10. Left-associative grammar (LAG)

10.1 Rule types and derivation order

10.1.1 The notion *left-associative*

When we combine operators to form expressions, the order in which the operators are to be applied may not be obvious. For example, a + b + c can be interpreted as ((a + b) + c) or as (a + (b + c)). We say that + is *left-associative* if operands are grouped left to right as in ((a + b) + c). We say it is *right-associative* if it groups operands in the opposite direction, as in (a + (b + c)).

A.V. Aho & J.D. Ullman 1977, p. 47

10.1.2 Incremental left- and right-associative derivation

left-associative:	*right-associative:*
a	a
(a + b)	(b + a)
((a + b) + c)	(c + (b + a))
(((a + b) + c) + d)	(d + (c + (b + a)))
...	...
$\Longrightarrow$	$\Longleftarrow$

10.1.3 Left-associative derivation order

Derivation is based on the principle of possible *continuations*

Used to model the time-linear structure of language

10.1.4 Irregular bracketing structures corresponding to the trees of C- and PS-grammar

```
(((a + b) + (c +d)) + e)
((a + b) + ((c +d)) + e)
(a + ((b + c)) + (d + e))
((a + (b + c)) + (d + e))
(((a + b) + c) + (d +e))
        ...
```

The number of these irregular bracketings grows exponentially with the length of the string and is infinite, if bracketings like (a), ((a)), (((a))), etc., are permitted.

10.1.5 Irregular bracketing structure

Derivation is based on the principle of possible *substitutions*

Used to model constituent structure

10.1.6 The principle of possible continuations

Beginning with the first word of the sentence, the grammar describes the possible continuations for each sentence start by specifying the rules which may perform the next grammatical composition (i.e., add the next word).

10.1.7 Schema of left-associative rule in LA-grammar

r_i: cat_1 cat_2 $\Rightarrow$ cat_3 rp_i

10.1.8 Schema of a canceling rule in C-grammar

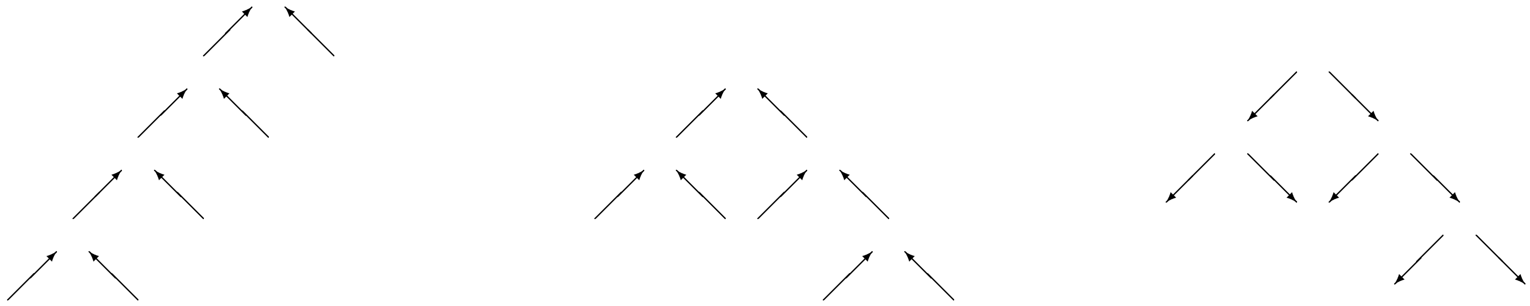
$$\alpha_{(Y|X)} \circ \beta_{(Y)} \Rightarrow \alpha\beta_{(X)}$$

10.1.9 Schema of a rewrite rule in PS-grammar

A $\rightarrow$ B C

10.1.10 Three conceptual derivation orders

LA-grammar C-grammar PS-grammar



bot.-up left-associative *bottom-up amalgamating* *top-down expanding*

10.2 Formalism of LA-grammar

10.2.1 Algebraic definition of LA-grammar

A left-associative grammar (or LA-grammar) is defined as a 7-tuple $<$W, C, LX, CO, RP, ST_S, ST_F $>$, where

