基于合一的Earley句法分析

詹卫东
zwd@pku.edu.cn

冯志伟、孙乐译《自然语言处理综论》电子工业出版社2005年版。第11章。
Daniel Jurafsky & James H. Martin, 2000, Speech and Language Processing, Pearson Education, Inc., Prentice Hall.

对Earley算法进行改进

（1）根据重写规后所附的合一约束为一个节点生成特征结构

（2）对chart中的状态进行改进

（3）对Predicator，Scanner，Completer等操作进行改进

（4）生成新状态时对特征结构的蕴涵关系进行检查

带合一约束的CFG规则

- 将重写规则中所附的约束转写为节点的特征结构

  S → *NP VP*

  <*NP* HEAD AGREEMENT> = <*VP* HEAD AGREEMENT>

  <*S* HEAD > = <*VP* HEAD>

$$\begin{bmatrix} \text{S} & [\ \text{HEAD} & \boxed{1}\ ] \\ \text{NP} & [\ \text{HEAD} & [\ \text{AGREEMENT} \quad \boxed{2}\ ]\ ] \\ \text{VP} & [\ \text{HEAD} & \boxed{1}\ [\ \text{AGREEMENT} \quad \boxed{2}\ ]\ ] \end{bmatrix}$$

- 特征结构可以实现为有向无环图(DAG)

为Earley分析法中的状态增加特征结构字段

- 改进后的状态包含4部分：

  （1）重写规则，代表分析子树

  （2）子树分析的完成状况，用点标记 · 表示

  （3）子树完成部分与输入中词的位置对应关系

  （4）特征结构

（1）$S \rightarrow \cdot NP\ VP$, $[0,0]$, DAG

（2） （3） （4）

Predicator操作

- 每当一个状态被Predicator操作加入状态表时，该规则对应的特征结构也作为状态的一个字段加入。

Predicator: 对于状态 $Z \rightarrow \alpha \cdot X\ \beta\ [j, k]\ DAG_z$ 其中X是非终结符
对于语法中每条形如 $X \rightarrow \gamma\ DAG_x$ 的规则，都可以形成一个新状态： $X \rightarrow \cdot \gamma\ \ [k, k]\ DAG_x$

Scanner操作

Scanner: 对于状态Z → α · X β [j, k] DAG_z 其中X是终结符

如果X与输入字符串中第k个字符匹配，就将词典中X的特征结构DAG_x跟DAG_z合一，若成功，则形成一个新状态：

Z → α X · β [j, k+1] New-DAG

否则，不改变当前状态集。

Completer操作

Completer : 对于一个已经“完成”的状态 $Z \rightarrow \gamma \cdot [j, k]$ DAG_z

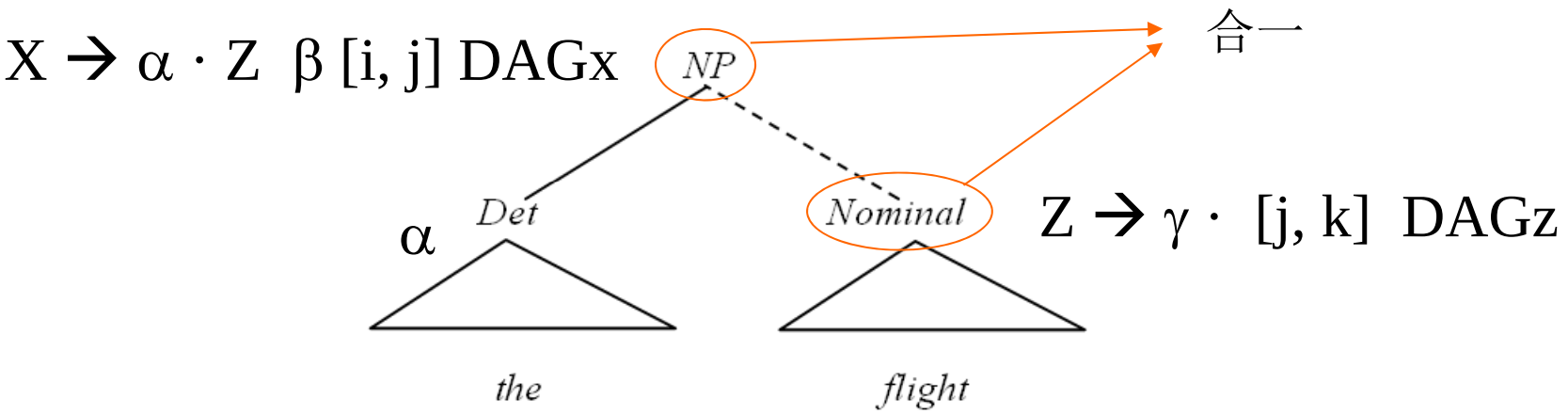
如果已有状态集合中有形如 $X \rightarrow \alpha \cdot Z\ \beta [i, j]$ DAG_x这样的状态，就将DAG_z跟DAG_x进行合一运算，二者合一的结果（记作New-DAG）若为成功，则形成一个新状态：

