \rightarrow \alpha \cdot \gamma \beta \rangle$ 的不同字符类型数

“遇见”仅意味着“状态转移”的条件（对分析格局的预测），
比如1{0遇见N}，表示状态0如果遇到N，则转移到状态1

状态构造算法示例-1

规则集

- (0) $S' \rightarrow S$
- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS \text{ 的}$
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

状态是分析
过程中的一个个“格局”，
状态编号0,1,2,...等并无顺序
含义

0:

$\langle S' \rightarrow \cdot S, \$ \rangle$
 $\langle S \rightarrow \cdot NP VP, \$ \rangle$
 $\langle NP \rightarrow \cdot N, V \rangle$
 $\langle NP \rightarrow \cdot CS \text{ 的}, V \rangle$
 $\langle CS \rightarrow \cdot NP V', \text{的} \rangle$

2 {0遇见NP}:

$\langle S \rightarrow NP \cdot VP, \$ \rangle$
 $\langle VP \rightarrow \cdot V NP, \$ \rangle$
 $\langle CS \rightarrow NP \cdot V', \text{的} \rangle$
 $\langle V' \rightarrow \cdot V V, \text{的} \rangle$

1 {0遇见N}:

$\langle NP \rightarrow N \cdot, V \rangle$

NP可能是“N”，也可能是“CS
的”，但NP后面必是“V”，CS
后面必是“的”，S后面必是“\$”

3 {0遇见S}:

$\langle S' \rightarrow S \cdot, \$ \rangle$

如果预读到V，则可以
调用2号规则进行归约

如果预读到\$，则可以
调用0号规则进行归约

说明：状态6中，从 $\langle CS \rightarrow \cdot NP V', \text{的} \rangle$ 开始，根据算法步骤3，又可得到 $\langle NP \rightarrow \cdot N, V \rangle$ ，跟已有的 $\langle NP \rightarrow \cdot N, \$ \rangle$ 合并，即成 $\langle NP \rightarrow \cdot N, \$|V \rangle$

状态构造算法示例-2

$\text{First}(V') = \{V\}$

4 {0遇见CS}:
 $\langle NP \rightarrow CS \cdot \text{的}, V \rangle$

5 {2遇见VP}:
 $\langle S \rightarrow NP VP \cdot, \$ \rangle$

6 {2遇见V}:
 $\langle V' \rightarrow V \cdot V, \text{的} \rangle$
 $\langle VP \rightarrow V \cdot NP, \$ \rangle$
 $\langle NP \rightarrow \cdot N, \$|V \rangle$
 $\langle NP \rightarrow \cdot CS \text{ 的}, \$|V \rangle$
 $\langle CS \rightarrow \cdot NP V', \text{的} \rangle$

7 {2遇见V'}:
 $\langle CS \rightarrow NP V' \cdot, \text{的} \rangle$

8 {4遇见“的”}:
 $\langle NP \rightarrow CS \text{ 的} \cdot, V \rangle$

9 {6遇见V}:
 $\langle V' \rightarrow V V \cdot, \text{的} \rangle$

状态构造算法示例-3

10 {6遇见N}:
<NP \rightarrow N \cdot , \$ | V>

11 {6遇见NP}:
<VP \rightarrow V NP \cdot , \$>
<CS \rightarrow NP \cdot V', 的>
<V' \rightarrow \cdot V V, 的>

12 {6遇见CS}:
<NP \rightarrow CS \cdot 的, \$ | V>

13 {11遇见V}:
<V' \rightarrow V \cdot V, 的>

14 {11遇见V'}:
<CS \rightarrow NP V' \cdot , 的>

= 状态7

14 {12遇见“的”}:
<NP \rightarrow CS 的 \cdot , \$ | V>

15 {13遇见V}:
<V' \rightarrow V V \cdot , 的>

= 状态9

相同的状态，即相同的
“格局”（两个状态中所有项目都
相同），没有必要重新命名，
所以应该合并。

状态构造算法示例-4

0: <S' → · S, \$> <S → · NP VP, \$> <NP → · N, V> <NP → · CS 的, V> <CS → · NP V', 的>	1 {0遇见N}: <NP → N ·, V>	2 {0遇见NP}: <S → NP · VP, \$> <VP → · V NP, \$> <CS → NP · V', 的> <V' → · V V, 的>	3 {0遇见S}: <S' → S ·, \$>	4 {0遇见CS}: <NP → CS · 的, V>
5 {2遇见VP}: <S → NP VP ·, \$>	6 {2遇见V}: <V' → V · V, 的> <VP → V · NP, \$> <NP → · N, \$ V> <NP → · CS 的, \$ V> <CS → · NP V', 的>	7 {2遇见V'}: <CS → NP V' ·, 的>	8 {4遇见“的”}: <NP → CS 的 ·, V>	9 {6遇见V}: <V' → V V ·, 的>
10 {6遇见N}: <NP → N ·, \$ V>	11 {6遇见NP}: <VP → V NP ·, \$> <CS → NP · V', 的> <V' → · V V, 的>	12 {6遇见CS}: <NP → CS · 的, \$ V>	13 {11遇见CS}: <V' → V · V, 的>	14 {12遇见“的”}: <NP → CS 的 ·, \$ V>

LR分析算法 之 [分析表构造算法]

生成
转移
表

→1) 如果状态 s 遇见符号 x 转移到状态 s' ，那么在转移表（go to）中 s 为行， x 为列的格子里填入状态 s' （ s, s' 为整数， x 是非终结符或终结符）。

2) 条件同上。如果 x 是终结符，那么在动作表中的 s 为行、 x 为列的格子里填入动作“移进”（shift）。

生成
动作
表

3) 如果 s 中包含有项目元组 $\langle x \rightarrow \alpha \cdot, t \rangle$ ，其中 $x \rightarrow \alpha$ 是规则集中编号为 i 的产生式规则，那么在动作表中的 s 为行、 t 为列的格子里填入“归约 i ”（reduce）。

4) 如果 s 中包含有项目元组 $\langle S' \rightarrow S \cdot, \$ \rangle$ ，那么在动作表中的 s 为行、 $\$$ 为列的格子里填入“成功”（accept）。

5) 反复执行（1）—（4），直至所有状态均已遍历。最后动作表中所有没有填入内容的格子里的默认填入值为“报错”；转移表中所有没有填入内容的格子里的默认填入值为“不可转移”。

状态1中有 $\langle NP \rightarrow N \cdot, V \rangle$ ，对应第2条规则，因此，就在1行，V列中填“归约2”，表示调用第2条规则进行归约，意思是，碰到V时，对前面已经分析过的成分，比如N，进行归约（归约时调用第2条规则）

LR分析表示例-1

状态	动作表 (Action)				转移表 (Go to)							
	N	V	的	\$	N	V	的	S	NP	VP	CS	V'
0	移进				1			3	2		4	
1		归约 2										
2		移进				6				5		7
3				成功								
4			移进				8					
5				归约 1								
6	移进	移进			10	9			11		12	
7			归约 5									
8		归约 3										
9			归约 6									
10		归约 2		归约 2								
11		移进		归约 4		13						7
12			移进				14					
13		移进				9						
14		归约 3		归约 3								

只有遇到“终结符”，才可能发生“归约”操作，遇到非终结符，只可能发生“移进”操作

LR分析表示例-2

状态	动作表 (Action)				转移表 (Go to)				
	N	V	的	\$	S	NP	VP	CS	V'
0	移进 1				3	2		4	
1		归约 2							
2		移进 6					5		7
3				成功					
4			移进 8						
5				归约 1					
6	移进 10	移进 9				11		12	
7			归约 5						
8		归约 3							
9			归约 6						
10		归约 2		归约 2					
11		移进 13		归约 4					7
12			移进 14						
13		移进 9							
14		归约 3		归约 3					

“移进”后面的数字表示“状态编号”；
“归约”后面的数字表示规则编号。

LR分析算法过程描述-1

分析栈:

分析句法构造的所有操作都发生在“分析栈”
“输入缓冲区”的指针只用来预读下一个字符，
以决定分析栈里该如何操作

分析过程中，不断按“状态” — “字符” — “状态” — “字符”
— ... 的顺序向栈中压进当前分析状态及等待归约的字符

待分析字符串指针：指向输入缓冲区中当前待分析字符

输入：符号串 $W = w_1w_2\dots$ ，文法规则集 G ，LR分析表

输出：若 W 是合法句子，输出“成功”，否则输出“错误”