1. W is a finite set of *word surfaces*.
2. C is a finite set of *category segments*.
3. LX $\subset$ (W $\times$ C^+) is a finite set comprising the *lexicon*.
4. CO = (co_0 ... co_{n-1}) is a finite sequence of total recursive functions from (C^* $\times$ C^+) into C^* $\cup$ $\{\perp\}$, called *categorial operations*.
5. RP = (rp_0 ... rp_{n-1}) is an equally long sequence of subsets of n, called *rule packages*.
6. ST_S = {(cat_s rp_s), ...} is a finite set of *initial states*, whereby each rp_s is a subset of n called start rule package and each cat_s ϵ C^+.
7. ST_F = {(cat_f rp_f), ...} is a finite set of *final states*, whereby each cat_f ϵ C^* and each rp_f ϵ RP.

10.2.2 A concrete LA-grammar is specified by

1. a lexicon LX (cf. 3),
2. a set of initial states ST_S (cf. 6),
3. a sequence of rules r_i, each defined as an ordered pair (co_i, rp_i), and
4. a set of final states ST_F.

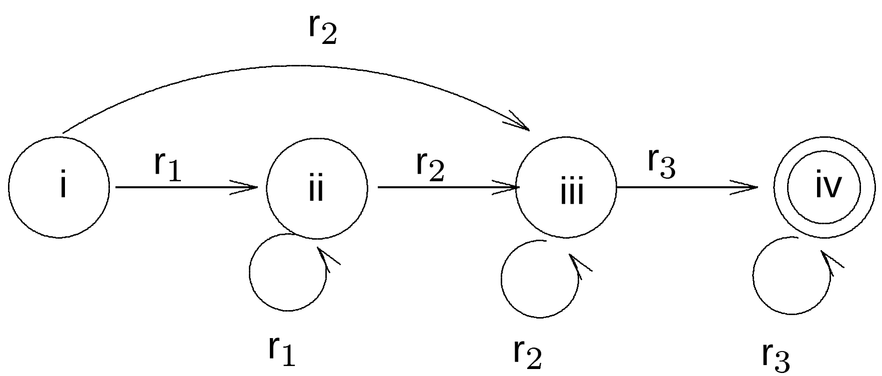
10.2.3 LA-grammar for $a^k b^k$

LX $=_{def}$ {[a (a)], [b (b)]}
$ST_S =_{def}$ {[(a) $\{r_1, r_2\}$]}
r_1: (X) (a) $\Rightarrow$ (aX) $\{r_1, r_2\}$
r_2: (aX) (b) $\Rightarrow$ (X) $\{r_2\}$
$ST_F =_{def}$ {[ε rp_2]}.

