计算语言学

第11讲 形式语法理论II

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

- 基于简单范畴的语法
  - 树粘接语法
  - 定子句语法
- 基于复杂范畴的语法
- 不使用范畴的语法
- 工程性语法

树粘接语法 1

- 树粘接语法、树邻接语法
  Tree Adjoining Grammar，Tree Adjunct Grammar
- A. Joshi, L. Levy, & M. Takahashi, 1975, Tree Adjunct Grammar, Journal of Computer & System Science, 1975, 10(1): pp136-163.
- Anne Abeillé, Owen Rambow, 2000, Tree Adjoining Grammars : Formalisms, Linguistic Analysis and Processing, CSLI Publications.
- http://www.cis.upenn.edu/~xtag/

树粘接语法 2

- 形式定义：
  - 树连接语法是树改写系统，由五元组G=(V_N, V_T,S,I,A)来表示；
  - V_N为非终结符(nonterminal)的有限集合；
  - V_T为终结符(terminal)的有限集合；
  - S为起始符，是V_N的一个元素；
  - I为初始树（Initial Tree）的有限集合；
  - A是辅助树（Auxiliary Tree）的有限集合。

树粘接语法 3

- I是初始树（Initial Tree）的有限集合，初始树定义如下：
  - 非叶结点都是非终结符
  - 叶子结点可以是终结符或者非终结符，非终结符的叶子结点都标记为“替换（Substitution）”，用下箭头↓表示；
- A是辅助树（Auxiliary Tree）的有限集合，辅助树定义如下：
  - 非叶结点都是非终结符；
  - 叶子结点可以是终结符或者非终结符，非终结符的叶子结点除了一个足结点（foot node）外都标记为“替换”，足结点用一个星号*表示，足结点的标记必须与根结点一致。

树粘接语法 4

- 基本树和导出树
  - 基本树（elementary tree）：初始树和辅助树又统称为基本树；
  - 导出树（derived tree）：由两棵树组合成的新树成为导出树；
- 树的组合操作
  - 替换（substitution）
  - 粘接（adjoining）

树粘接语法 5

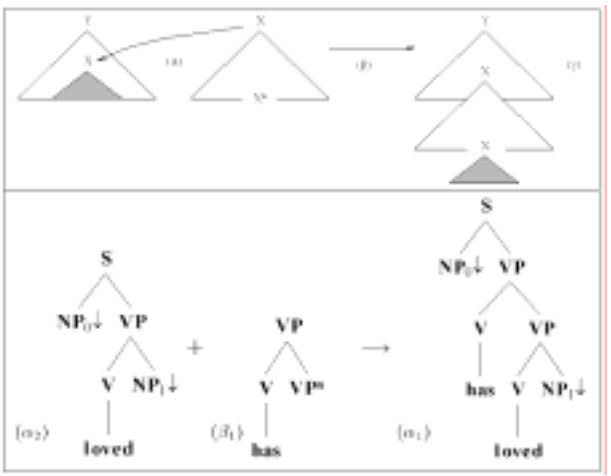
- 替换操作（substitution）

  将一棵树A的一个标记为“替换”的叶结点n用另外一棵树B来取代，B的根结点标记必须于结点n的标记相同；
- 粘接操作（adjoining）

  对于一棵树A和一棵辅助树B执行以下操作：
  - 将A的某一个子树t摘下；
  - 将A中原来子树t所在的位置用辅助树B替换，要求子树t的标记必须与B的根结点标记相同；
  - 将辅助树B上足结点用子树t替换；