▶ 比如有状态“11 {6遇见NP}”， $x=6$ ， $A=NP$ ，则把NP,11压入栈中

LR分析算法过程描述-2

- 1) 把状态0压入分析栈， $w\$$ 放入输入缓冲区中，指针 p 指向 $w\$$ 的第一个符号；
- 2) 循环执行下面的语句
 - a) 设 s 是分析栈的栈顶状态，并且 c 是 p 所指向的当前字符；
 - b) 若 $\text{Action}[s, c]=\text{移进}k$ ，则把 c 和 k 先后压入分析栈中， p 指向下一个输入符号；
 - c) 若 $\text{Action}[s, c]=\text{归约}j$ ，并且第 j 条产生式为 $A \rightarrow \beta$ (β 长度为 m)，则
 - (i) 从栈顶弹出 $2*m$ 个符号；
 - (ii) 设 x 为当前栈顶，把 A 和 $\text{Go to}[x, A]$ 先后推入分析栈中；
 - d) 若 $\text{Action}[s, c]=\text{成功}$ ，则宣布分析成功，算法结束；
 - e) 若 $\text{Action}[s, c]=\text{报错}$ ，则宣布分析失败，算法结束；

说明： $\text{Action}[x,y]$ 表示分析表中 x 行， y 列的动作值，
 $\text{Go to}[x,y]$ 表示分析表中 x 行， y 列的转移值（下一个状态号）。

LR分析算法过程示例

action(0, N)=移进1, 把N, 1压入栈中

action(1, V)=归约2, 把N, 1弹出, 把NP, Go to[0, NP]=2压入栈中

action(2, V)=移进6, 把V, 6压入栈中

移进时,
将终结符
和状态号
压入栈中

缓冲区
待分析
字符数
减少

分析栈	字符串指针	规则使用序列
0	N V N V V 的	<>
0 N 1	V N V V 的	<>
0 NP 2	V N V V 的	<2>
0 NP 2 V 6	N V V 的	<2>
0 NP 2 V 6 N 10	V V 的	<2>
0 NP 2 V 6 NP 11	V V 的	<2, 2>
0 NP 2 V 6 NP 11 V 13	V 的	<2, 2>
0 NP 2 V 6 NP 11 V 13 V 9	的	<2, 2>
0 NP 2 V 6 NP 11 V' 7	的	<2,2,6>
0 NP 2 V 6 CS 12	的	<2,2,6,5>
0 NP 2 V 6 CS 12 的 14	\$	<2,2,6,5>
0 NP 2 V 6 NP 11	\$	<2,2,6,5,3>
0 NP 2 VP 5	\$	<2,2,6,5,3,4>
0 S 3	\$	<2,2,6,5,3,4,1>
成功	\$	<2,2,6,5,3,4,1>

归约时,
将规则右部
弹出, 将
规则左部
非终结符和
状态号
压入栈中

栈中
字符数
减少

分析树形成过程示意图-1

调用规则:

栈操作: 0压入栈中,
栈顶为0
栈: 0

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

N V N V V 的 \$

分析树形成过程示意图-2

调用规则：

栈操作：Action[0,N]=移进1，
将N 1压入栈中，指针下移到V，
栈顶为1

栈：0 N 1

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

N **V** N V V 的 \$

分析树形成过程示意图-3

调用规则： 2

栈操作： Action[1,V]=归约2,
第2条规则为NP→N,
将N 1弹出栈， 0为当前栈顶，
Goto[0,NP]=2,
将NP,2压入栈，
栈顶为2

栈： 0 NP 2

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

NP
|
N **V** N V V 的 \$

分析树形成过程示意图-4

调用规则：

栈操作：Action[2,V]=移进6，
将V 6压入栈中，指针下移到N，
栈顶为6

栈：0 NP 2 V 6

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

NP

|

N

V

N

V

V

的

\$

分析树形成过程示意图-5

调用规则：

栈操作：Action[6,N]=移进10,
将N 10压入栈中，指针下移到V，
栈顶为10

栈：0 NP 2 V 6 N 10

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

NP
|
N V N **V** V 的 \$

分析树形成过程示意图-6

调用规则：2

栈操作：Action[10,V]=归约2，
第2条规则为NP→N，
将N 10弹出栈，6为当前栈顶，
Go to[6,NP]=11，
将NP,11压入栈，
栈顶为11
栈：0 NP 2 V 6 NP 11

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

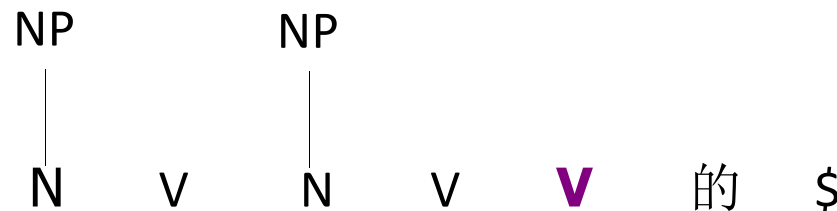


分析树形成过程示意图-7

调用规则:

栈操作: Action[11,V]=移进13,
将V 13压入栈中, 指针下移到“V”,
栈顶为13
栈: 0 NP 2 V 6 NP 11 V 13

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

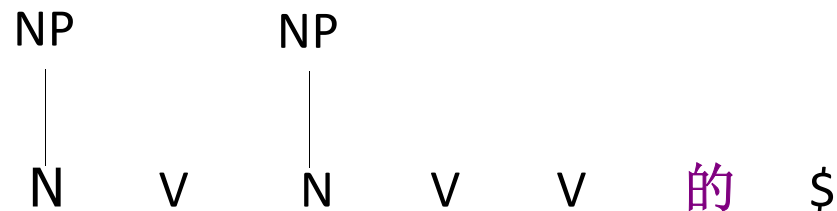


分析树形成过程示意图-8

调用规则：

栈操作：Action[13,V]=移进9,
将V 9压入栈中，指针下移到“的”，
栈顶为9
栈： 0 NP 2 V 6 NP 11 V 13 V 9

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

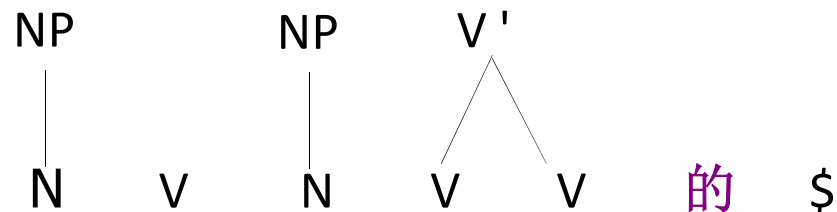


分析树形成过程示意图-9

调用规则：6

栈操作：Action[9,的]=归约6,
第6条规则 $V' \rightarrow V V$ ，长度为2
将V 13 V 9弹出栈，11为当前栈顶，
Go to[11, V']=7,
将V' 7压入栈中，
栈顶为7
栈：0 NP 2 V 6 NP 11 V' 7

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

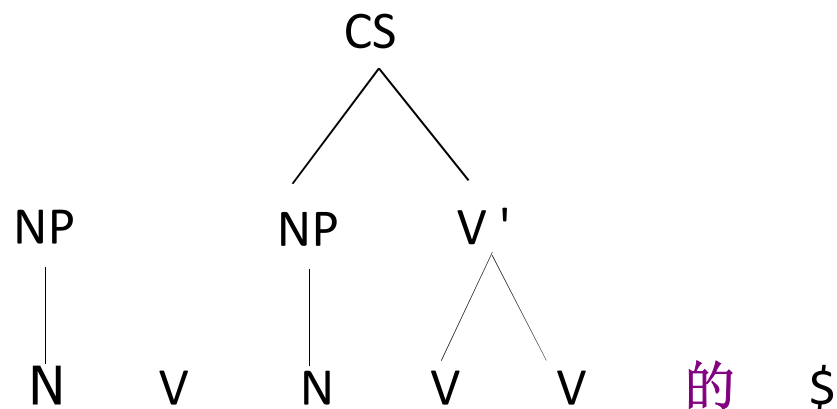


分析树形成过程示意图-10

调用规则：5

栈操作：Action[7,的]=归约5,
第5条规则 $CS \rightarrow NP V'$ ，长度为2
将NP 11 V' 7弹出栈，6为当前栈顶，
Go to[6, CS]=12,
将CS 12压入栈中，
栈顶为12
栈：0 NP 2 V 6 CS 12