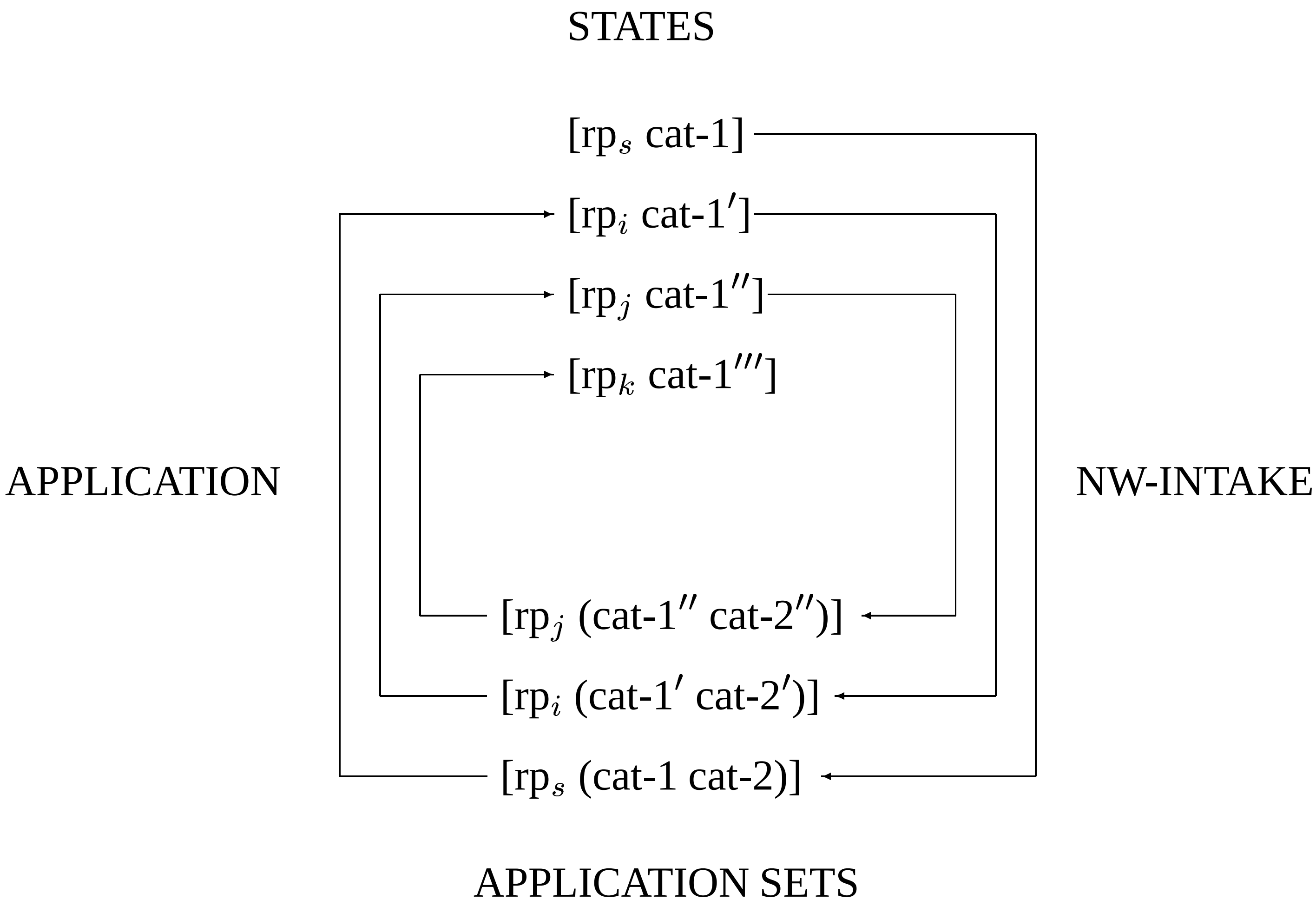
10.2.4 LA-grammar for $a^k b^k c^k$

LX $=_{def}$ {[a (a)], [b (b)], [c (c)]}
ST$_S$ $=_{def}$ {[(a) {r_1, r_2}]}
r_1: (X) (a) ⇒ (aX) {r_1, r_2}
r_2: (aX) (b) ⇒ (Xb) {r_2, r_3}
r_3: (bX) (c) ⇒ (X) {r_3}
ST$_F$ $=_{def}$ {[ε rp$_3$]}.

10.2.5 The finite state backbone of the LA-grammar for $a^k b^k c^k$



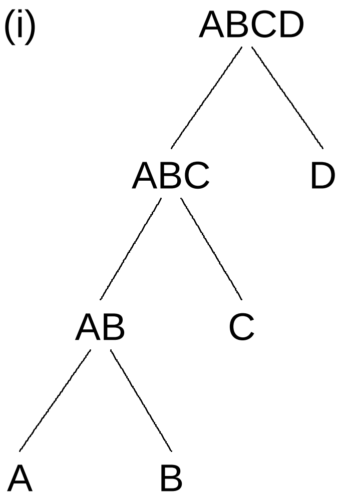
10.2.6 Recursion of left-associative algorithm



10.3 Time-linear analysis

10.3.1 LA-trees as structured lists

(i)



(ii)

ABCD
(D
ABC)
(C
AB)
(B
A)

(iii)