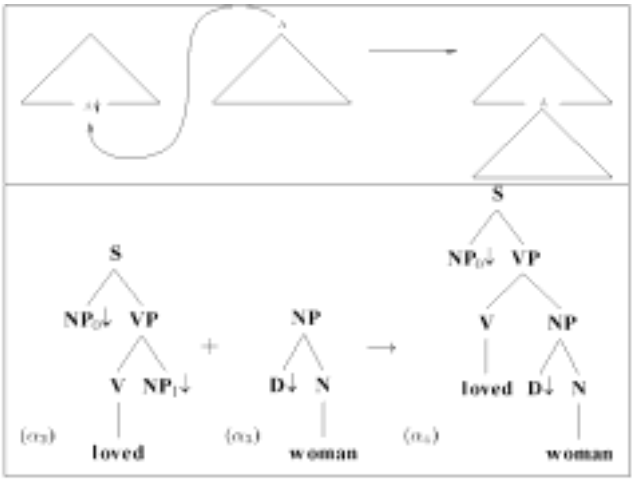
树粘接语法 6

粘接操作



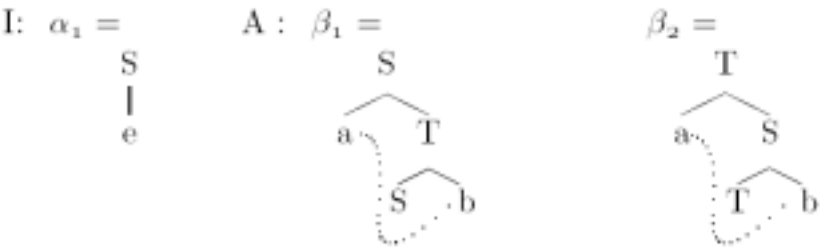
树粘接语法 7

替换操作



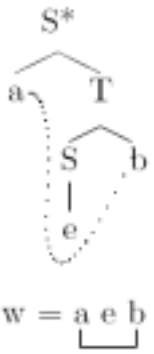
树粘接语法 8

- 例子：假设有以下语法G=(I,A)：



树粘接语法 9

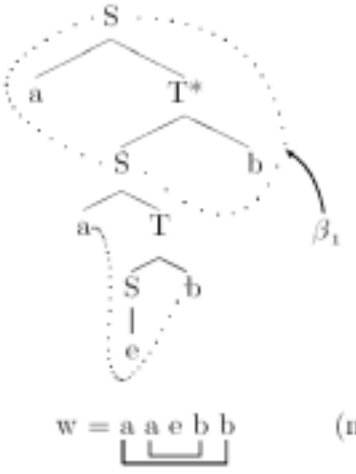
假设有一棵树：　　将β_1作用于S：



星号*表示将要执行粘接操作的位置

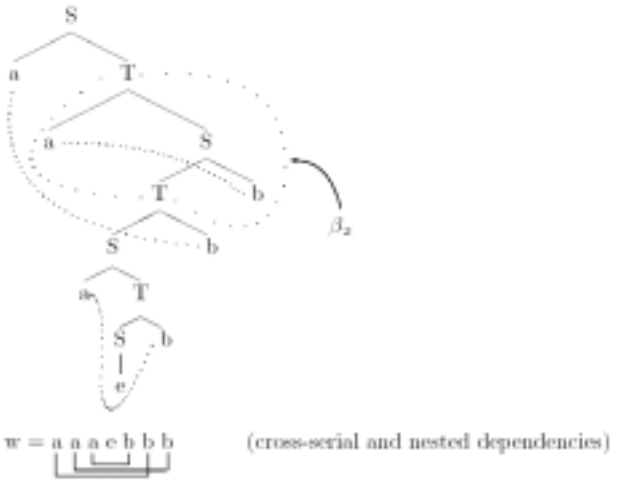
树粘接语法 10

将β_1作用于S：



树粘接语法 11

将β_2作用于T：



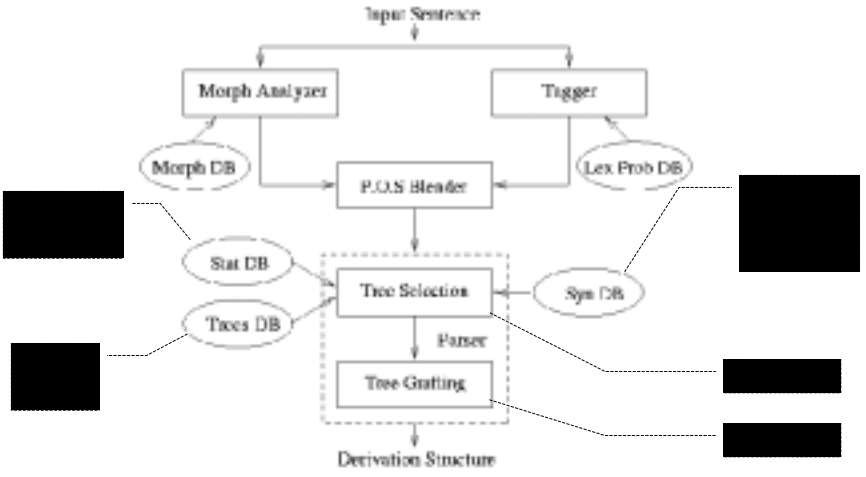
树粘接语法 12

- 树粘接语法是一种“温和（mild）”的上下文有关语法，其描述能力介于上下文有关语法和上下文无关语法之间； (如上述例子中交叉序列依赖就不可能由上下文无关文法产生)
- 树粘接语法的识别算法的时间复杂度是句子长度的6次方，相比之下，上下文无关语法的识别算法的时间复杂度是句子长度的3次方。

XTAG项目：LTAG理论针对英语的实践

- XTAG是宾州大学开展的一个采用LTAG来描述英语语法的项目
- LTAG是“词汇化树粘接语法（Lexicalized TAG）”，与普通树粘接语法的区别在于要求每一个初始树和辅助树都至少有一个终结符结点
- 一个LTAG树结构中的“词”节点被称作是这个树的“锚点”（anchor）
- LTAG中加入了基于合一的特征约束（unification-based features，参见后面介绍的特征结构与合一）

XTAG系统的结构



XTAG系统的基本组成与规模

- 形态分析器与形态数据库：大约317000个词汇（含各种屈折形式），其中根词大约90000个，所以记录以屈折形式建立索引，带有根词形式和词性信息及词缀信息
- 词性标注器（POS Tagger）和句法分析器（Parser）
- 句法数据库（Syn Database）：超过31000个词，每个词包含词性信息，它所属的树结构信息和特征约束信息；
- 树数据库（Tree Database）：1004棵树，由53个树家族（Tree Family）和221个其他类型的树结构（Individual Tree）组成；

定子句语法 1

- 定子句语法、Definite Clause Grammar
- 定子句语法是对CFG的一种简单扩充
- 定子句语法可以直接转换成Prolog语句
- 现在大部分Prolog语言都实现了对定子句语法的支持
- 在Prolog语言的支持下，定子句语法可以直接实现语言的识别、生成、分析，而不需要另外编程

定子句语法 2

- Prolog语言中对定子句语法的形式定义

  <definite-clause> ::= <LHS> --> <RHS> .

  <LHS> ::= <non-terminal>

  <nonterminal> ::= <identifier> (<arg> { , <arg> })

  <RHS> ::= <RHS-item> { , <RHS-item> }

  <RHS-item> ::= <nonterminal> | <terminal-list> | <logic>

  <terminal-list> ::= [<terminal> { , <terminal> }]

  <logic> ::= "{" <goal> "}"

- 上面最后一行<goal>表示Prolog语言中的其他目标。花括号用引号括住表示它不是表示重复多次的含义，而是语言中直接出现花括号。

定子句语法 3

- 从上面的定义可以看到，定子句语法在非终结符中引入了多个参数，这样可以简化上下文无关文法的表达。如下例所示：

$$
\begin{aligned}
S &\rightarrow NP(x_{num})\, VP(x_{num}) \\
NP(x_{num}) &\rightarrow PN(x_{num}) \\
VP(x_{num}) &\rightarrow TV(x_{num})\, NP(x_{num}) \\
VP(x_{num}) &\rightarrow IV(x_{num}) \\
PN(s) &\rightarrow bob \\
PN(p) &\rightarrow they \\
IV(s) &\rightarrow flies \\
IV(p) &\rightarrow fly \\
TV(s) &\rightarrow watches
\end{aligned}
$$

- 如果用上下文无关语法表示，那么分别要将单数和复数形式的NP、VP、PN、IV、TV等定义为不同的非终结符，并写出几套不同的规则，规则多而且重复。

定子句语法 4

- 定子句语法可以直接翻译成标准Prolog语句：
  - 将单个方括号中的终结符列表替换成多个独立方括号的列表，每个方括号中只有一个非终结符。例如将 [and, then] 变成 [and], [then]；
  - 在上面的基础上，计算规则右部RHS-item的项数，计算时跳过花括号中的逻辑子句(<goal>)；
  - 在规则左边的非终结符的参数列表中增加两个参数：L_0 和 L_n；
  - 对于规则右部的第i个RHS-item，如果是非终结符，那么在非终结符的参数表中增加两个参数L_{i-1}, L_i；
  - 对于规则右部的第i个RHS-item，如果是终结符t，那么将它替换成目标：'C'(L_{i-1}, *t*, L_i)。
  - 对于花括号中的RHS-item（<goal>）不做任何变化；
  - 将符号 --> 替换成 :- 。
  - 说明：谓词'C'(L_1,t,L_2) 成立的条件是L_1=[t|L_2]。

定子句语法 5

- 例子：给出定子句语法及其对应的Prolog语句：

S --> NP, VP	S(L0,L2) :- NP(L0,L1), VP(L1,L2)
NP --> det, N	NP(L0,L2) :- det(L0,L1), N(L1,L2)
VP --> V, NP	VP(L0,L2) :- V (L0,L1), NP(L1,L2)
det --> the	Det(L0,L1) :- 'C'(L0,the,L1)
N --> boy	N(L0,L1) :- 'C'(L0,boy,L1)
N --> girl	N(L0,L1) :- 'C'(L0,girl,L1)
V --> likes	V(L0,L1) :- 'C'(L0,likes,L1)

定子句语法 6

- 给定上述语法，求解：

```
|?- s([the,boy,likes,the,girl],[]).
Yes
or
|?- s(S,[]).
S = [the,boy,likes,the,boy] ;
S = [the,boy,likes,the,girl] ;
S = [the,girl,likes,the,boy] ;
S = [the,girl,likes,the,girl] ;
no
```


定子句语法 7

- 对于上述第一个查询，调用过程为：

```
call: s([the,boy,likes,the,girl],[])
call: np([the,boy,likes,the,girl],T1)
call: det([the,boy,likes,the,girl],T2)
exit: det([the,boy,likes,the,girl],[boy,likes,the,girl])
call: n([boy,likes,the,girl],T1)
exit: n([boy,likes,the,girl],[likes,the,girl])
call: vp([likes,the,girl],[])
...
```