$X \rightarrow \alpha\ Z \cdot \beta [i, k]$ New-DAG

否则，不改变当前状态集。

Completer操作

- 每当一个子树(s_1)完成分析时，触发Completer操作
- 检查当前状态集中是否有子树(s_2)等待子树(s_1)来完成分析
- 将s_1对应的特征结构与s_2对应的特征结构进行合一
- 若合一失败，不产生新的状态
- 若合一成功，则产生新状态，并把合一结果作为新状态的特征结构



Completer示例

- *NP* → *Det* ·*Nominal*, [0,1], DAG_1

$$\begin{bmatrix} \text{NP} & \begin{bmatrix} \text{HEAD} & \boxed{1} \end{bmatrix} \\ \text{DET} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} & \boxed{2} \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{NOMINAL} & \begin{bmatrix} \text{HEAD} & \boxed{1} \begin{bmatrix} \text{AGREEMENT} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- *Nominal* → *Noun* ·, [1,2], DAG_2

$$\begin{bmatrix} \text{NOMINAL} & \begin{bmatrix} \text{HEAD} & \boxed{1} \end{bmatrix} \\ \text{NOUN} & \begin{bmatrix} \text{HEAD} & \boxed{1} \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- 将DAG_1中的*Nominal*同DAG_2中的*Nomial*进行合一

Completer操作中检查DAG蕴涵关系

- 为了保证不重复分析子树，在Earley算法中，如果新产生的状态和状态集合中的某个状态相同，新状态将不被加入状态集
- 在基于合一的Earley算法中，同时还要检查状态所附的特征结构是否具有蕴涵关系，若新状态的特征结构被状态集中的状态的特征结构蕴涵，则不被加入状态集。