(A
B)
(AB
C)
(ABC
D)
ABCD

10.3.2 LA-grammar derivation of $a^k b^k$ for $k = 3$

```
NEWCAT>  a a a b b b
    *START-0
    1
       (A) A
       (A) A
    *RULE-1
    2
       (A A) A A
       (A) A
    *RULE-1
    3
       (A A A) A A A
       (B) B
    *RULE-2
    4
       (A A) A A A B
       (B) B
    *RULE-2
    5
       (A) A A A B B
       (B) B
    *RULE-2
    6
       (NIL) A A A B B B
```

10.3.3 Interpretation of a history section

```
active rule package:          *START-0
composition number:           1
sentence start:               (A) A
next word:                    (A) A
successful rule:             *RULE-1
next composition number:      2
result:                       (A A) A A
```

10.3.4 Overlap between history sections

```
active rule package:           *RULE-1
composition number:            2
sentence start :               (A A) A A
next word:                     (A) A
successful rule :              *RULE-1
next composition number:       3
result:                        (A A A) A A A
```

10.4 Absolute type transparency of LA-grammar

10.4.1 Parsing aaabbbccc with active rule counter

```
NEWCAT>  a a a b b b c c c
;  1: Applying rules (RULE-1 RULE-2)
;  2: Applying rules (RULE-1 RULE-2)
;  3: Applying rules (RULE-1 RULE-2)
;  4: Applying rules (RULE-2 RULE-3)
;  5: Applying rules (RULE-2 RULE-3)
;  6: Applying rules (RULE-2 RULE-3)
;  7: Applying rules (RULE-3)
;  8: Applying rules (RULE-3)
; Number of rule applications: 14.

    *START-0
    1
       (A) A
       (A) A
    *RULE-1
    2
       (A A) A A
       (A) A
    *RULE-1
    3
       (A A A) A A A
       (B) B
    *RULE-2
    4
       (A A B) A A A B
       (B) B
    *RULE-2
    5
       (A B B) A A A B B
       (B) B
    *RULE-2
    6
        (B B B) A A A B B B
        (C) C
    *RULE-3
    7
        (C C) A A A B B B C
        (C) C
    *RULE-3
    8
        (C) A A A B B B C C
        (C) C
    *RULE-3
    9
        (NIL) A A A B B B C C C
```

10.4.2 Generating a representative sample in $a^k b^k c^k$

```
NEWCAT> (gram-gen 3 '(a b c))

Parses of length 2:
 A B
   2   (B)
 A A
   1   (A A)

Parses of length 3:
 A B C
   2 3   (NIL)
 A A B
   1 2   (A B)
 A A A
   1 1   (A A A)

Parses of length 4:
 A A B B
   1 2 2   (B B)
 A A A B
   1 1 2   (A A B)
 A A A A
   1 1 1   (A A A A)

Parses of length 5:
 A A B B C
   1 2 2 3   (B)
 A A A B B
   1 1 2 2   (A B B)
A A A A B
  1 1 1 2   (A A A B)

Parses of length 6:
A A B B C C
  1 2 2 3 3   (NIL)
A A A B B B
  1 1 2 2 2   (B B B)
A A A A B B
  1 1 1 2 2   (A A B B)

Parses of length 7:
A A A B B B C
  1 1 2 2 2 3   (B B)
A A A A B B B
  1 1 1 2 2 2   (A B B B)

Parses of length 8:
A A A B B B C C
  1 1 2 2 2 3 3   (C)
A A A A B B B B
  1 1 1 2 2 2 2   (B B B B)

Parses of length 9:
A A A B B B C C C
```

