

10. Linksassoziative Grammatik (LAG)

10.1 Regeltypen und Ableitungsordnung

10.1.1 Der Begriff *linksassoziativ*

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 $(a + (b + c))$.

[Wenn wir Operatoren kombinieren, um Ausdrücke zu bilden, ist die Ordnung, in der die Operatoren anzuwenden sind, nicht immer offensichtlich. Zum Beispiel kann $a + b + c$ interpretiert werden als $((a + b) + c)$ oder als $(a + (b + c))$. Wir sagen, daß $+$ linksassoziativ ist, wenn die Teilausdrücke von links nach rechts in der Form der folgenden Klammerung erscheinen: $((a + b) + c)$. Wir nennen einen Operator rechtsassoziativ, wenn er die Teilausdrücke in der umgekehrten Richtung gruppiert: $(a + (b + c))$.]

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

10.1.2 Inkrementelle links- oder rechts-assoziative Ableitung

linksassoziativ:

$$\begin{aligned}
 & a \\
 & (a + b) \\
 & ((a + b) + c) \\
 & (((a + b) + c) + d) \\
 & \dots
 \end{aligned}
 \implies$$

rechtsassoziativ:

$$\begin{aligned}
 & a \\
 & (b + a) \\
 & (c + (b + a)) \\
 & (d + (c + (b + a))) \\
 & \dots
 \end{aligned}
 \impliedby$$

10.1.3 Linksassoziative Ableitungsordnung

Ableitung basiert auf dem Prinzip der möglichen *Fortsetzungen*. Modelliert die zeitlineare Struktur der Sprache.

10.1.4 Unregelmäßige Klammerstrukturen, die den Bäumen der C- und PS-Grammatik entsprechen

$$\begin{aligned}
 & (((a + b) + (c + d)) + e) \\
 & ((a + b) + ((c + d) + e)) \\
 & (a + ((b + c) + (d + e))) \\
 & ((a + (b + c)) + (d + e)) \\
 & (((a + b) + c) + (d + e)) \\
 & \dots
 \end{aligned}$$

Die Zahl der unregelmäßigen Klammerstrukturen wächst exponentiell mit der Länge der Kette – oder ist bei Klammerungen wie (a), ((a)), (((a))) etc. sogar unendlich.

10.1.5 Unregelmäßige Klammerstrukturen

Ableitung basiert auf dem Prinzip der möglichen *Substitutionen*. Modelliert Konstituentenstrukturen.

10.1.6 Das Prinzip der möglichen Fortsetzungen

Beginnend mit dem ersten Wort des Satzes beschreibt die LAGrammatik für jeden wohlgeformten Satzanfang die möglichen Fortsetzungen, indem sie die Regeln angibt, die für die jeweils nächste Komposition (d. h. das Anhängen des nächsten Wortes) grammatisch möglich sind.

10.1.7 Schema of left-associative rule in LA-grammar

$$r_i: \text{cat}_1 \text{ cat}_2 \Rightarrow \text{cat}_3 \text{ rp}_i$$

10.1.8 Schema of a canceling rule in C-grammar

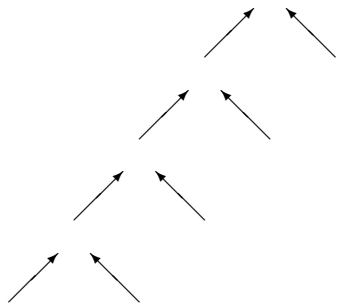
$$\alpha_{(Y|X)} \circ \beta_{(Y)} \Rightarrow \alpha\beta_{(X)}$$

10.1.9 Schema of a rewrite rule in PS-grammar

$$A \rightarrow B C$$

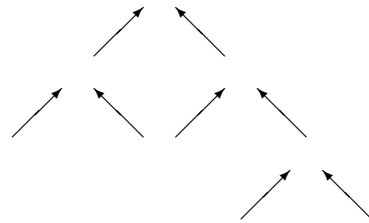
10.1.10 Three conceptual derivation orders

LA-Grammatik



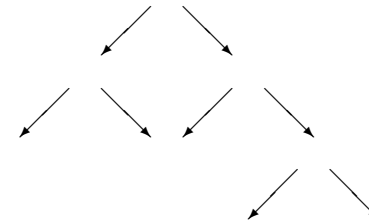
bot.-up left-associative

C-Grammatik



bottom-up amalgamating

PS-Grammatik



top-down expanding

10.2 Formalismus der LA-Grammatik

10.2.1 Algebraische Definition der LA-Grammatik

Eine LA-Grammatik ist als 7-Tupel $\langle W, C, LX, CO, RP, ST_S, ST_F \rangle$ definiert.