定子句语法 8

- 通过给DCG子句添加一个表示句法结构的参数，还可以使用Prolog来进行句法分析。进行句法分析的DCG如下所示：

```
s(s(NP)) --> np(NP), vp(VP).
vp(vp(VG, NP)) --> verb_group(VG), np(NP).
vp(vp(Verb, NP)) --> verb(Verb), np(NP).
verb_group(verb_group(Aux, Verb)) --> aux(Aux), verb(Verb).
np(np(Noun)) --> noun(Noun).
np(np(Adj, Noun)) --> adj(Adj), noun(Noun).
adj(adj(flying)) --> [flying].
aux(aux(are)) --> [are].
noun(noun(planes)) --> [planes].
pronoun(pronoun(they)) --> [they].
verb(verb(are)) --> [are].
verb(verb(flying)) --> [flying].
```

- 查询：| ?- *s(Tree, [they,are,flying,planes],[]).* 返回*Tree*为句法树

定子句语法 9

- 通过给定子句语法添加其他的Prolog目标，还可以进行语义计算。例如给出数学表达式的定子句语法：

```
expr(Z) --> num(Z).
expr(Z) --> num(X), "+", expr(Y), {Z is X+Y}.
expr(Z) --> num(X), "-", expr(Y), {Z is X-Y}. num(Z) --> num(Z, 0).
num(Z, Accum) --> digit(D1), {Z is 10*Accum+D1}.
num(Z, Accum) --> digit(D1), {A2 is 10*Accum+D1}, num(Z, A2).
digit(Z) --> [D], {"0" =< D, D =< "9", Z is D - "0"}.
expr_value(L, V) :- expr(V, L, []).
```

- 查询：

```
| ?- expr_value("301+20+3+5", S).
S = 329 ?
yes
| ?- expr_value("324-24+1", V).
V = 299 ?
```

基于简单范畴的语法的缺陷

- 范畴划分的粗细度难以把握

  例如英语句子的构成规则：

  S → NP VP

  如果考虑到英语主谓语单复数的搭配，就要将NP和VP分成NPsingular和NPplural和VPsingular和VPplural，并其将规则改写成：

  S → NPsingular VPsingular

  S → NPplural VPplural

  这不仅导致上述规则要写成两套，而且所有与NP、VP相关的规则都至少要写两套，如果考虑到主谓语人称的搭配、语义的搭配，NP和VP的范畴还要细分，这将会导致规则中出现大量的冗余，规则库不堪重负，难以维护

对单一特征的改进

- 采用类似DCG中添加参数的办法

  将he表示为：R(person3,singular,subject)

  将is表示为：V(person3,singular,present)

- 采用乔姆斯基标准理论中次范畴化（subcategorization）的方法

  将he表示成：R,+person3,+singular,+subject

  将is表示成：V,+person3,+singular,+present

- 这两种处理方法能够起到一定的作用，不过在处理更为复杂的约束情形时，表达起来仍然不够方便

特征结构的引入

- 特征结构（Feature Structure）
  复杂特征集（Complex Feature Set）
- 特征结构定义为“特征”的集合
- 所谓“特征”，是一个由“属性”和“值”组成的二元组，“属性”也称为“特征名”，“值”也称为“特征值”
- 在特征结构中，要求所有的“特征”的“属性”互不相同

$$\begin{pmatrix} attribute_1=value_1 \\ attribute_2=value_2 \\ \cdots \\ \cdots \\ \cdots \\ attribute_n=value_n \end{pmatrix}$$

特征结构举例

- he：

$$\begin{pmatrix} cat=R \\ lex=he \\ per=3 \\ num=singular \\ case=subject \\ sem=animal \\ \cdots \\ \cdots \end{pmatrix}$$

- is：

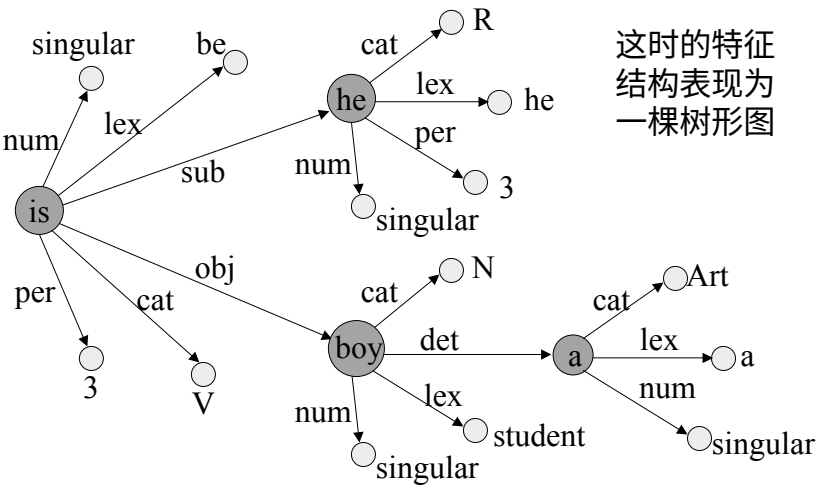
$$\begin{pmatrix} cat=V \\ lex=be \\ per=3 \\ num=singular \\ time=present \\ \cdots \\ \cdots \end{pmatrix}$$

特征结构的嵌套

- 特征结构中，某些特征的值可以是另一个特征结构
- 为区别于特征结构形式的特征值，我们把简单的字符串形式的特征值称为原子；
- 嵌套的特征结构可以表示更加复杂的语言单位，如短语和句子
- 例子：He is a boy

```
cat=V
lex=be
per=3
num=singular
…
sub=  [ cat=R
        lex=he
        per=3
        num=singular
        … ]
obj=  [ cat=N
        lex=boy
        num=singular
        …
        det= [ cat=Art
               lex=a
               num=singular
               … ] ]
```


特征结构表示为图形



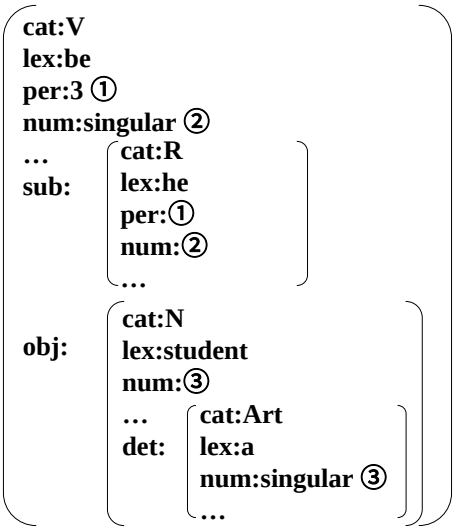
这时的特征结构表现为一棵树形图

特征结构的重入

- 重入（Reentrance）是特征结构的一个重要特点
- 重入表示在一个特征结构中，有两个特征共享同一个特征值
- 例如，在一个英语句子中，我们要求主语和谓语的人称特征共享同一个特征值，单复数特征也共享同一个特征值。