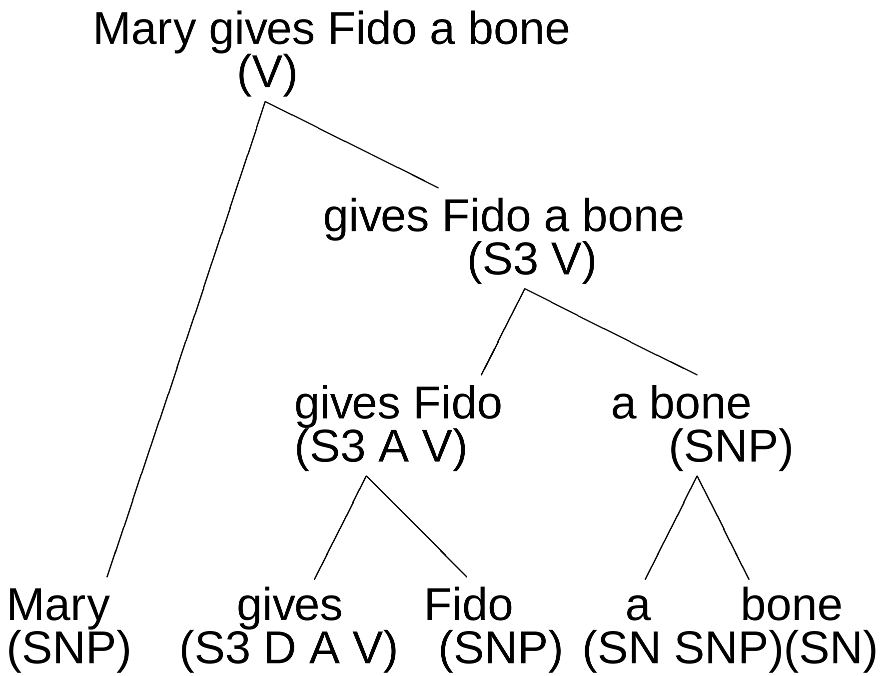
```
   1 1 2 2 2 3 3 3   (NIL)
 A A A A B B B B C
   1 1 1 2 2 2 2 3   (B B B)

Parses of length 10:
 A A A A B B B B C C
   1 1 1 2 2 2 2 3 3   (B B)

Parses of length 11:
 A A A A B B B B C C C
   1 1 1 2 2 2 2 3 3 3   (B)

Parses of length 12:
 A A A A B B B B C C C C
   1 1 1 2 2 2 2 3 3 3 3   (NIL)
```

10.4.3 Complete well-formed expression in $a^k b^k c^k$

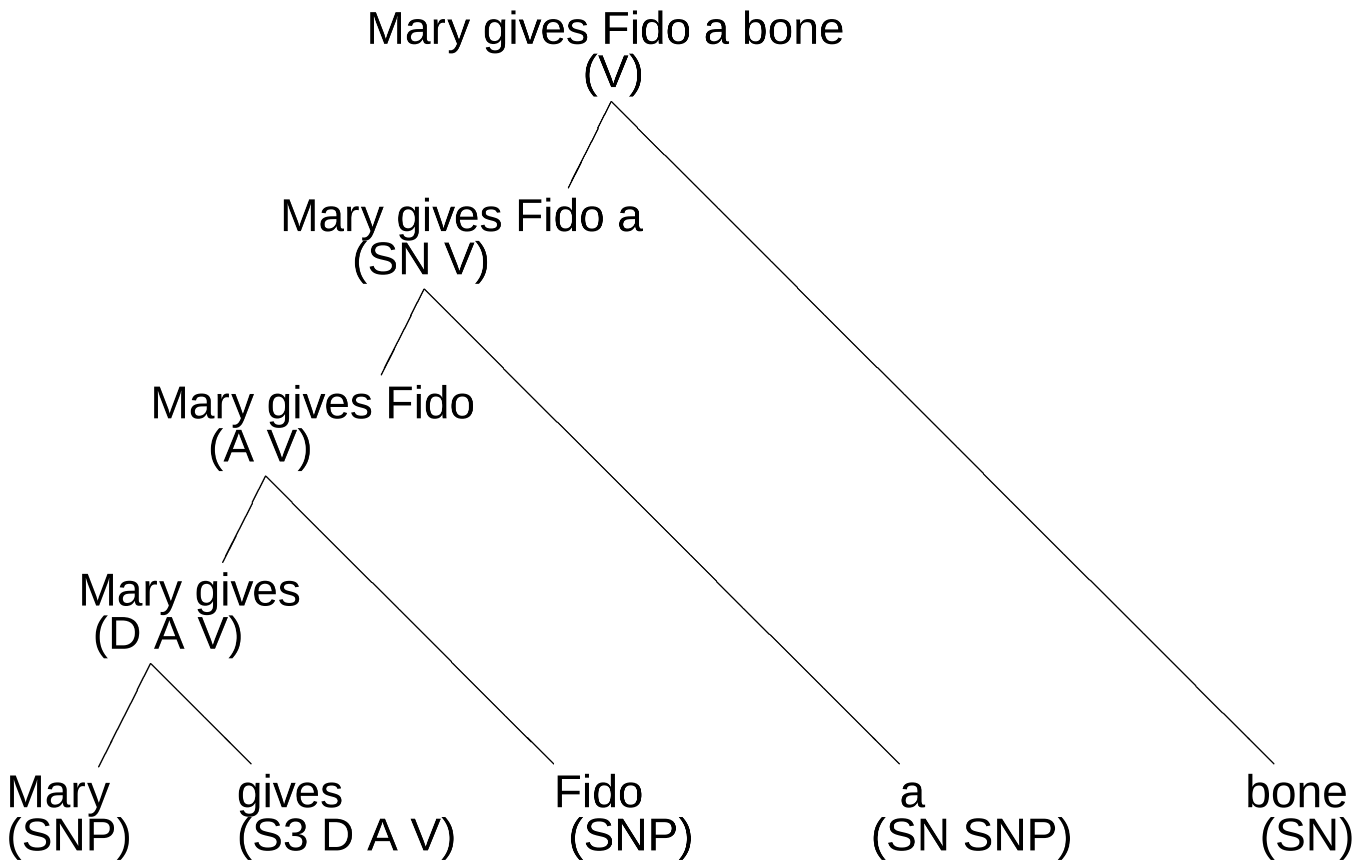
```
    A A A B B B C C C
      1 1 2 2 2 3 3 3   (NIL)
```