Dabei gilt

1. W ist eine endliche Menge von *Wortformoberflächen*.
2. C ist eine endliche Menge von *Kategoriesegmenten*.
3. $LX \subset (W \times C^+)$ ist eine endliche Menge, die das *Lexikon* umfaßt.
4. $CO = (co_0 \dots co_{n-1})$ ist eine endliche Sequenz aus total rekursiven Funktionen aus $(C^* \times C^+)$ in $C^* \cup \{\perp\}$, genannt *kategoriale Operationen*.
5. $RP = (rp_0 \dots rp_{n-1})$ ist eine ebenso lange Sequenz von Untermengen von n , genannt *Regelpakete*.
6. $ST_S = \{(cat_s \ rp_s), \dots\}$ ist eine endliche Menge von *Anfangszuständen*, wobei jedes rp_s , genannt *Anfangsregelpaket*, eine Untergruppe von n ist und jedes $cat_s \in C^+$.
7. $ST_F = \{(cat_f \ rp_f), \dots\}$ ist eine endliche Menge von *Endzuständen*, wobei jedes $cat_f \in C^*$ und jedes $rp_f \in RP$.

10.2.2 Eine konkrete LA-Grammatik wird spezifiziert durch

1. ein Lexikon LX (siehe 3),
2. eine Menge von Anfangszuständen ST_S (siehe 6),
3. eine Sequenz von Regeln r_i , definiert als (co_i, rp_i) , (siehe 4 und 5) und
4. eine Menge von Endzuständen ST_F (siehe 7)

10.2.3 LA-Grammatik für $a^k b^k$

$$LX =_{def} \{[a(a)], [b(b)]\}$$

$$ST_S =_{def} \{[(a) \{r_1, r_2\}]\}$$

$$r_1: (X) (a) \Rightarrow (aX) \{r_1, r_2\}$$

$$r_2: (aX) (b) \Rightarrow (X) \{r_2\}$$

$$ST_F =_{def} \{[\varepsilon rp_2]\}.$$

10.2.4 LA-Grammatik für $a^k b^k c^k$

$LX =_{def} \{[a(a)], [b(b)], [c(c)]\}$

$ST_S =_{def} \{[(a) \{r_1, r_2\}]\}$

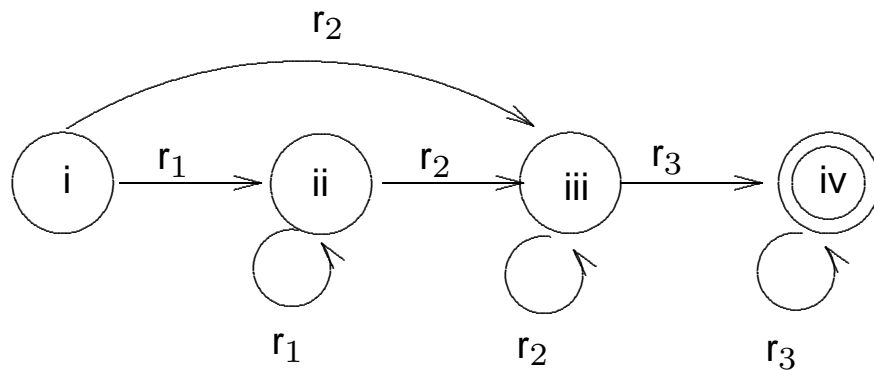
$r_1: (X) (a) \Rightarrow (aX) \{r_1, r_2\}$