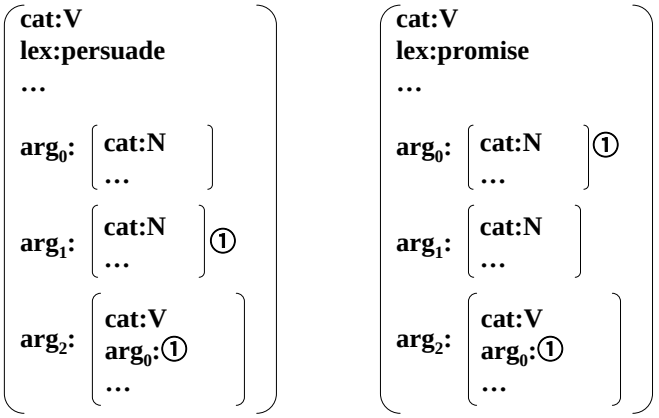
特征结构的重入举例

- 例子：
  He is a student.
- 在特征结构表示中，一般用带方框（这里用圆圈表示）的数字表示重入的特征结构
- 在重入的多个特征结构中，最多只能在一处说明其特征值

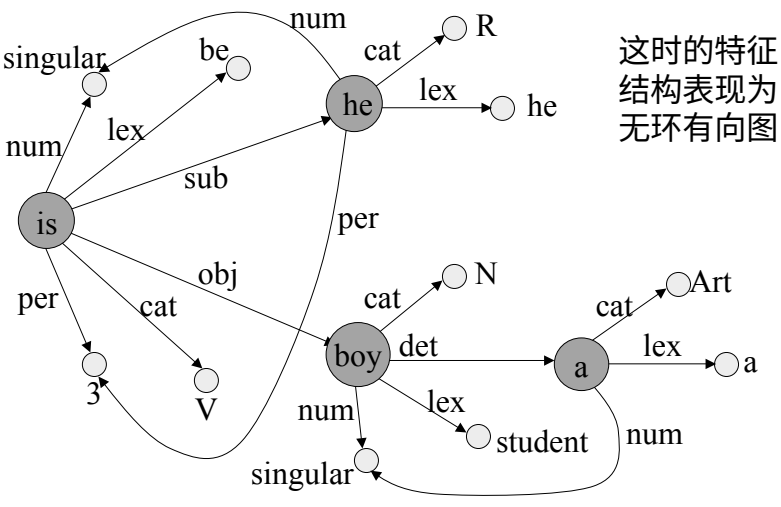


特征结构的重入举例（续）

- 英语Persuade和Promise的区别

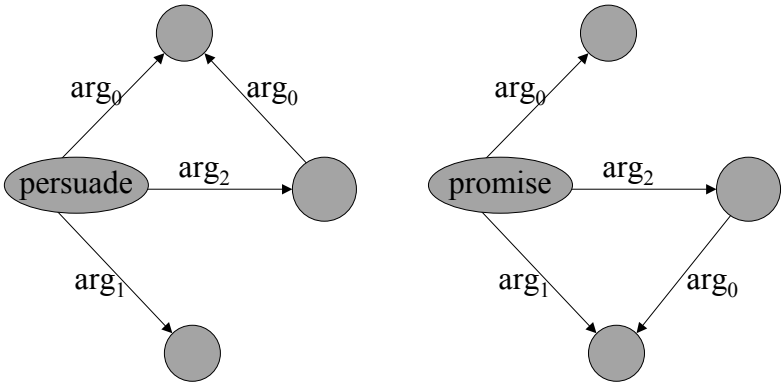


特征结构表示为图形（续）



这时的特征结构表现为无环有向图

特征结构表示为图形（再续）



合一运算 1

- 合一是定义在两个特征结构之间的一种操作；
- 合一的含义是：将两个独立的特征结构共享为一个单一的特征结构；
- 合一操作有成功或失败两种可能：
  - 合一成功，则原来的两个独立的特征结构成为同一个特征结构；
  - 合一失败，维持原状。
- 注意：合一失败和合一结果为空是不同的
  - 合一失败，两个特征结构之间不发生共享；
  - 合一结果为空，表示合一成功，两个特征结构共享，变成同一个特征结构，只是这个特征结构是空特征结构；
  - 只有两个空特征结构合一，结果才是空。

合一运算 2

- 定义：符号α(f)，表示复杂特征集α的名为f的特征的值。
- 定义：若α、β均为复杂特征集，f是任意特征名，则α、β是相容的递归定义如下：
  - 若α(f)=a、β(f)=b，a、b都是原子，α、β是相容的当且仅当a=b。
  - 若α(f)、β(f)均为复杂特征集，α、β相容当且仅当α(f)、β(f)相容。

合一运算 3

- 用符号$\overline{\cup}$表示合一运算，则合一运算可递归定义如下：
  - 如果a、b都是原子，若a=b，则a$\overline{\cup}$b=a，否则a、b合一失败
  - 如果α、β均为复杂特征集，则
    - 若α(f)=v，但β(f)的值未经定义，则<f,v>属于α$\overline{\cup}$β；
    - 若β(f)=v，但α(f)的值未经定义，则<f,v>属于α$\overline{\cup}$β；
    - 若α(f)=v_1，β(f)=v_2，且v_1与v_2相容，则<f,($v_1\overline{\cup}v_2$)>属于α$\overline{\cup}$β；否则α、β合一失败；

合一运算举例

$$\begin{bmatrix} \text{词性：动词} \\ \text{功能：谓语} \\ \text{结构：述宾} \\ \text{及物：是} \end{bmatrix} \bar{\cup} \begin{bmatrix} \text{词语：咳嗽} \\ \text{词性：动词} \\ \text{及物：否} \end{bmatrix} \rightarrow \text{失败}$$

$$\begin{bmatrix} \text{cat：V} \\ \text{lex：go} \\ \text{num：plural} \\ \text{Time：present} \end{bmatrix} \bar{\cup} \begin{bmatrix} \text{num：plural} \\ \text{time：past} \end{bmatrix} \rightarrow \text{失败}$$

合一运算举例（续）

$$A = \begin{bmatrix} \text{Agree:} & \begin{bmatrix} \text{Number: Singular} \end{bmatrix} \\ \text{Subject：} & \begin{bmatrix} \text{Agree：} & \begin{bmatrix} \text{Number: Singular} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$B = \begin{bmatrix} \text{Subject：} & \begin{bmatrix} \text{Agree：} & \begin{bmatrix} \text{Person: 3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$A \bar{\cup} B = \begin{bmatrix} \text{Agree:} & \begin{bmatrix} \text{Number: Singular} \end{bmatrix} \\ \text{Subject：} & \begin{bmatrix} \text{Agree：} & \begin{bmatrix} \text{Number: Singular} \\ \text{Person: 3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

合一运算举例（再续）

$$B = \begin{bmatrix} \text{Subject}: \begin{bmatrix} \text{Agree}: \begin{bmatrix} \text{Person: 3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$C = \begin{bmatrix} \text{Agree}: \begin{bmatrix} \text{Number: Singular} \end{bmatrix} \textcircled{1} \\ \text{Subject}: \begin{bmatrix} \text{Agree}: \textcircled{1} \end{bmatrix} \end{bmatrix}$$

$$B \bar{\cup} C = \begin{bmatrix} \text{Agree}: \begin{bmatrix} \text{Number: Singular} \\ \text{Person: 3} \end{bmatrix} \textcircled{1} \\ \text{Subject}: \begin{bmatrix} \text{Agree}: \textcircled{1} \end{bmatrix} \end{bmatrix}$$