10.5 LA-grammar for natural language

10.5.1 Constituent structure analysis in C-grammar



10.5.2 Time-linear analysis in LA-grammar



10.5.3 Categorial operation combining Mary **and** gives

(SNP) (N D A V) ⇒ (D A V)

10.5.4 Categorial operation combining Mary gives **and** Fido

(D A V) (SNP) ⇒ (A V)

10.5.5 Categorial operation combining Mary gives Fido **and** a

(A V) (SN SNP) ⇒ (SN V)

10.5.6 Categorial operation combining Mary gives Fido a **and** book

(SN V) (SN) ⇒ (V)

10.5.7 Left-associative parsing of example 10.5.2

```
NEWCAT>  Mary gives Fido a bone \.

    *START
    1
       (SNP) MARY
       (S3 D A V) GIVES
    *NOM+FVERB
    2
       (D A V) MARY GIVES
       (SNP) FIDO
    *FVERB+MAIN
    3
       (A V) MARY GIVES FIDO
       (SN SNP) A
    *FVERB+MAIN
    4
       (SN V) MARY GIVES FIDO A
       (SN) BONE
    *DET+NOUN
    5
       (V) MARY GIVES FIDO A BONE
       (V DECL) .
    *CMPLT
    6
       (DECL) MARY GIVES FIDO A BONE .
```

10.5.8 Analysis of a discontinuous element

```
NEWCAT>  Fido dug the bone up \.

    *START
    1
       (SNP) FIDO
       (N A UP V) DUG
    *NOM+FVERB
    2
       (A UP V) FIDO DUG
       (SN SNP) THE
    *FVERB+MAIN
    3
       (SN UP V) FIDO DUG THE
       (SN) BONE
    *DET+NOUN
    4
       (UP V) FIDO DUG THE BONE
       (UP) UP
    *FVERB+MAIN
    5
       (V) FIDO DUG THE BONE UP
       (V DECL) .
    *CMPLT
    6
       (DECL) FIDO DUG THE BONE UP .
```

10.5.9 LA-analysis of ungrammatical input

```
NEWCAT>  the young girl give Fido the bone \.

ERROR
Ungrammatical continuation at: "GIVE"

    *START
    1
       (SN SNP) THE
       (ADJ) YOUNG
    *DET+ADJ
    2
       (SN SNP) THE YOUNG
       (SN) GIRL
     *DET+NOUN
    3
       (SNP) THE YOUNG GIRL
```