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

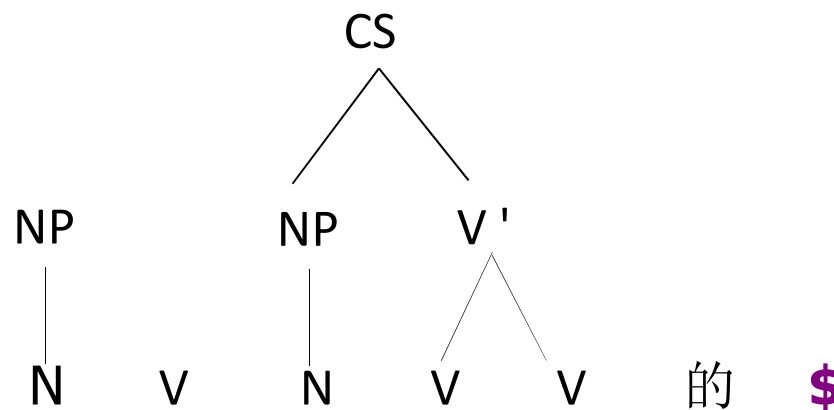


分析树形成过程示意图-11

调用规则:

栈操作: Action[12,的]=移进14,
将“的 14”压入栈中, 指针下移到\$
栈顶为14
栈: 0 NP 2 V 6 CS 12 的 14

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

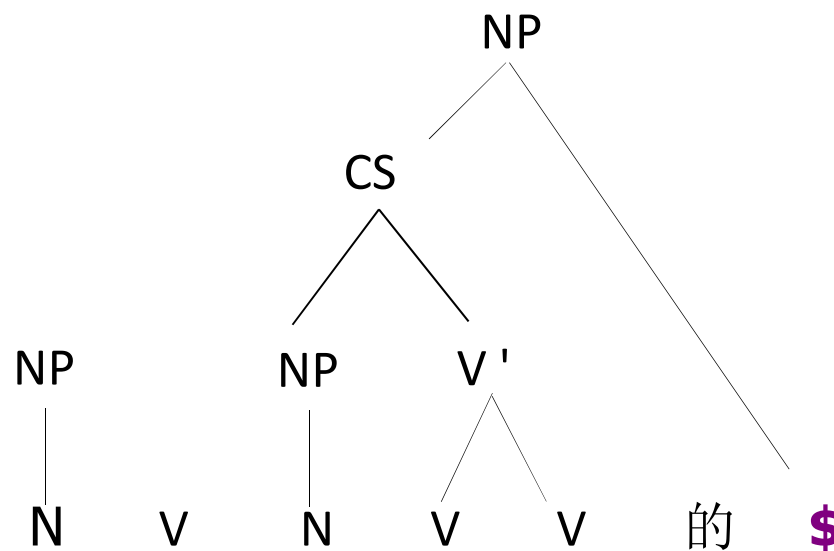


分析树形成过程示意图-12

调用规则：3

栈操作：Action[14,\$]=归约3,
第3条规则NP→CS 的，长度为2
将CS 12 的 14弹出栈，6为当前栈顶
Go to[6, NP]=11,
将NP,11压入栈，
栈顶为11
栈： 0 NP 2 V 6 NP 11

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

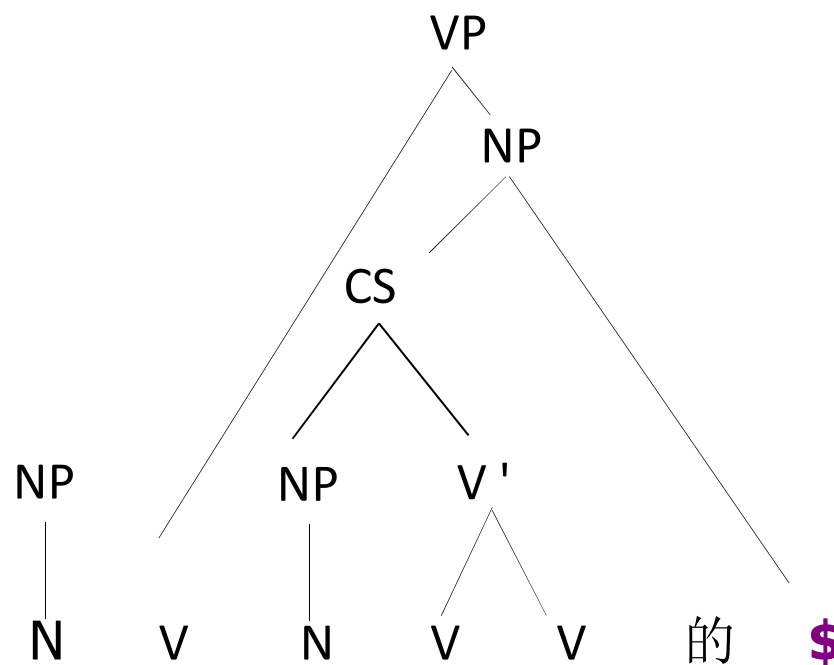


分析树形成过程示意图-13

调用规则：4

栈操作：Action[11,\$]=归约4,
第4条规则 $VP \rightarrow V \ NP$ ，长度为2
将V 6 NP 11弹出栈，2为当前栈顶
Go to[2, VP]=5,
将VP,5压入栈，
栈顶为5
栈：0 NP 2 VP 5

- (1) $S \rightarrow NP \ VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS \text{ 的}$
- (4) $VP \rightarrow V \ NP$
- (5) $CS \rightarrow NP \ V'$
- (6) $V' \rightarrow V \ V$

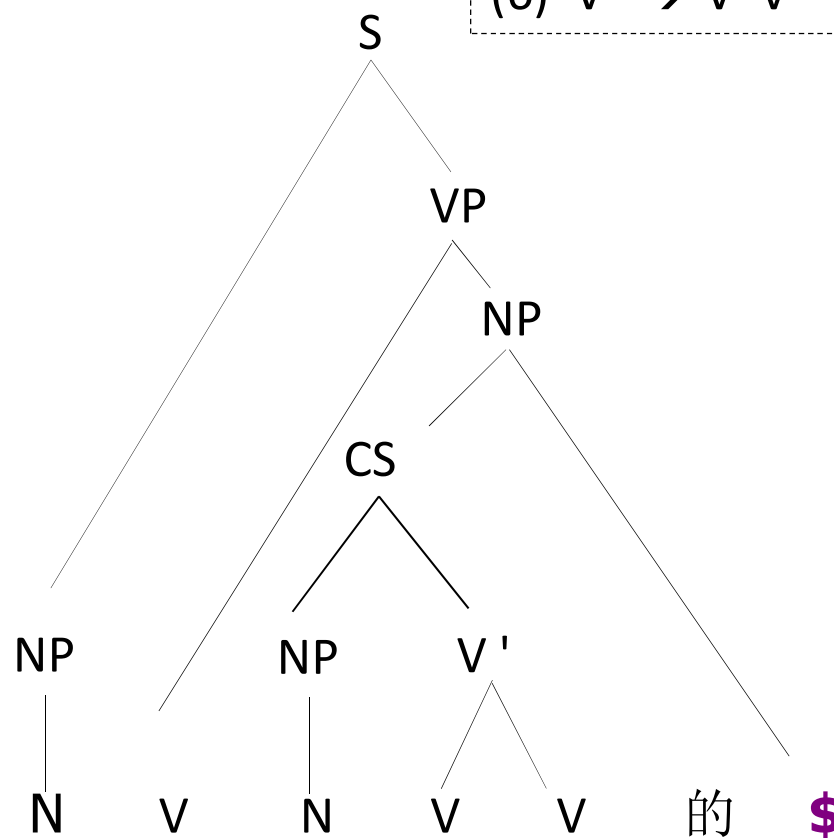


分析树形成过程示意图-14

调用规则：1

栈操作：Action[5,\$]=归约1,
第1条规则 $S \rightarrow NP VP$ ，长度为2
将NP 2 VP 5弹出栈，0为当前栈顶
Go to[0, S]=3,
将S,3压入栈，
栈顶为3
栈：0 S 3