合一运算的特点 1

- 交换律：$A \bar{\cup} B = B \bar{\cup} A$
- 结合律：$A \bar{\cup} (B \bar{\cup} C) = (A \bar{\cup} B) \bar{\cup} C$
  - 结合律是合一运算的一个重要特点
  - 满足结合律说明合一运算的执行顺序与结果无关
  - 合一运算的结合律使得特征结构真正成为一种“描述性”知识表示方法，而不是“过程性”的表示方法
  - “描述性”知识表示方法的含义在于，对于一个变量的约束和赋值是等同的，我们可以在对一个变量赋值之前就给出对它的约束，而不必等到对这个变量赋值之后才对它进行约束
  - 比如，我们可以在词典中指出，英语动词persuade的arg_3的arg_1必须和persuade的arg_2合一，虽然这时我们并不知道在具体的句子中persuade的各个arg是什么
  - 特征结构的“描述性”特点有利于在词典中给出词语的个性化描述

合一运算的特点 2

如果把自然语言看作是一个传递和负载信息的系统，并且承认自然语言中的句法成分和语义成分都可由较小的单位合成较大的单位，那么，采用合一运算作为句法语义分析的基本运算便是非常理想的了。这是因为：

- 一个语言单位所负载的信息可以分布在各个成分之中，每个成分所负载的可以只是部分信息。通过合一运算，在小成分组合成大成分的过程中，小成分所负载的信息也同时被传递和累加为大成分所负载的信息，信息只逐渐增加而不会减少。
- 由于句法和语义分析都以合一作为基本运算，不仅句子的合法性可以通过语义手段来判断，而且，还可以把句子的句法结构和语义表示用合一运算这种方式更加自然的衔接起来。
- 对不同的复杂特征集进行合一运算，其结果同运算所进行的先后次序无关，不论合一从那个方向开始，也不论是先合一还是后合一，合一的结果都是相同的。合一运算的这种无序性非常便于并行处理，而且还使我们有可能自由地选择分析算法和自然语言描述的语法理论。

非重入特征结构与伪合一 1

- 特征结构和合一运算具有强大的表达能力，不过由于要考虑重入问题，实现的算法较为复杂，时空复杂度也较高
- “非重入特征结构和伪合一”是对特征结构和合一运算的一种简化
- 在“非重入特征结构”中，特征结构中不允许重入，从图形表示来看，特征结构不是一个无环有向图，而是一棵树形图；
- 在“伪合一”中，两个特征结构合一成功后并不共享，而仍然是两个独立的特征结构

非重入特征结构与伪合一 2

- 非重入特征结构与伪合一的引入，使得知识表示不再具有描述性特点，合一的结果与合一操作的执行顺序有关
- 比如依次执行A$\bar{\cup}$B、A$\bar{\cup}$C、B$\bar{\cup}$D，假设A和B初始值都为空，C和D不相容，如果采用全合一运算，最终的结果将合一失败，如果采用伪合一运算，那么最终的结果不会失败，而是A=C，B=D
- 在伪合一情况下，如果我们在词典中指定英语动词persuade的arg_3的arg_1必须和persuade的arg_2合一就毫无意义
- 伪合一的实现算法相对简单，执行效率也较高，而且非重入的特征结构的表达能力至少也强于单一的范畴，因此这种知识表示方法也有一定的使用价值

功能合一语法 1

- 功能合一语法
  Functional Unification Grammar，FUG
- Martin Kay, 1979, *Functional Grammar*, In Proceedings of the 4th Annual Meeting of the Berkeley Linguistics Society.
- Martin Kay, 1985, *Parsing in Functional Grammar*, In D.Dowty, L. Karttunen, and A. Zwicky eds, Natural Language Parsing, Cambridge University Press, Cambridge, 1985.

功能合一语法－特点

- 弱化线性序结构关系：序列关系不严格要求
- 强调功能结构：采用复杂的特征结构进行描述
- 对所有语言单位统一采用FD形式描述：
  词条定义，句法规则、语义信息、以及句子的结构功能等，全部用复杂特征集来表示。也就是说，它试图以单一的形式结构模式来描述特征组合、功能分配、词条和组成成分的顺序等，达到对句子的完全功能描述。
- 既适合于分析，也适合于生成（Generation），在自然语言生成中有较多应用。

功能合一语法：功能描述 1

- 在FUG中，用于描述语言单位、词典和规则的特征结构被称为功能描述（Functional Description，FD）
- FD由一些特征（Feature）通过合取（conjunction）或者析取（disjunction）组成。用方括号[ ]表示合取，用花括号{ }表示析取。
- 每一个特征是一个属性－值对，记为：
  attribute＝value
- 合取相当于我们前面介绍的特征结构中的特征之间的关系，合取的各个特征的属性必须各不相同
- 析取是一种“或”的关系
- 可以对析取的结果进行合取，也可以对合取的结果进行析取

功能合一语法：功能描述 2

$$\begin{bmatrix} \text{Cat=S} \\ \text{Head}= \begin{bmatrix} \text{Subj}= \overset{①}{\begin{bmatrix} \text{Cat=NP} \\ \text{Num=Sing} \\ \begin{Bmatrix} \text{Case=Nom} \\ \text{Case=Acc} \end{Bmatrix} \end{bmatrix}} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \text{Cat=S} \\ \text{Object = […]} \\ \text{Topic = <Object>} \end{bmatrix}$$

- 路径（path）：是一些列属性组成的序列，如<Head Subj> 。
- 对于一个FD，可以通过一条路径，取得其内部某一个FD的值。如上左例中，路径<Head Subj>对应的值是FD①
- 如果沿该路径取值过程中遇到析取的FD，那么通过该路径最后取得的值也是析取的FD。
- 在FD内部，可以通过路径来表示重入的FD，如上右例，表示一个句子的话题就是该句子的宾语。

功能合一语法：功能描述 3

- 在FD中，特征的值有四种形式：
  - 原子
  - 成分集合（Feature Set）：是路径的集合。每一条路径对应于该短语的一个句法成分。成分集合用于表示一个语言单位中应该出现哪些句法成分
  - 模式（Pattern）：是由在成分集合中出现的路径组成的一个正则表达式，这个正则表达式规定了这些句法成分出现的顺序要受到怎样的制约。
  - 功能结构（FD）

功能合一语法：规则举例

$$\left\{\begin{array}{l}\left[\begin{array}{l}\text{Head} = [\ \text{Head} = [\ \text{Cat} = \text{V}\ ] \\ \text{CSet} = \{\ \langle\text{Head Head Subj}\rangle\ \langle\text{Head}\rangle\ \} \\ \text{Pat} = (\ \langle\text{Head Head Subj}\rangle\ \langle\text{Head}\rangle\ )\end{array}\right] \\ \left[\begin{array}{l}\text{Head} = \left[\begin{array}{l}\text{Cat} = \text{V} \\ \left\{\begin{array}{l}\text{Obj} = \text{NONE} \\ \text{Obj} = [\ \text{Cat} = \text{NP}\ ]\end{array}\right. \\ \text{CSet} = \text{NONE}\end{array}\right]\end{array}\right] \\ \left[\begin{array}{l}\text{Head} = \left[\begin{array}{l}\text{Cat} = \text{N} \\ \text{CSet} = \text{NONE}\end{array}\right]\end{array}\right]\end{array}\right\}$$