$r_2: (aX) (b) \Rightarrow (Xb) \{r_2, r_3\}$

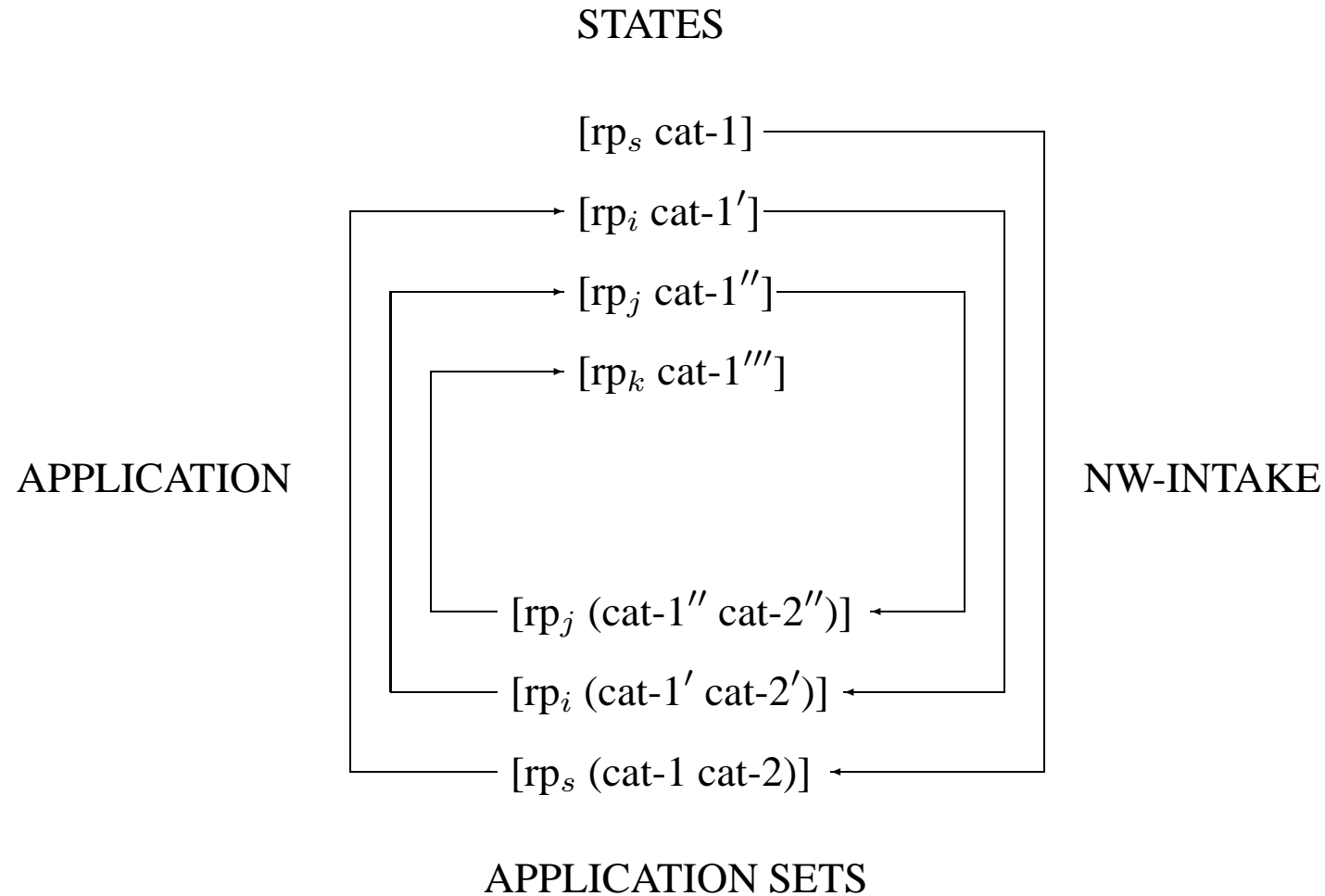
$r_3: (bX) (c) \Rightarrow (X) \{r_3\}$

$ST_F =_{def} \{[\varepsilon rp_3]\}$.