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$

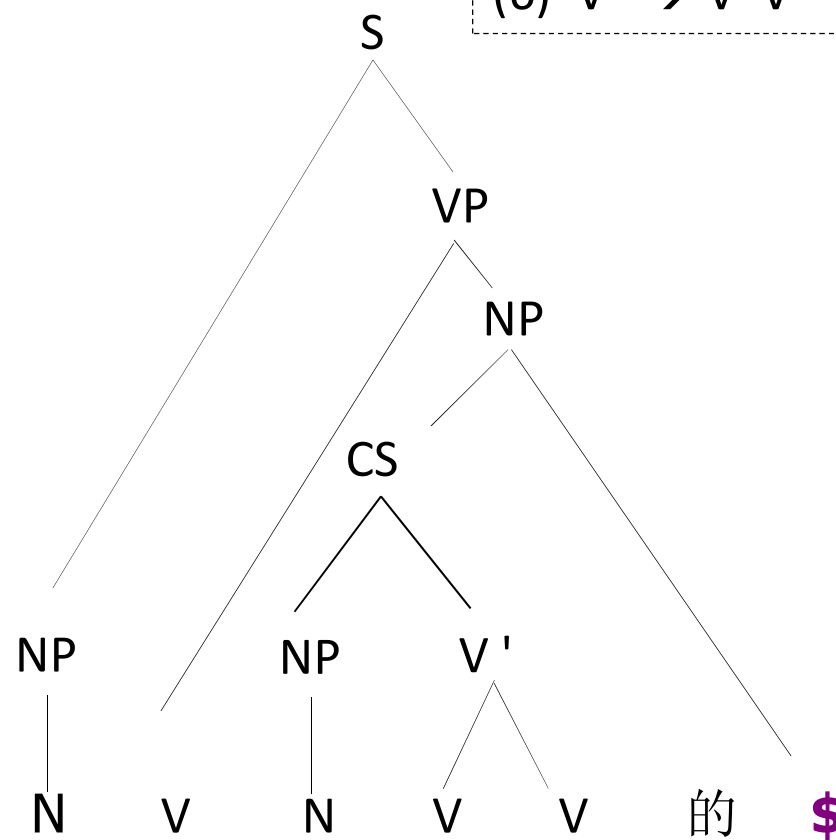


分析树形成过程示意图-15

调用规则：

栈操作：Action[3,\$]=成功,
分析成功，结束。

- (1) $S \rightarrow NP VP$
- (2) $NP \rightarrow N$
- (3) $NP \rightarrow CS$ 的
- (4) $VP \rightarrow V NP$
- (5) $CS \rightarrow NP V'$
- (6) $V' \rightarrow V V$



LR分析算法的不足

(1) 不允许回溯

(2) 只能分析标准LR文法，不能处理有歧义的文法

描述自然语言的文法不可避免地存在着歧义

因此，难以直接用LR分析算法分析自然语言

Tomita (1987) 对标准LR算法提出了改进：GLR算法

Tomita算法/Generalized LR算法

- 1) GLR分析表允许有多重入口（即一个格子里有多个动作）
- 2) 将线性分析栈改进为图分析栈处理分析动作的歧义（分叉）
- 3) 采用共享子树结构来表示局部分析结果，节省空间开销
- 4) 通过节点合并，压缩局部歧义

GLR分析示例

I	saw	a	girl	with	a	telescope
<i>Pron</i>	<i>V</i>	<i>Det</i>	<i>N</i>	<i>Prep</i>	<i>Det</i>	<i>N</i>

(0) $S' \rightarrow S$

(2) $VP \rightarrow V$

(4) $VP \rightarrow V NP NP$

(6) $NP \rightarrow Det N$

(8) $NP \rightarrow NP PP$

(1) $S \rightarrow NP VP$

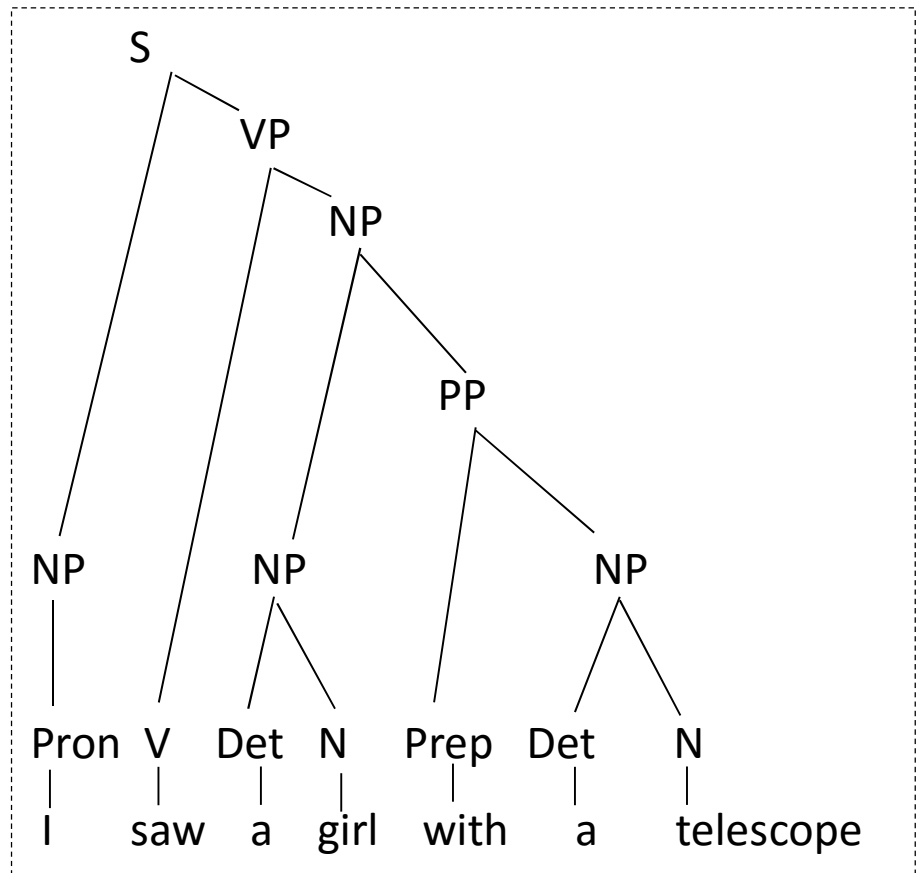
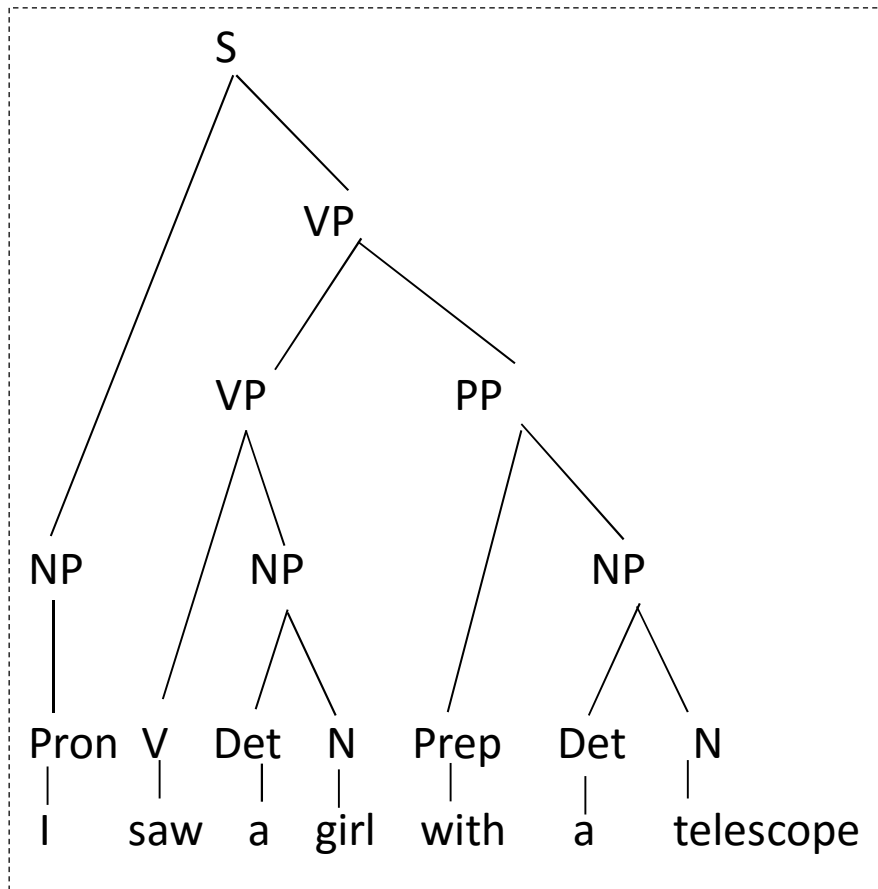
(3) $VP \rightarrow V NP$