功能合一语法：词典举例

$$\left[\begin{array}{l}\text{Cat} = \text{V} \\ \text{Lex} = \text{eats} \\ \text{Tense} = \text{Pres} \\ \text{Subj} = \left[\begin{array}{l}\text{Pers} = 3 \\ \text{Num} = \text{Sing} \\ \text{Anim} = +\end{array}\right] \\ \text{Obj} = [\ \text{Cat} = \text{NP}\ ]\end{array}\right]$$

词汇功能语法

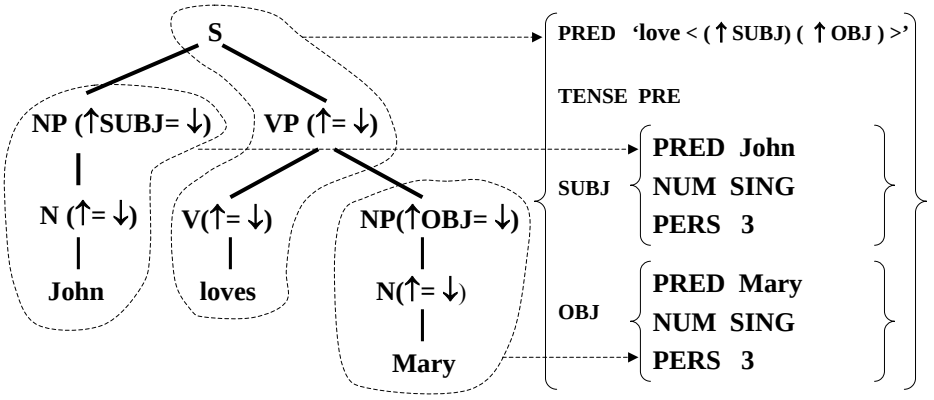
- 词汇功能语法
  Lexical Function Grammar，LFG
- Kaplan, R. and J.Bresnan, 1982, *Lexical Functional Grammar : A Formal System for Grammatical Representation*, In J.Bresnan ed. The Mental Representation of Grammatical Relations, MIT Press 1982.
- http://www-lfg.stanford.edu/lfg/bresnan/
- http://www.parc.xerox.com/istl/members/kaplan/
- http://csli-publications.stanford.edu/LFG/
  （每年一度的国际LFG学术会议电子论文集）

词汇功能语法－特点

- 以短语结构语法来构造句法树（成分结构，即c-结构），不使用转换规则和深层结构的概念；
- 以特征结构（功能结构，即f-结构）作为表达语法信息的主要手段；
- 以合一作为运算的基本方式；
- 以词汇中的信息作为语法信息的主要的来源。

词汇功能语法－两种结构

成分结构（c-结构）　　功能结构（f-结构）



词汇功能语法－规则

S→	NP	VP
	(↑SUBJ)= ↓	↑=↓
VP→	V	NP
	↑=↓	(↑OBJ)= ↓
VP→	V	
	↑=↓	
NP→	N	
	↑=↓	

↑表示父亲节点；
↓表示当前节点；
↑SUBJ 表示父亲节点的SUBJ特征；
＝表示合一

词汇功能语法－词典

John	**N**	**(↑PRED)='John'**
		(↑NUM)=SING
		(↑PERS)=3
Mary	**N**	**(↑PRED)='Mary'**
		(↑NUM)=SING
		(↑PERS)=3
loves	**V**	**(↑PRED)='love < (↑SUBJ), (↑OBJ) >'**
		(↑TENSE)=PRE
		(↑SUBJ NUM)=SING
		(↑SUBJ PERS)=3

词汇功能语法－句子的合法性

- 以动词为中心进行检查：
  - 一个论元只允许有一个值（唯一性）；
  - 每个论元都应该有值（完备性）；
  - 不该有的论元不应该有值（一致性）；

  不满足这三个条件的句子为不合法的句子
- 例如假设句子的中心动词为**loves**，其**PRED**特征的值为：

  (↑PRED)='love < (↑SUBJ), (↑OBJ) >'

  那么该句子的特征结构中必须有且只有SUBJ和OBJ这两个论元具有特征值

中心词驱动的短语结构语法

- 中心词驱动的短语结构语法
  Head-Driven Phrasal Structure Grammar, HPSG
- Pollard, Carl and Ivag A. Sag. 1987. *Information Based Syntax and Semantics*. CSLI Lecture Notes, No.13, The University of Chicago Press,Chicago.
- Pollard, Carl and Ivag A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press,Chicago.
- Sag, Ivan A. & Thomas Wasow, 1999, *Syntactic Theory: A Formal Introduction*, CSLI Publications, Stanford, California.
- http://ling.ohio-state.edu/research/hpsg/
- http://hpsg.stanford.edu/
- http://csli-publications.stanford.edu/LFG/
  （每年一度的国际HPSG学术会议电子论文集）

中心词驱动的短语结构语法 —— 特点

- 面向表层（surface oriented）

  不使用深层结构，没有空语类
- 基于约束（constraint based）

  整个构架是建立在约束机制上的。词项（lexical entry）描述、语法规则、语法原则等都是通过约束来实现的，并且约束的实现顺序是自由的（order-independent）。
- 词汇主义（strict lexicalism）

  语法规则的重担几乎全部转移到了词汇上，词汇承载了几乎所有的句法语义信息，是绝对的词汇主义。

中心词驱动的短语结构语法 —— 特征结构的表示

- 方括号 […] 表示特征结构，又称属性特征矩阵（attribute-value matrix AVM）
- 尖括号〈…〉表示特征结构的列表
- 标签（tag）①，②，③…等表示特征结构的共享（structure sharing）

注：一般HPSG的文献中标签的数字外面是方框而不是圆框，这里为排版方便起见采用圆框

中心词驱动的短语结构语法 —— 句法特征SYN

- SYN： SYN用来描述符号的句法信息。采用X-Bar理论的描述方法，提供对短语的HEAD（中心语）、COMP（补足语）、SPR（限定语）的约束

中心词驱动的短语结构语法 —— 语义特征SEM

- SEM：HPSG 主要借鉴了情景语义学（situation semantics）的研究成果，其语义描写的目的是想说明，在一个事件中，谁对谁做了什么（who did what to whom）、发生在什么时间、什么地点等。对一个符号的语义描写主要包括三个部分：MODE、INDEX、RESTR。MODE对应于我们常说的陈述、疑问、祈使和指称，INDEX表示一个事件中角色的编号，RESTR是事件成立必须满足的一些条件