NP → · *Det NP*, [*i*,*i*], *DAG*

若状态集中的状态对*Det*没有约束，而新产生的状态要求*Det*必须是单数的则新状态不必加入。

基于合一的Earley分析算法

设输入字符串长度为n, 字符间隔可记做0,1,2,…,n

（1）将文法规则中形如 $S\rightarrow \alpha\ DAG_s$ 的规则形成为状态：

<$S\rightarrow \cdot\alpha$ [0, 0] DAG_s> 加入到状态集合中（种子状态/seed state）

（2）对当前分析句子的每个词，依次进行循环：

对状态集中的每个状态，依次进行循环：

i）如果当前状态是[未完成状态]，且点后不是终结符，则

执行Predicator；

ii）如果当前状态是[未完成状态]，且点后是终结符，则

执行Scanner；

iii）如果当前状态是[完成状态]，则

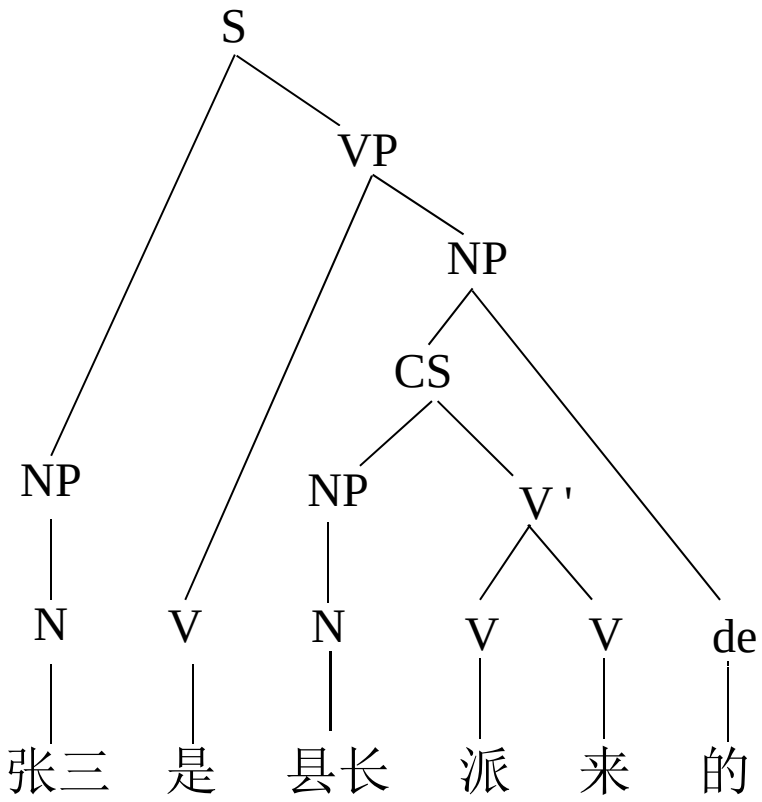
执行Completer；

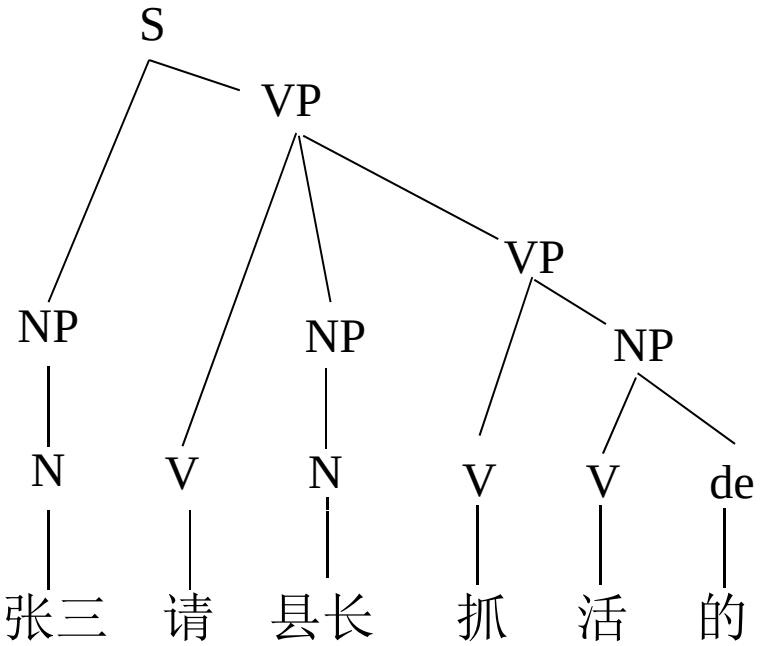
（3）如果最后得到形如<$S\rightarrow \alpha\cdot$ [0, n] DAG_s>这样的状态，那么输入字符串被接受为合法的句子，否则分析失败