(5) $VP \rightarrow VP PP$

(7) $NP \rightarrow Pron$

(9) $PP \rightarrow Prep NP$

两棵句法树



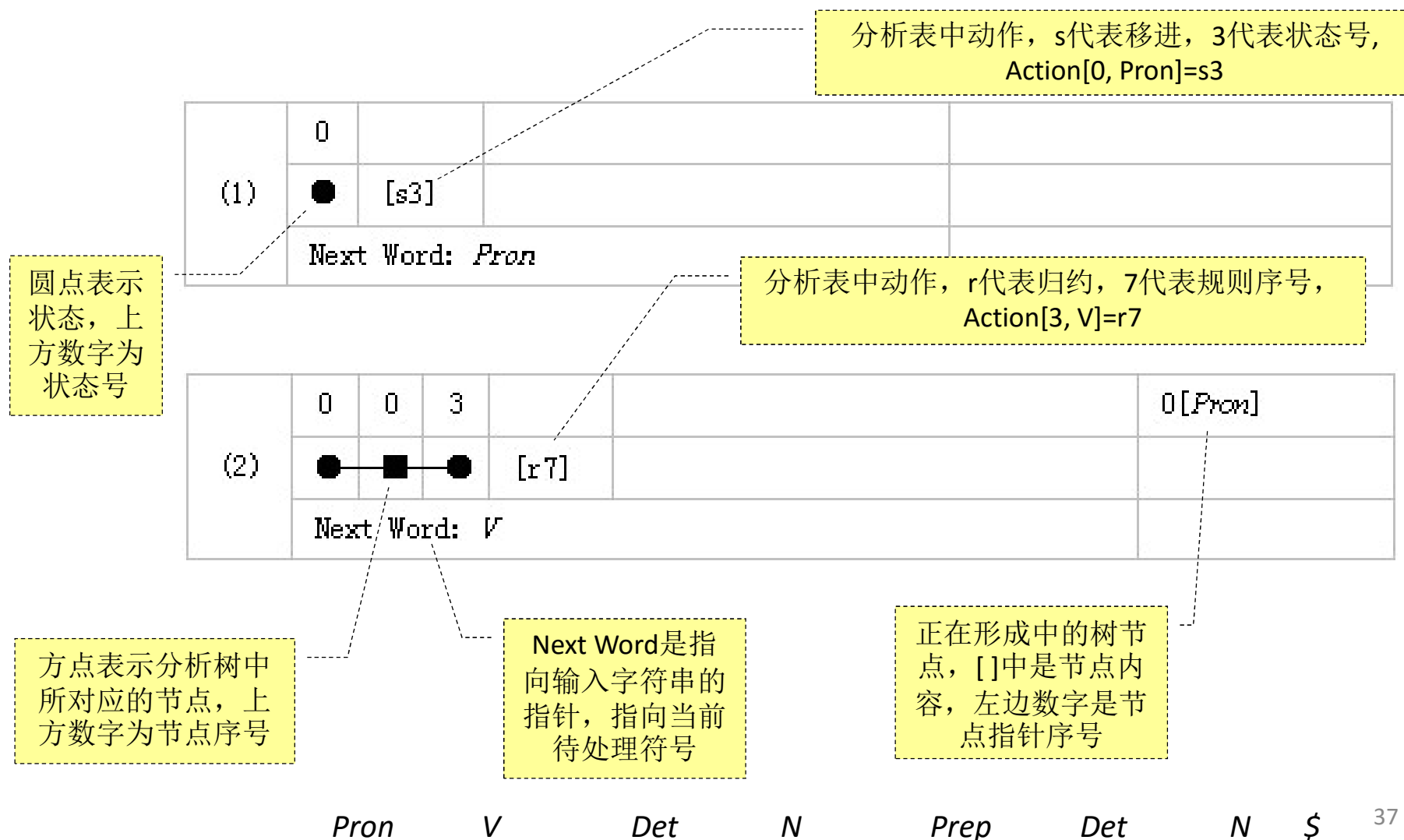
GLR分析表

s: shift
r: reduce

状态	ACTION						GOTO			
	<i>Det</i>	<i>N</i>	<i>Prep</i>	<i>Pron</i>	<i>V</i>	\$	<i>NP</i>	<i>PP</i>	<i>S</i>	<i>VP</i>
0	s2			s3			1		4	
1			s8		s6			7		5
2		s10								
3	r7		r7	r7	r7	r7				
4						acc				
5			s8			r1		9		
6	s2		r2	s3		r2	11			
7	r8		r8	r8	r8	r8				
8	s2			s3			13			
9			r5			r5				
10	r6		r6	r6	r6	r6				
11	s2		s8/r3	s3		r3	12	7		
12			s8/r4			r4		7		
13	r9		s8/r9	r9	r9	r9		7		