10.2.5 Das *finite-state*-Grundgerüst von LA- $a^k b^k c^k$

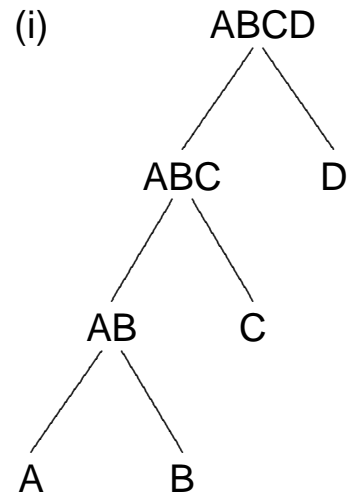


10.2.6 Rekursion des linksassoziativen Algorithmus



10.3 Zeitlineare Analyse

10.3.1 LA-Bäume als strukturierte Listen



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

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

10.3.2 LA-grammatische Ableitung von $a^k b^k$ 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 einer ‘history section’

aktives Regelpaket:	*START-0
Kompositionsnummer:	1
Satzanfang:	(A) A
nächstes Wort:	(A) A
Erfolgreiche Regel	*RULE-1
nächste Kompositionsnummer:	2
Ergebnis:	(A A) A A

10.3.4 Überlappung zwischen ‘history sections’

aktives Regelpaket:	*RULE-1
Kompositionsnummer:	2
Satzanfang:	(A A) A A
nächstes Wort:	(A) A
Erfolgreiche Regel:	*RULE-1
nächste Kompositionsnummer:	3
Ergebnis:	(A A A) A A A

10.4 Absolute Typentransparenz der LA-Grammatik

10.4.1 Parsen von aaabbbccc mit aktivem Regelzähler

```

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

*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

```

10.4.2 Erzeugung ‘Charakteristischer Beispiele’ 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:

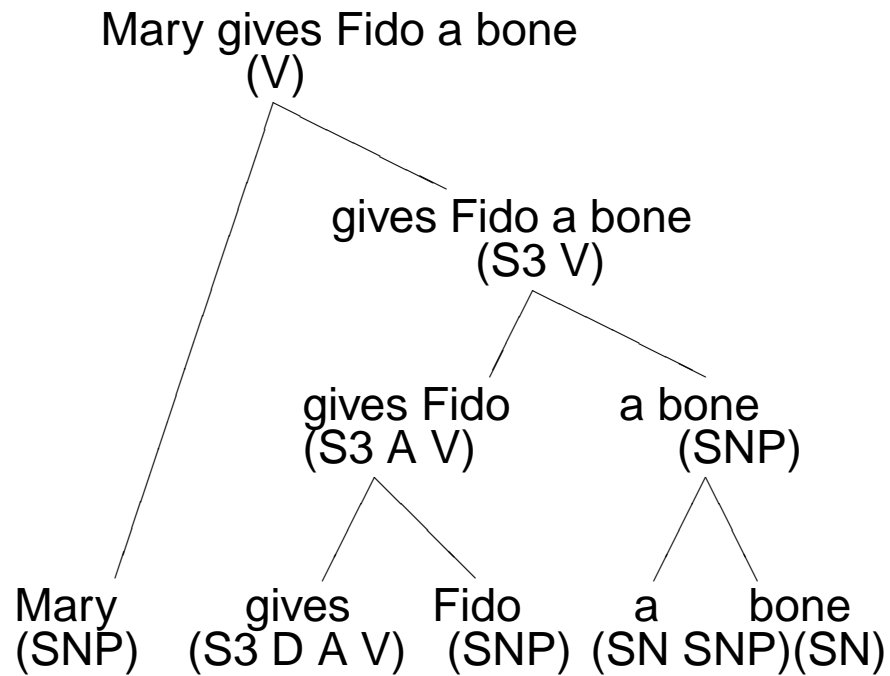
A A A A B B B B C C C C
 1 1 1 2 2 2 2 3 3 3 3 (NIL)