基于合一的Earley分析法示例

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

(7) NP → V 的
(8) VP → V NP VP

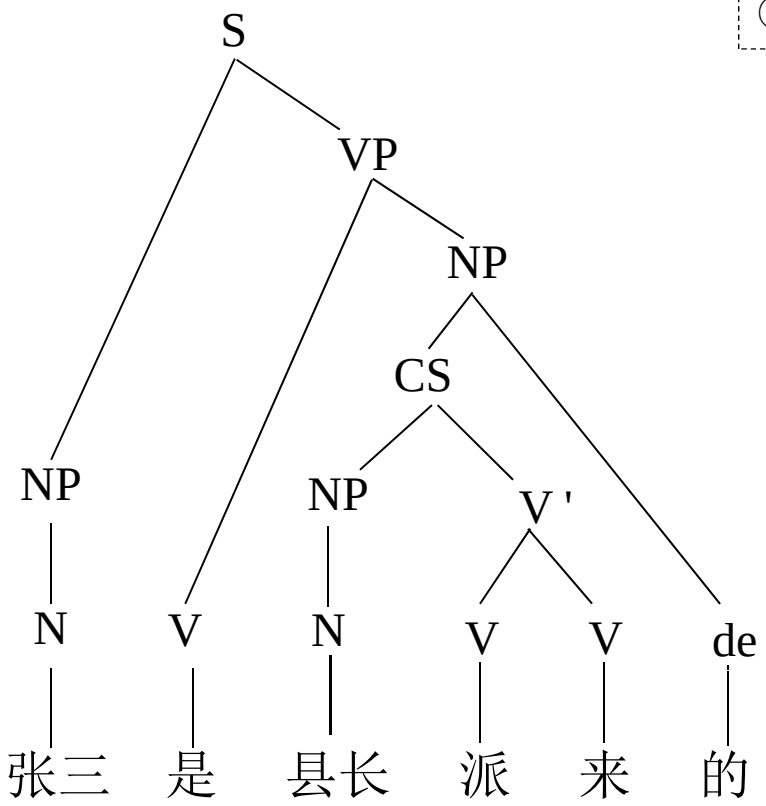


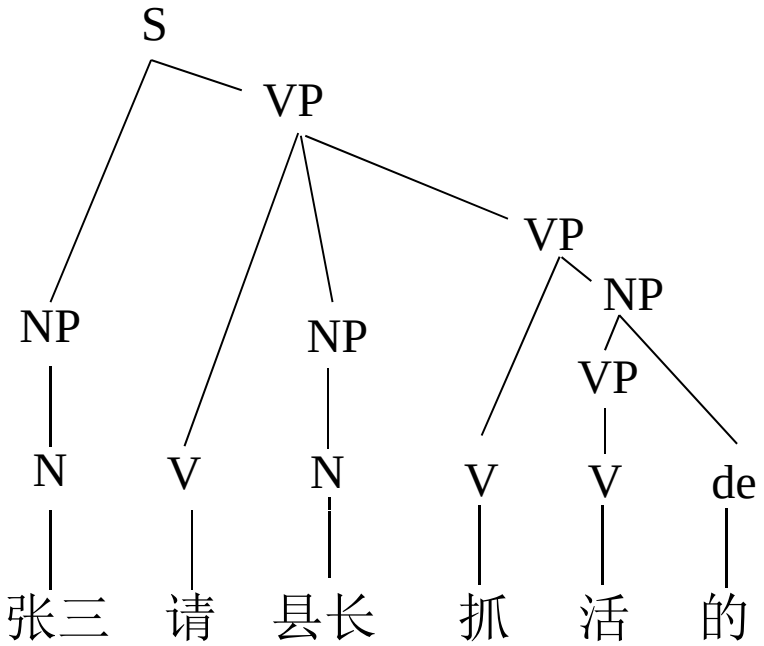


基于合一的Earley分析法示例

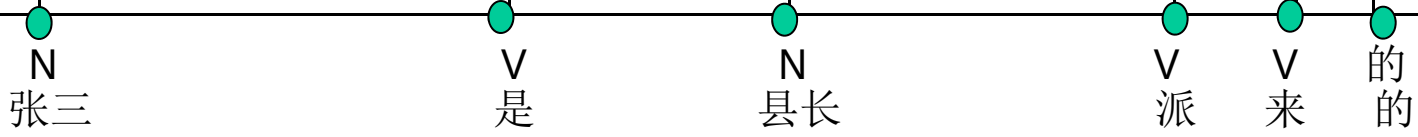
① 是 V［subcat:a ］
② 派 V［subcat:b ］
③ 来 V［subcat:c ］
④ 请 V［subcat:b ］
⑤ 抓 V［subcat:a ］
⑥ 活 V［subcat:d ］

(1) S → NP VP
(2) NP → N
(3) NP → CS 的
(4) CS → NP V'
(5) VP → V NP :: %V.subcat=a
(6) V' → V V :: %%V.subcat=c
(7) NP → V 的
(8) VP → V NP VP :: %V.subcat=b