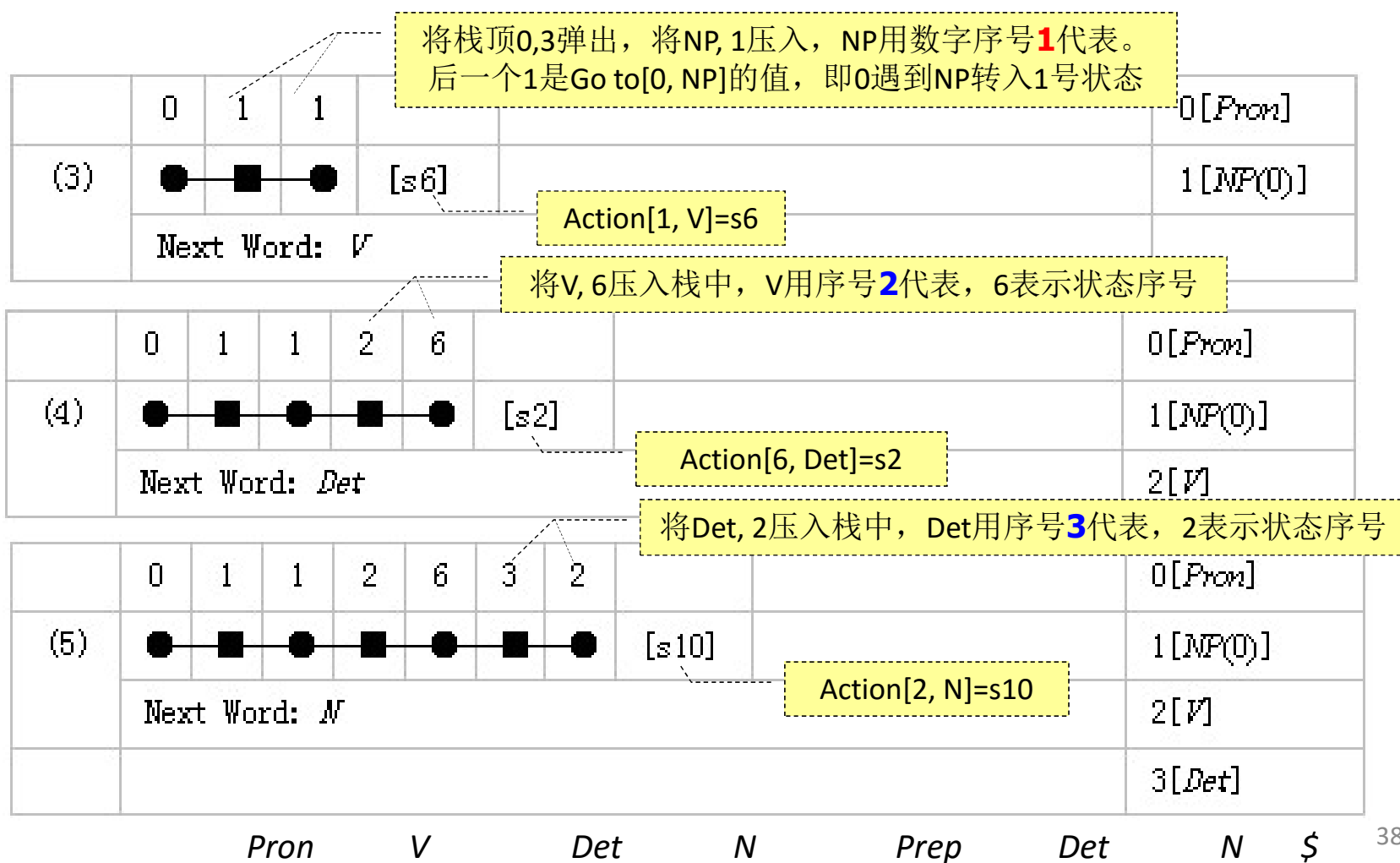
GLR分析过程-1

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |



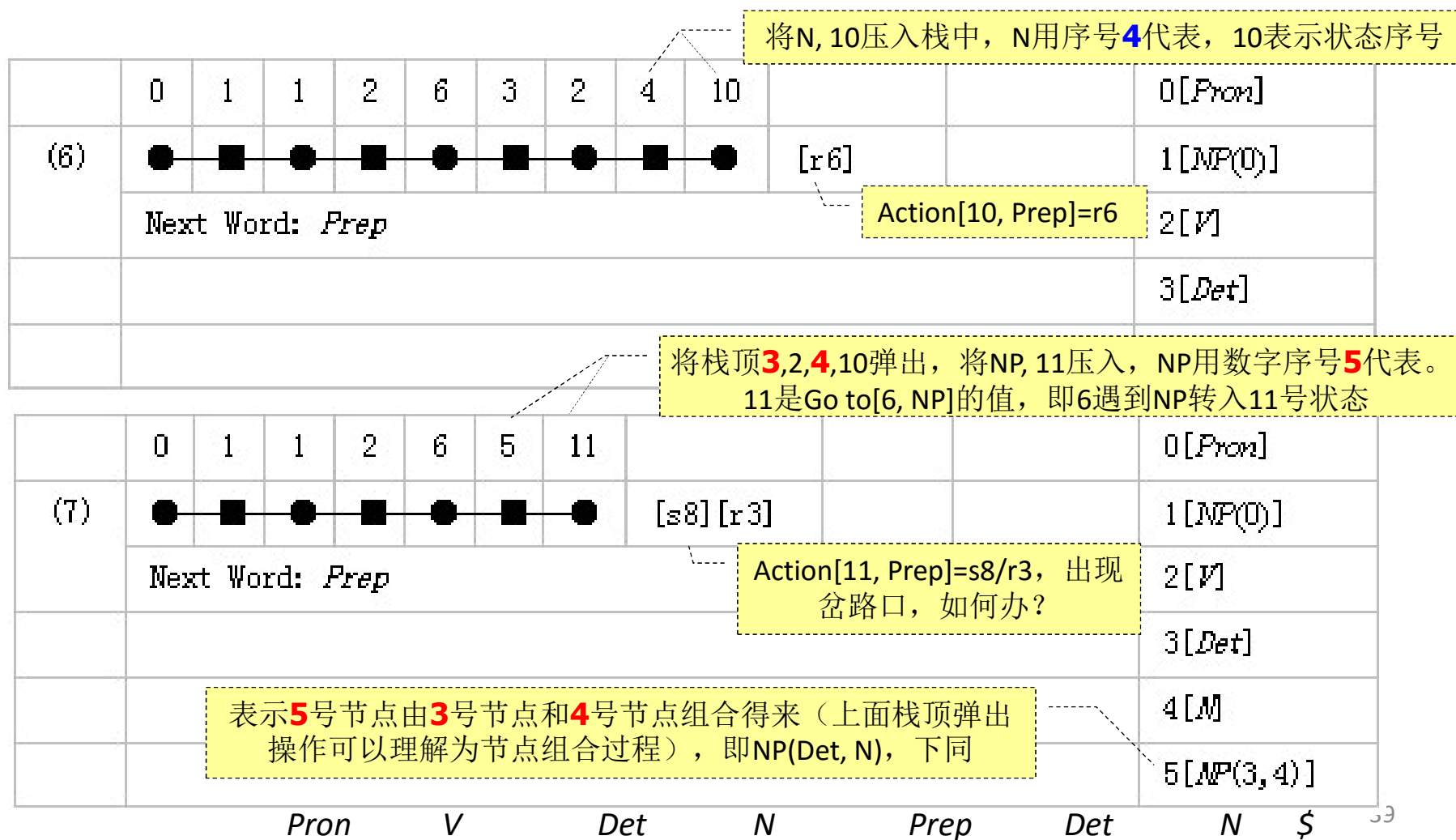
GLR分析过程-2

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |



GLR分析过程-3

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |



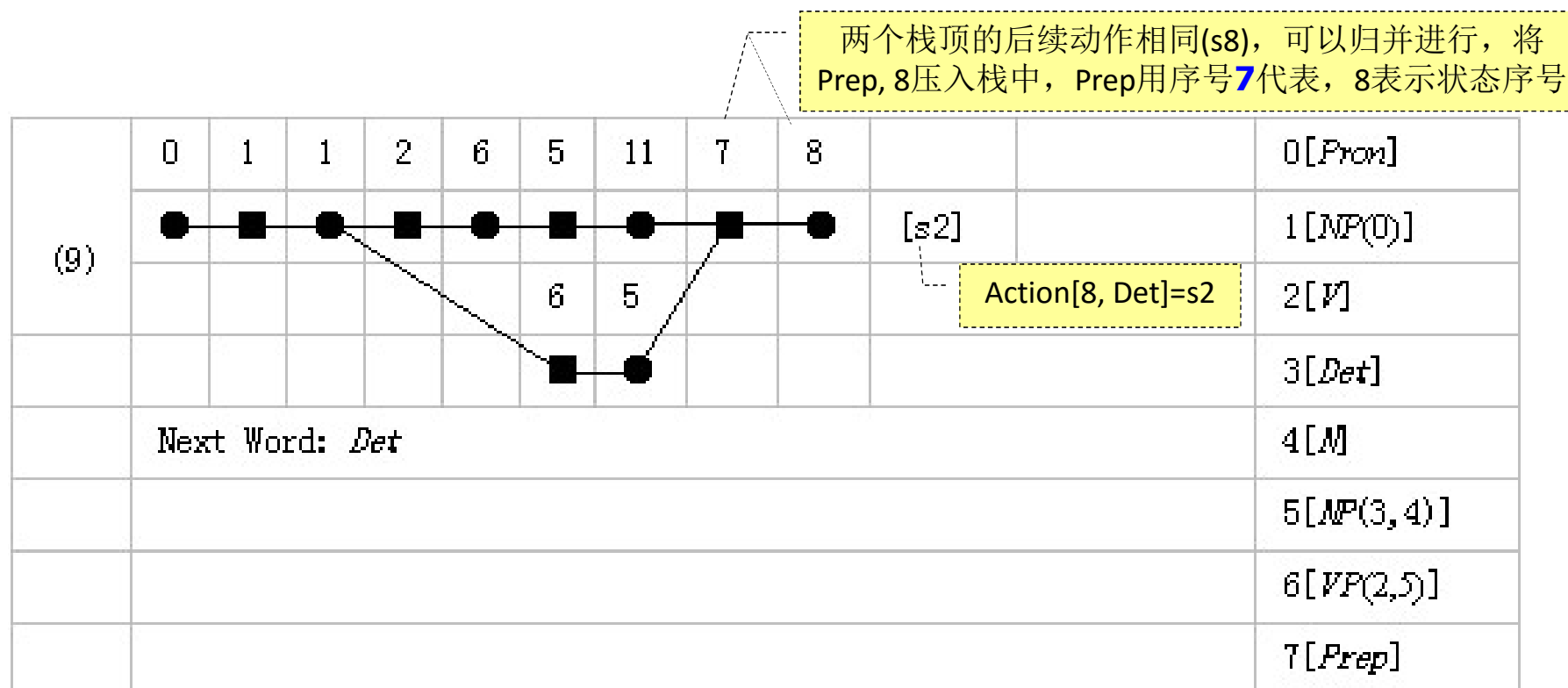
GLR分析过程-4

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |

(8)	0	1	1	2	6	5	11	[s8]	Action[11, Prep]=s8, 仍然保留	0[Pron]
	●	■	●	■	●	■	●			1[NP(0)]
						6	5			2[V]
						■	●	[s8]	Action[5, Prep]=s8	3[Det]
	Next Word: Prep									4[M]
	处理r3: 将当前栈顶4个元素复制一份(分叉), 然后将栈顶2,6,5,11弹出, 将VP, 5压入, VP用数字序号6代表。5是Go to[1, VP]的值, 即1号状态遇到VP转入5号状态									5[NP(3,4)]
										6[VP(2,5)]