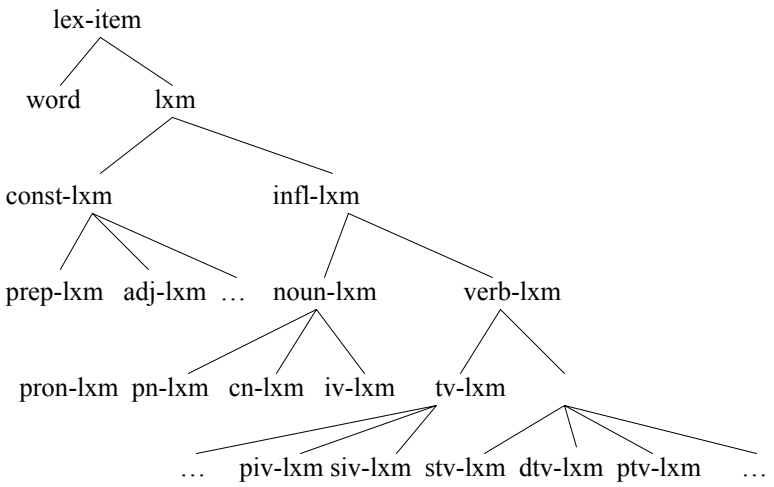
中心词驱动的短语结构语法 —— 论元结构特征AGR-ST

- AGR-ST：ARG-ST（ARGUMENT-STRUCTURE，论元结构） 是一个属性特征列表，它包含与动词共现的所有必需论元（arguments），是该动词的 SPR 和 COMPS 之和。例如 give 的ARG-ST 是 〈 NP_i ， NP_j ， NP_k 〉，意思是它需要一个主语和两个补足语（一个间接宾语和一个直接宾语）与其共现。在ARG-ST中，论元的排列次序和实际句子中的次序是相吻合的。NP_i 是 NP [SEM [INDEX i]] 的省写。NP_i 、NP_j 、NP_k 分别对应于 RESTR 中的 GIVER、GIVEN 和 GIFT，这样，give 的句法结构和语义特征就有机地联系起来了。

中心词驱动的短语结构语法 —— 类型化特征结构

- 类型化特征结构
  Typed （Sorted）Feature Structure
- 所谓类型化的特征结构，是指将特征结构分为一些类型（type），
  - 每一种类型的特征结构都必须具有一些指定的特征
  - 各种特征类型组成一个层级体系，下位类型集成上位类型的指定特征
  - 允许多重集成
- 通过类型化的特征结构，可以大大简化特征结构的描述

中心词驱动的短语结构语法 —— 词汇类型层次体系 1



中心词驱动的短语结构语法 —— 词汇类型层次体系 2

- lex-item (lexical-item) 是词汇项的总类
- word 指有词形变化的一个音义结合体，lxm 是不关注词形变化的一个词汇“家族”，是一个抽象的、静态的原型词(proto-word)如，walk, walks, walked 是不同的 word，但属于同一个 lxm。lxm是语言描写的出发点，word 是由lxm 演化而来的。
- 根据是否有形态变化，lxm分为infl-lxm (inflecting-lexeme) 和const-lxm (constant-lexeme)
- const-lxm进一步分成prep-lxm，adj-lxm 等
- infl-lxm分成noun-lxm和verb-lxm

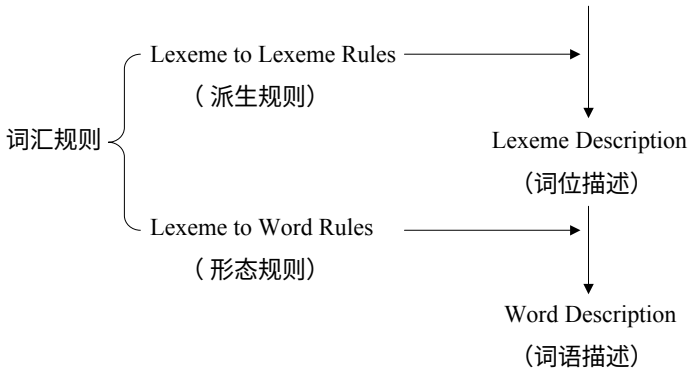
中心词驱动的短语结构语法 —— 词汇类型层次体系 3

- noun-lxm 进一步分成 pn-lxm(proper-noun-lexeme，专有名词)，cn-lxm (common-noun-lexeme，普通名词)和pron-lxm (pronoun-lexeme，代词)。
- verb-lxm 分成 iv-lxm(intransitive-verb-lexeme) 和 tv-lxm (transitive-verb-lexeme)。
- iv-lxm 进一步分成 siv-lxm (strict-intransitive-verb-lexeme)和 piv-lxm (prepositional- intransitive-verb-lexeme)。前者指不带任何宾语的不及物动词，如 “run”；后者指带介词结构宾语的不及物动词，如“rely” ，“I rely on you(我依靠你)”。
- tv-lxm 进一步分成 stv-lxm (strict-transitive-verb-lexeme，如 “devour”)，dtv-lxm(di transitive-verb-lexeme，如 give)，和 ptv-lxm (prepositional- transitive-verb-lexeme，如“put”)。

中心词驱动的短语结构语法 —— 词汇规则的运作

Basic Lexical Entry + Inherited Constraints = Lexeme Description

（词项基本信息） （词汇类体系结构运作） （词位描述）



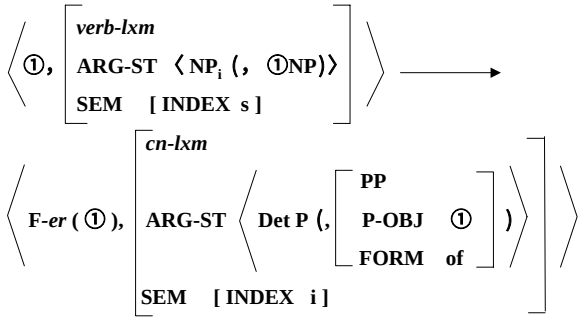
中心词驱动的短语结构语法 —— 形态规则举例

$$\left\langle ①, \begin{bmatrix} \textit{noun-lxm} \\ \textbf{ARG-ST} \ \langle [\textbf{COUNT} +] \rangle \end{bmatrix} \right\rangle \longrightarrow$$

$$\left\langle \mathbf{F}_{NPL}(①), \begin{bmatrix} \textit{word} \\ \textbf{SYN} \ \begin{bmatrix} \textbf{HEAD} \ \begin{bmatrix} \textbf{AGR} \ [\ \textbf{NUM} \ \ \textbf{pl}\ ] \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle$$

规则的输入是一个 *noun-lxm*，其指定语的特征是 [COUNT +] 。F_{NPL} 是一个形态函数，将一个名词由原形变成复数形式。输出的类由输入的 *noun-lxm* 变成了 *word*，并且具有属性 [NUM pl]。

中心词驱动的短语结构语法 —— 派生规则举例



动词名物化规则：ARG-ST 中的（ ）表示可选，“ARG-ST 〈 NP$_i$ (，①NP)〉”表示输入是一个及物动词或不及物动词，但不能是一个带介词结构做宾语的动词。“类”发生了变化，由输入的 *verb-lxm* 变成了 *cn-lxm* 。F-*er* 是一个形态函数，在动词词尾附加 “er”。输入动词的主语和输出名词具有相同的 INDEX （都是 i）。下面的例子可以帮助理解输出中的 PP：

He finds the key. → The finder of the key.