10.4.3 Vollständiger wohlgeformter Ausdruck in $a^k b^k c^k$

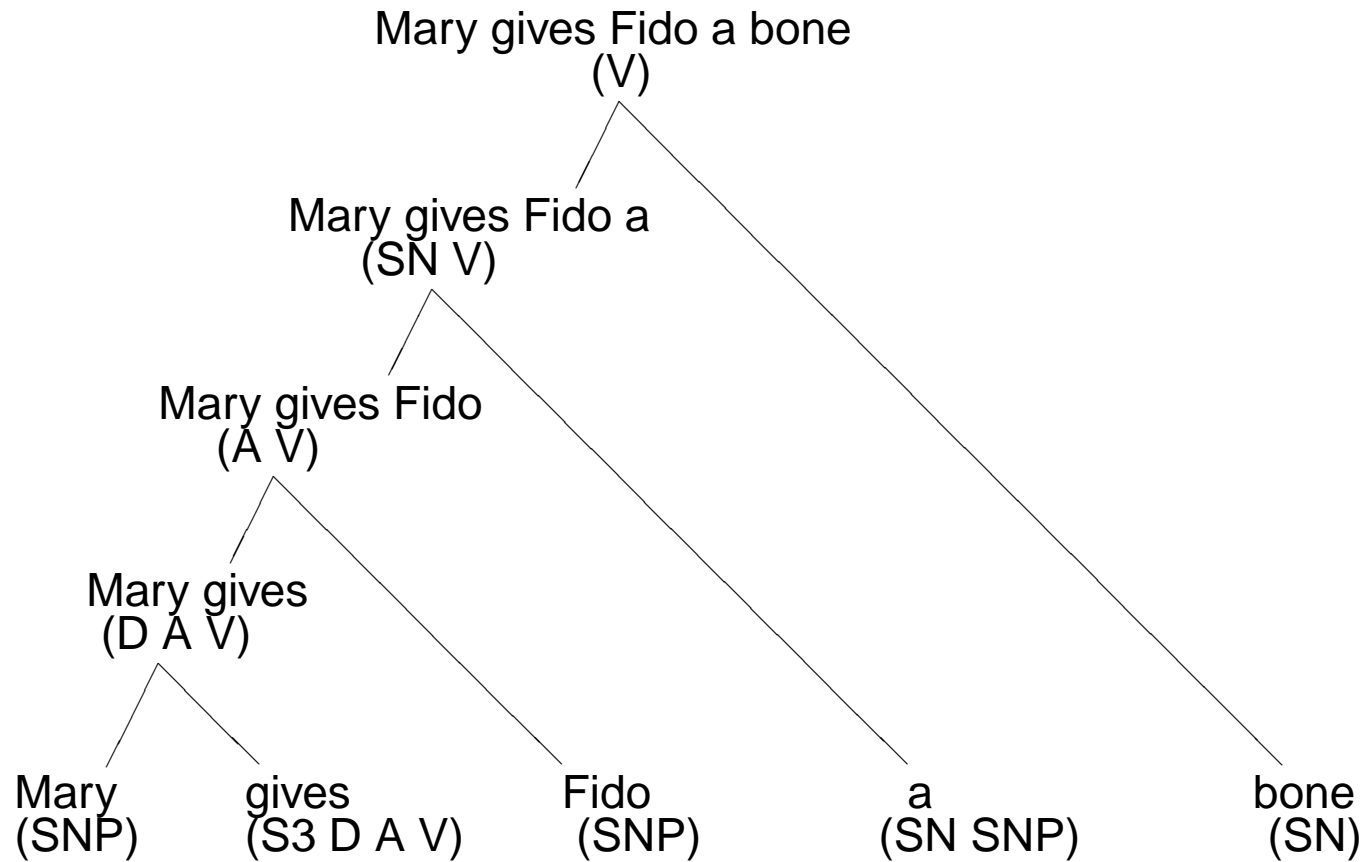
A A A B B B C C C
 1 1 2 2 2 3 3 3 (NIL)

10.5 LA-Grammatik für natürliche Sprachen

10.5.1 Konstituentenstrukturanalyse in der C-Grammatik



10.5.2 Zeitlineare Analyse in der LA-Grammatik



10.5.3 Kategoriale Operation der Komposition von Mary und gives $(\text{SNP}) (\text{N D A V}) \Rightarrow (\text{D A V})$ **10.5.4 Kategoriale Operation der Komposition von Mary gives and Fido** $(\text{D A V}) (\text{SNP}) \Rightarrow (\text{A V})$ **10.5.5 Kategoriale Operation der Komposition von Mary gives Fido and a** $(\text{A V}) (\text{SN SNP}) \Rightarrow (\text{SN V})$ **10.5.6 Kategoriale Operation der Komposition von Mary gives Fido a and book** $(\text{SN V}) (\text{SN}) \Rightarrow (\text{V})$

10.5.7 Automatische Parsinganalyse von Beispiel 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 Analyse eines diskontinuierlichen Elements

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 Analyse einer ungrammatischen Eingabe

```
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
```