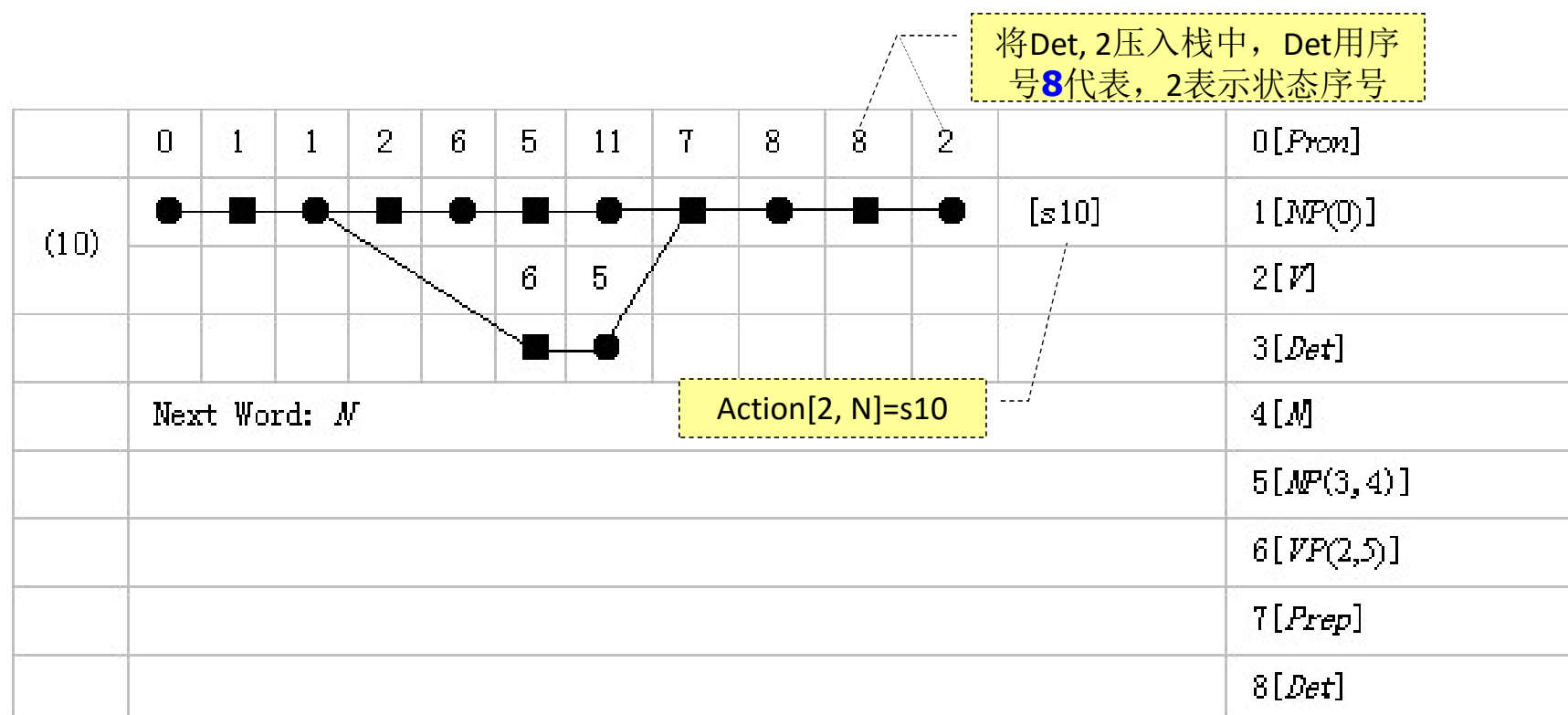
GLR分析过程-5

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |



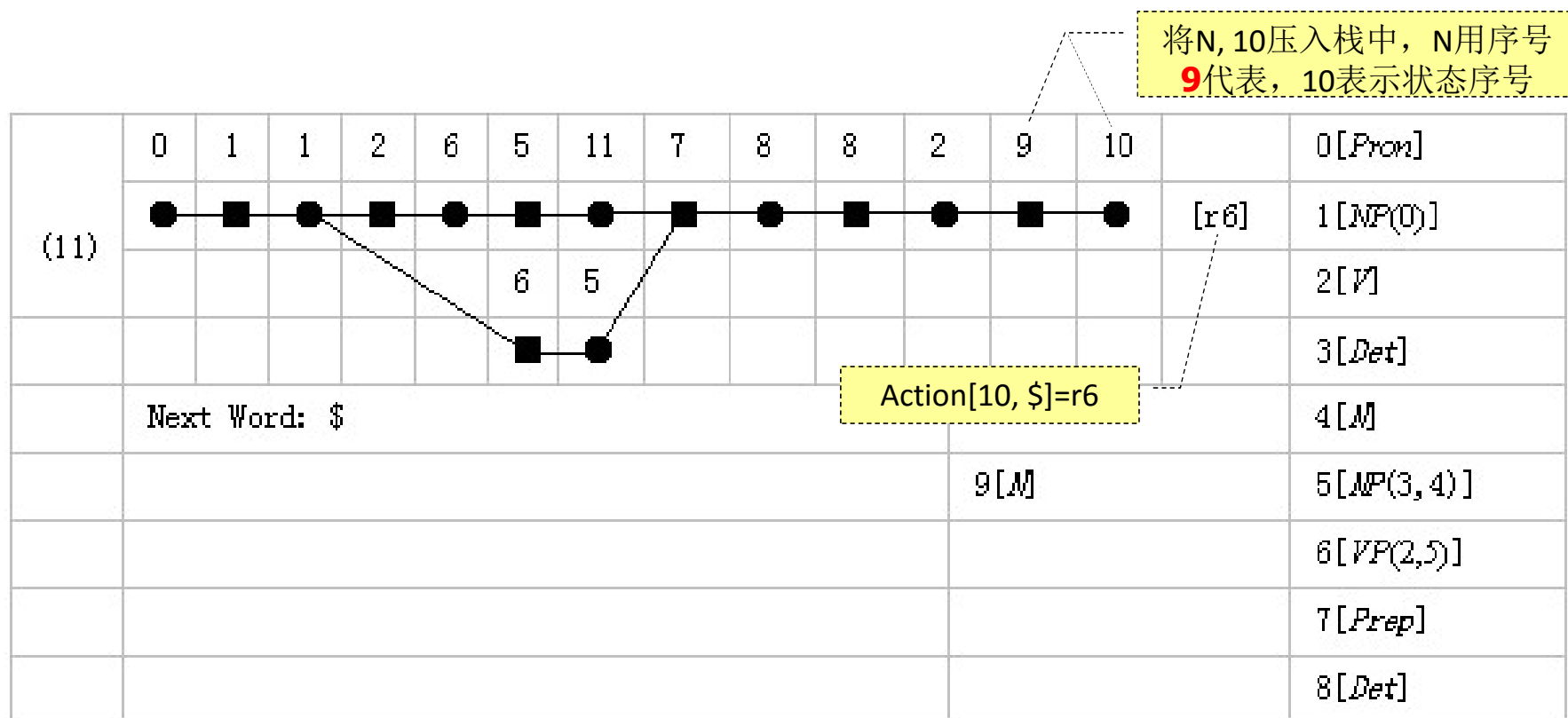
GLR分析过程-6

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |



GLR分析过程-7

- (0) $S' \rightarrow S$ (1) $S \rightarrow NP VP$
- (2) $VP \rightarrow V$ (3) $VP \rightarrow V NP$
- (4) $VP \rightarrow V NP NP$ (5) $VP \rightarrow VP PP$
- (6) $NP \rightarrow Det N$ (7) $NP \rightarrow Pron$
- (8) $NP \rightarrow NP PP$ (9) $PP \rightarrow Prep NP$



GLR分析过程-9

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |

将栈顶7,8,10,13弹出，刚好遇到分裂栈顶，将PP, 7和PP,9分别压入两个栈中，PP用数字序号11代表。7和9分别是Go to[11, PP]和Go to[5, PP]的值，即11号状态遇到PP转入7号状态,5号状态遇到PP转入9号状态