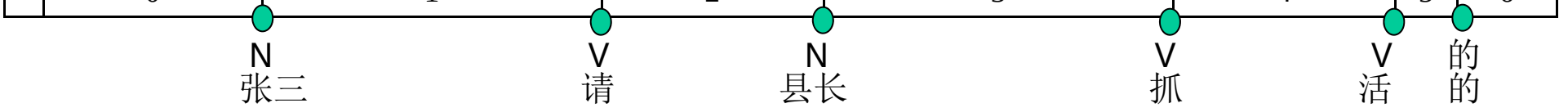




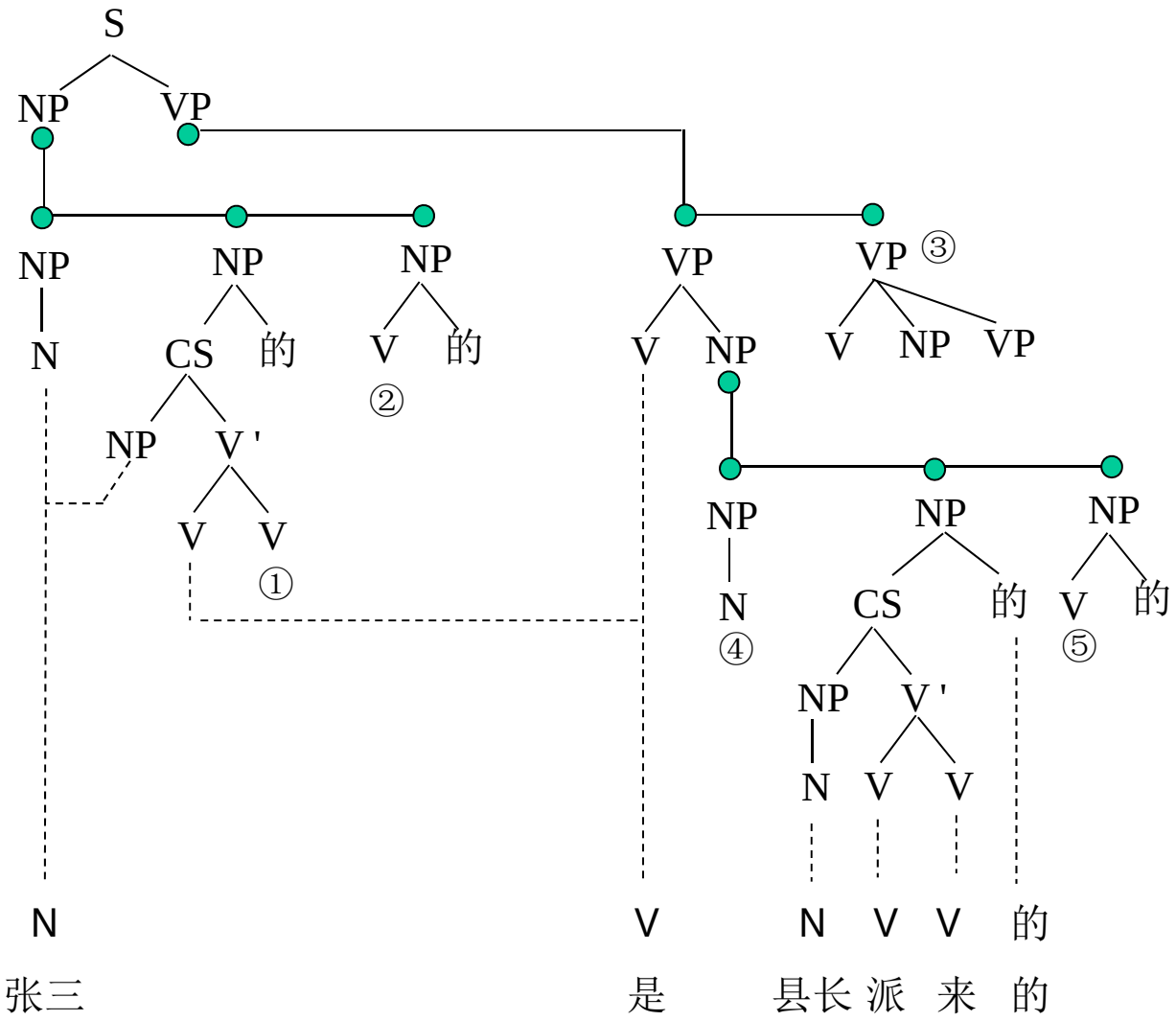
6	S → NP VP ·	VP → V NP · dag①-(5)	NP → CS 的 ·				归约 扫描
5			NP → CS · 的 CS → NP V ' ·	V ' → V V · dag②-(6)-③			归约 扫描
4				V ' → V · V dag②-(6)			扫描
3	S → NP VP ·	VP → V NP · dag①-(5)	CS → NP · V ' NP → N ·	V ' → · V V dag(6)			预测 归约 扫描
2		~~VP → V· NP VP dag①-(8)~~ VP → V· NP dag①-(5) V ' → V· V dag①-(6)	CS → · NP V ' NP → · V 的 NP → · N NP → · CS 的				预测 扫描
1	S → NP · VP CS → NP · V ' NP → N ·	VP → · V NP VP dag(8) VP → · V NP dag(5) V ' → · V V dag(6)					预测 归约 扫描
0	CS → · NP V ' NP → · V 的 NP → · N NP → · CS 的 S → · NP VP						预测 种子
	0	1	2	3	4	5	6