中心词驱动的短语结构语法 —— 规则与原则

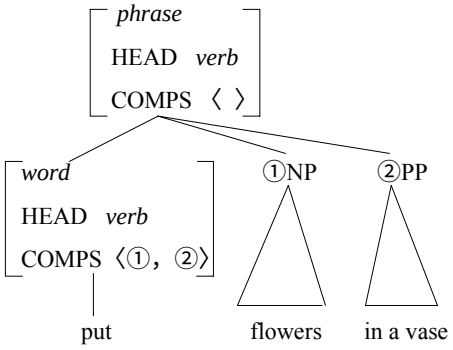
- 规则
  - 中心语－补足语规则
  - 中心语－限定语规则
  - 中心语－修饰语规则
  - 并列规则
- 原则
  - 中心语特征原则
  - 值传递原则
  - 语义承袭原则
  - 语义组合原则

中心词驱动的短语结构语法 —— 中心语－补足语规则 1

$$\begin{bmatrix} \textit{phrase} \\ \text{COMPS}\ \langle\ \rangle \end{bmatrix} \longrightarrow \text{H} \begin{bmatrix} \textit{word} \\ \text{COMPS}\ \langle ① \ldots\ldots ⓝ \rangle \end{bmatrix} \quad ① \ldots\ldots ⓝ$$

补足语就是X-Bar理论中的Complement，HPSG 用 COMPS 的属性来表示。COMPS 是一个属性特征（synsem）列表，其成分的排列次序和实际句子中的次序相吻合。中心语 — 补足语规则要求所有的补足语实现为中心语的姊妹节点。

中心词驱动的短语结构语法 —— 中心语－补足语规则 2



中心语－补足语规则的应用示例

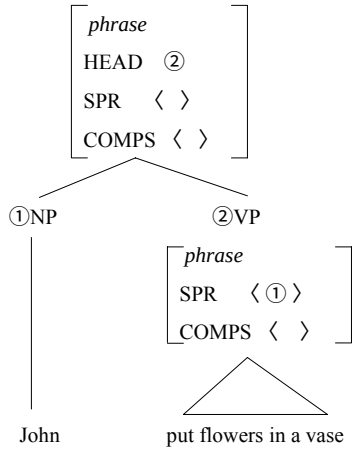
中心词驱动的短语结构语法
—— 中心语－限定语规则 1

$$\begin{bmatrix} \textit{phrase} \\ \text{COMPS} \ \langle\ \rangle \\ \text{SPR} \quad \langle\ \rangle \end{bmatrix} \longrightarrow ① \text{ H} \begin{bmatrix} \textit{phrase} \\ \text{SPR} \quad \langle\ ① \rangle \end{bmatrix}$$

限定语就是X-Bar理论中的Specifier，用 SPR 的属性来表示。同COMPS 一样，SPR 也是一个属性特征列表。对 VP而言，它的 SPR 是一个NP；对 NP 而言，它的 SPR 是 Det。

中心语总是先和补足语捆绑，再和限定语捆绑。

中心词驱动的短语结构语法
—— 中心语－限定语规则 2



中心语－限定语规则的应用示例

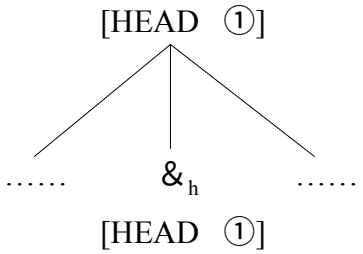
中心词驱动的短语结构语法
—— 中心语－修饰语规则

$$[\ phrase\ ] \longrightarrow \text{H}① \quad [\ phrase\ ] \quad \begin{bmatrix} phrase \\ \text{HEAD}\ [\ \text{MOD}\ ①\ ] \end{bmatrix}$$

修饰语指修饰中心语的成分，HPSG 用 MOD 的属性来表示修饰语的修饰功能，MOD 同样是一个属性特征列表。例如形容词的 MOD 属性是 [MOD NP]，副词是 [MOD VP]。

中心词驱动的短语结构语法
—— 中心语特征原则

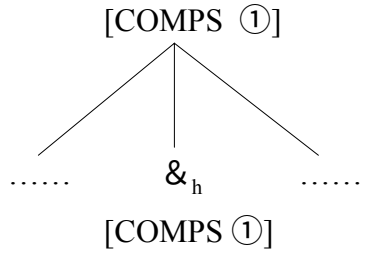
中语特征原则（Head Feature Principle HFP）：
任何一个中心语短语中（headed phrase），父结点的中心语特征值和中心语子结点的中心语特征值合一

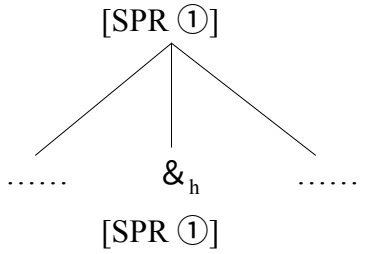


中心词驱动的短语结构语法 —— 值传递原则

值传递原则（Valence Principle）：

除非特别说明，父结点的 SPR 和 COMPS 的特征值和中心语子结点合一

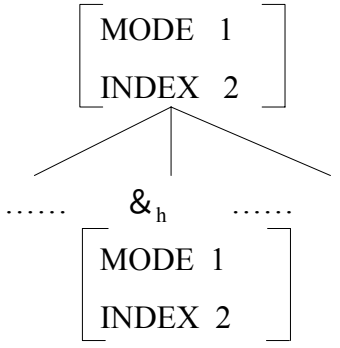




中心词驱动的短语结构语法 —— 语义承袭原则

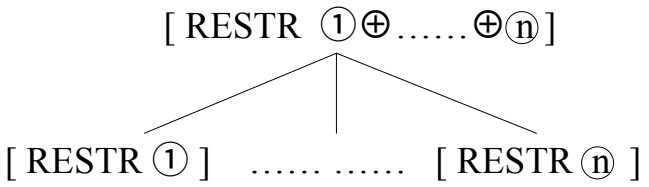
语义承袭原则（Semantic Inheritance Principle）

在一个中心语短语中，母亲的MODE和INDEX特征值和中心语子女同。



中心词驱动的短语结构语法 ——语义组合原则

语义组合原则（Semantic Compositionality Principle）在一个合式短语结构（well-formed phrase structure）中，母亲的 RESTR 值等于所有子女的 RESTR 值之和。



中心词驱动的短语结构语法 —— 起始符

HPSG 理论起始符的概念类似于PSG中的句子S，它是一个满足了一定约束条件的 *phrase*：

$$\begin{bmatrix} \textit{phrase} \\ \text{SYN} \begin{bmatrix} \text{HEAD} \begin{bmatrix} \textit{verb} \\ \text{FORM fin} \end{bmatrix} \\ \text{SPR} \quad \langle\ \rangle \\ \text{COMPS} \langle\ \rangle \end{bmatrix} \end{bmatrix}$$

首先，这个 *phrase* 的中心语是一个 *verb*, 它的形式是有定的（finite）。
其次，它是完全饱和的（saturated），SPR 和 COMPS的值都为空。

复习思考题

- 设想一下如何为基于特征结构与合一的语法体系建立概率模型？这种概率模型的最大问题是什么？
- 用程序实现一个特征结构的合一算法。如果特征结构中允许回路，会出现什么问题？
- 尝试用LFG和HPSG分析汉语句子：
  咬死了猎人的狗不见了
- 尝试用Prolog语言实现一部简单的汉语定子句语法，并用来分析至少10个汉语句子