	0	1	1	2	6	5	11	11	7							0[Pron]
(13)	●	■	●	■	●	■	●	■	●	[r8]						1[NP(0)]
						6	5	11	9							2[V]
						■	●	■	●	[r5]						3[Det]
	Next Word: \$															4[M]
													9[M]			5[NP(3,4)]
													10[NP(8,9)]			6[VP(2,5)]
													11[PP(7,10)]			7[Prep]
																8[Det]

GLR分析过程-10

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |

将栈顶5,11,11,7弹出，将NP, 11压入，NP用数字序号12代表。
11是Go to[6, NP]的值，即6号状态遇到NP转入11号状态

	0	1	1	2	6	12	11									0[Pron]
(14)	●	■	●	■	●	■	●									1[NP(0)]
						6	5	11	9							2[V]
						■	●	■	●							3[Det]
	Next Word: \$															4[M]
														9[M]		5[NP(3,4)]
														10[NP(8,9)]		6[VP(2,5)]
														11[PP(7,10)]		7[Prep]
														12[NP(5,11)]		8[Det]

GLR分析过程-11

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |

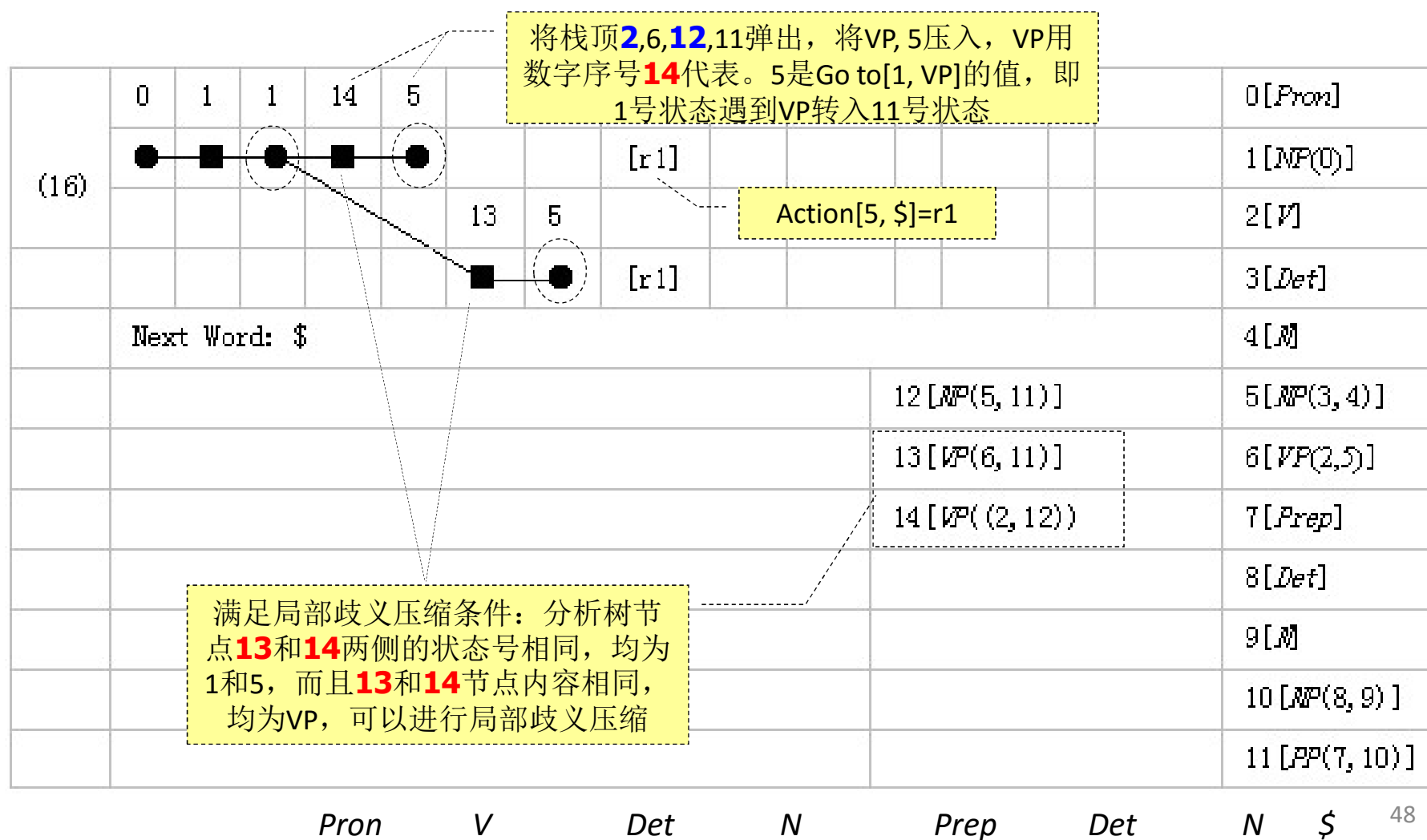
将栈顶6,5,11,9弹出，将VP, 5压入，VP用数字序号13代表。
5是Go to[1, VP]的值，即1号状态遇到VP转入5号状态

(15)	0	1	1	2	6	12	11									0[Pron]
	●	■	●	■	●	■	●	[r3]								1[NP(0)]
																2[V]
								13	5							3[Det]
								■	●	[r1]						4[M]
	Next Word: \$															5[NP(3, 4)]
																6[VP(2, 5)]
																7[Prep]
																8[Det]
																9[M]

Action[5, \$]=r1

GLR分析过程-12

- $$\begin{array}{ll} (0) S' \rightarrow S & (1) S \rightarrow NP VP \\ (2) VP \rightarrow V & (3) VP \rightarrow V NP \\ (4) VP \rightarrow V NP NP & (5) VP \rightarrow VP PP \\ (6) NP \rightarrow Det N & (7) NP \rightarrow Pron \\ (8) NP \rightarrow NP PP & (9) PP \rightarrow Prep NP \end{array}$$



GLR分析过程-13

- (0) $S' \rightarrow S$
- (1) $S \rightarrow NP VP$
- (2) $VP \rightarrow V$
- (3) $VP \rightarrow V NP$
- (4) $VP \rightarrow V NP NP$
- (5) $VP \rightarrow VP PP$
- (6) $NP \rightarrow Det N$
- (7) $NP \rightarrow Pron$
- (8) $NP \rightarrow NP PP$
- (9) $PP \rightarrow Prep NP$

	0	1	1	13	5										0[<i>Pron</i>]	
(17)	●	■	●	■	●	[r1]									1[NP(0)]	
	Next Word: \$															2[<i>V</i>]
													9[<i>M</i>]			3[<i>Det</i>]
													10[NP(8, 9)]			4[<i>M</i>]
													11[PP(7, 10)]			5[NP(3, 4)]
													12[NP(5, 11)]			6[VP(2, 5)]
													13[VP(6, 11) (2, 12)]			7[<i>Prep</i>]
																8[<i>Det</i>]

进行局部歧义压缩

Pron V Det N Prep Det N \$

GLR分析过程-14

- | | |
|------------------------------|------------------------------|
| (0) $S' \rightarrow S$ | (1) $S \rightarrow NP VP$ |
| (2) $VP \rightarrow V$ | (3) $VP \rightarrow V NP$ |
| (4) $VP \rightarrow V NP NP$ | (5) $VP \rightarrow VP PP$ |
| (6) $NP \rightarrow Det N$ | (7) $NP \rightarrow Pron$ |
| (8) $NP \rightarrow NP PP$ | (9) $PP \rightarrow Prep NP$ |

将栈顶1,1,13,5弹出，将S,4压入，S用数字序号14代表。4是Go to[0, S]的值，即0号状态遇到S转入4号状态

	0	14	4														0[Pron]
(18)	●	■	●	[acc]													1[NP(0)]
	Next Word: \$				Action[4, \$]=acc												2[V]
										9[M]							3[Det]
										10[NP(8, 9)]							4[M]
										11[PP(7, 10)]							5[NP(3, 4)]
										12[NP(5, 11)]							6[VP(2, 5)]
										13[VP(6, 11) (2, 12)]							7[Prep]
										14[S(1, 13)]							8[Det]

GLR分析树（压缩—共享森林）