6	S → NP VP ·	VP →V NP VP · dag④-(8)		VP → V NP · dag⑤-(5) VP → V NP · VP dag⑤-(8)	CS→ NP · V ' NP → V 的 ·		归约 扫描
5				~~V ' → V V · dag⑤-(6)-⑥~~	NP → V · 的		扫描
4				V ' → V · V dag⑤-(6) VP → V · NP dag⑤-(5) VP → V · NP VP dag⑤-(8)	CS → · NP V ' NP → · V 的 NP → · N NP → · CS 的		预测 扫描
3		VP →V NP · VP dag④-(8)	CS → NP · V ' NP → N ·	V ' → · V V dag(6) VP → · V NP dag(5) VP → · V NP VP dag(8)			预测 归约 扫描
2		VP → V · NP VP dag④-(8) ~~VP → V · NP dag④-(5)~~ V ' → V · V dag④-(6)	CS → · NP V ' NP → · V 的 NP → · N NP → · CS 的				预测 扫描
1	S → NP · VP CS → NP · V ' NP → N ·	VP → · V NP VP dag(8) VP → · V NP dag(5) V ' → · V V dag(6)					预测 归约 扫描
0	CS → · NP V ' NP → · V 的 NP → · N NP → · CS 的 S → · NP VP						预测 种子
	0	1	2	3	4	5	6



Earley算法构造分析树示意图



① 因为“是”的后面不是V，该节点“到此为止”。

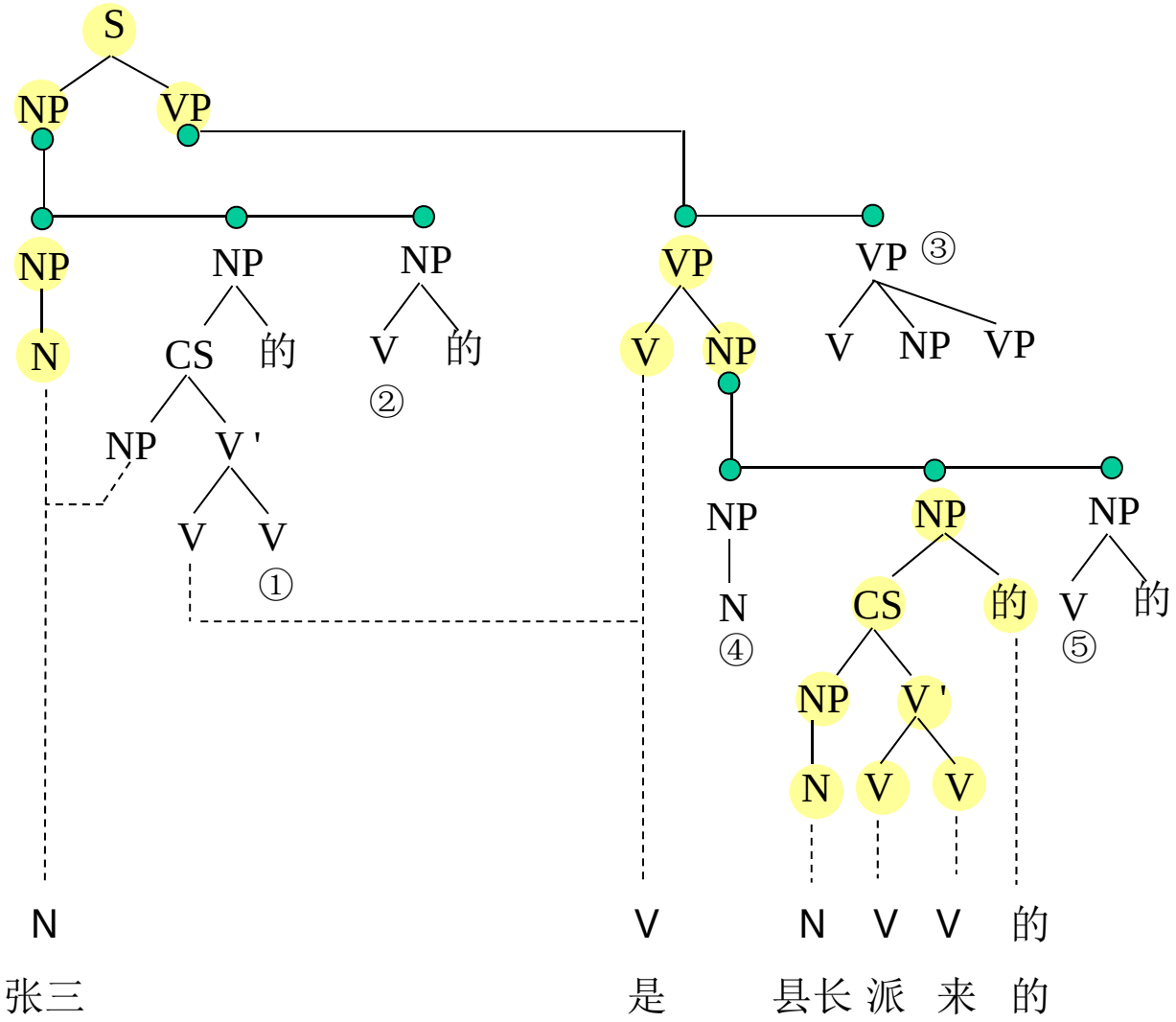
② 起始位置不是“V”。

③该规则的约束要求V的子类为b，而“是”的子类特征值为a，合一失败。

④ N如果跟“县长”匹配成功，则分析完毕，但句子并未结束。

⑤ V 跟“县长/N”不匹配

Earley算法构造分析树示意图



① 因为“是”的后面不是V，该节点“到此为止”。

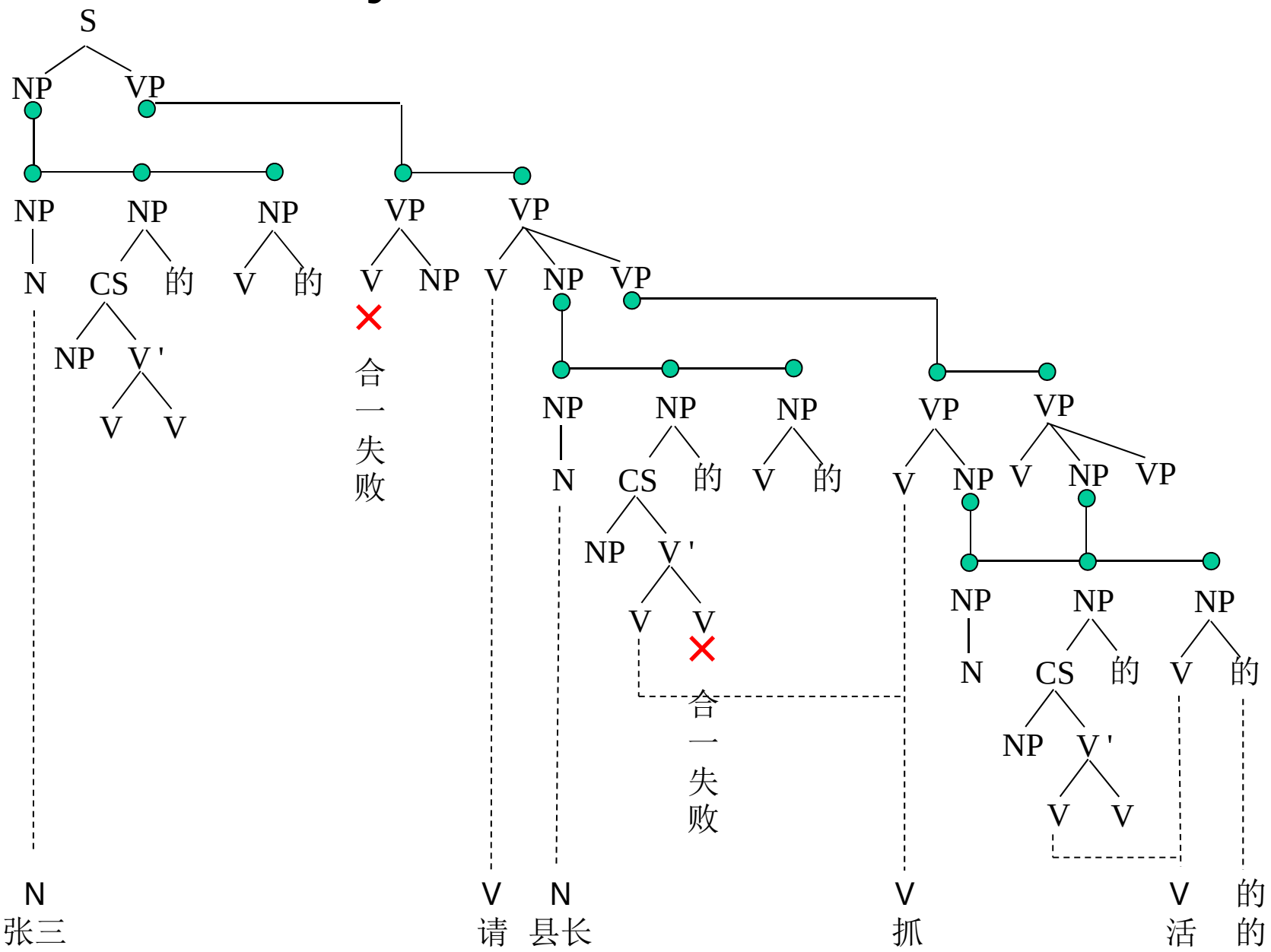
② 起始位置不是“V”。

③该规则的约束要求V的子类为b，而“是”的子类特征值为a，合一失败。

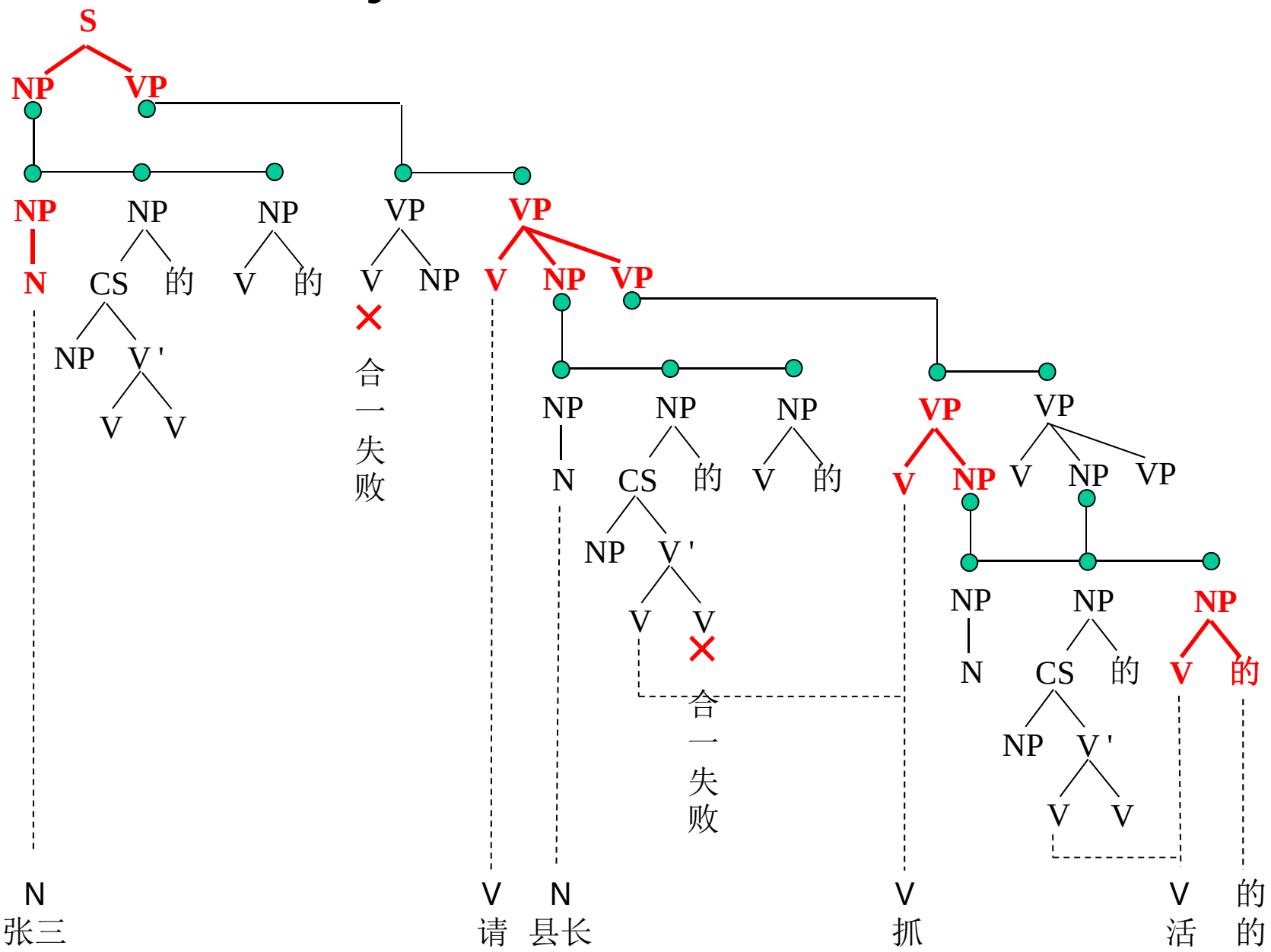
④ N如果跟“县长”匹配成功，则分析完毕，但句子并未结束。

⑤ V 跟“县长/N”不匹配

Earley算法构造分析树示意图



Earley算法构造分析树示意图



练 习

请构造带有合一描述的上下文无关文法规则集和词典，然后用Earley算法分析下面句子的结构：

1. 张三 请 李四 抓 活 的 回来
2. 张三 说 李四 是 要饭 的 装阔