

# Einführung in die Computerlinguistik und Sprachtechnologie

Vorlesung im WS 2009/10

Prof. Dr. Udo Hahn

Lehrstuhl für Computerlinguistik  
Institut für Germanistische Sprachwissenschaft  
Friedrich-Schiller-Universität Jena

# Phonologisch- Graphematische Analyse

- Schreibfehler
  - **F**eler      **A**uslassung
  - **F**ehler      **E**infügung
  - **F**whler      **E**rsetzung
  - **F**heler      **V**ertauschung
- Namensähnlichkeiten
  - **M**eier, **M**eyer, **M**aier, **M**ayer, **M**ayr, ...
  - Al-**d**schasira, Al-**d**schazirah, Al **J**azeera,
  - **C**ondol**e**ezza (1,96M), **C**ondol**e**eza (2,15M), **C**ondol**e**za (2,05M), **C**ondol**e**esa (3,6K), **C**ondol**e**sa (0,9K), **C**ondol**i**za (13K), **C**ondol**i**sa (0,8K), **C**ondol**i**ssa (0,3K) **C**ondol**e**ezza (12K), **K**ondol**e**ezza (0,1K), **K**ondol**i**sa, (0,8K) Rice [Google]
- Silbentrennung

2

## Silbentrennung

- Ziel: Zerlegung eines Wortes in seine Silbenbestandteile entsprechend den Regeln der deutschen Silbentrennung
  - Einfache Wörter
    - Spra-che, Sil-be, tren-nen, Wor-tes, bes-te
  - Zusammengesetzte Wörter
    - Vor-silbe, Silben-trennung, ab-ge-trennt
    - Silben-trennungs-programm,
    - Recht-schreib-prüfungs-klausur
    - Wort-ungetüm (nicht: Wortun-getüm)

3

## Illustration des Silbentrennungsproblems

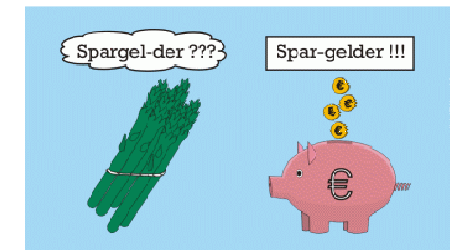
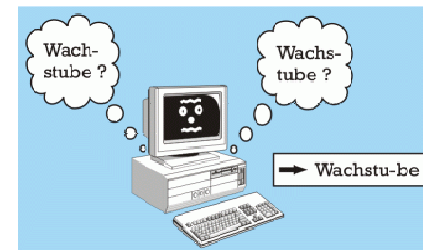
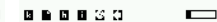
### Palm-Applikation: ohne Silbentrennung

SISIS ist ein Verfahren zur automatischen Silbentrennung deutscher Wörter nach alter oder neuer Rechtschreibung, das an unserem Institut entwickelt wurde. Es ist sicher und sinnentsprechend, da unsichere Trennstellen erkannt und unterdrückt



### Palm-Applikation: mit Silbentrennung

SISIS ist ein Verfahren zur automatischen Silbentrennung deutscher Wörter nach alter oder neuer Rechtschreibung, das an unserem Institut entwickelt wurde. Es ist sicher und sinnentsprechend, da unsichere Trennstellen erkannt und unterdrückt werden und sinnent-



## Bestandteile der Problemlösung

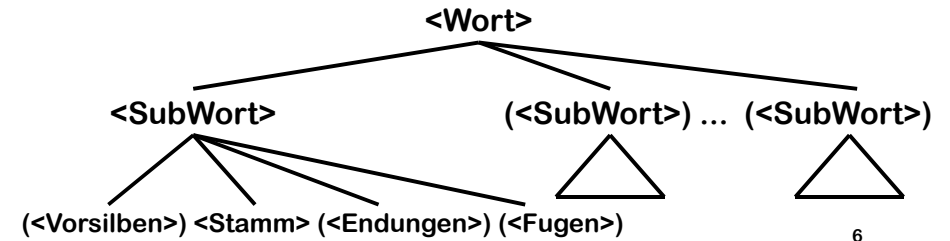
- Linguistisches Wissen (deklarativ)
  - Morphologische Struktur von Wörtern
  - Graphematische Trennungsregeln für einfache und zusammengesetzte Wörter
  - Lexikon
- Computerlinguistisches Wissen
  - Silbentrennungsalgorithmus (prozedural)

5

## Linguistisches Wissen: Morphologische Struktur von Wörtern

### BNF-Darstellung

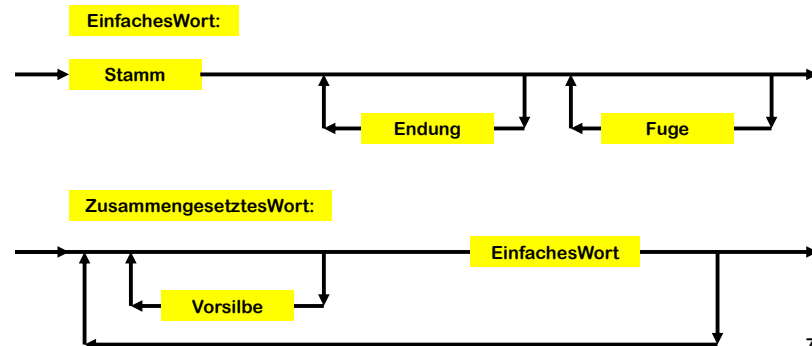
```
<Wort> ::= <SubWort> <SubWort>*
<SubWort> ::= <Vorsilbe>* <Stamm> <Endung>* <Fuge>*
```



6

## Linguistisches Wissen: Morphologische Struktur von Wörtern

### Syntax-Diagramme



7

## Linguistisches Wissen: Graphematische Trennungsregeln

### Regel 1 (Konsonanten)

1. Innerhalb einer Konsonantenfolge wird vor dem letzten Konsonanten bzw. bei einem einzelnen vor diesem getrennt.
  - Bsp.: Hal-le, Kat-ze, Re-ga-le, ...
2. **CH, SCH, PH, TH** werden als ein Konsonant gewertet.
  - Bsp.: Ver-wandt-schaft, ...

8

## Linguistisches Wissen: Graphematische Trennungsregeln

### Regel 2 (Vokale)

1. Zwischen zwei Vokalen darf getrennt werden.
  - Bsp.: Ri-tu-al, ak-tu-ell, ...
2. Zwischen zwei gleichen Vokalen und den Folgen **IE, EI, AU, ÄU, EU** wird nicht getrennt.
  - Bsp.: Bau-er, Waa-ge, Wie-se, nicht: Se-en, Fre-u-de, ...
3. Zwei Vokale, auf die ein Konsonant und ein Vokal folgen, werden nicht getrennt; Trennstelle ist dann der Konsonant.
  - Bsp.: böige → böi-ge (statt bö-ige), ...

9

## Linguistisches Wissen: Graphematische Trennungsregeln

### Regel 3

1. Beim Trennen nach Regel 1 und 2 müssen vor und hinter der Trennstelle Bestandteile entstehen, die mindestens einen Vokal enthalten.
  - Bsp.: Sport-ler, nicht: Sportle-r, nicht: fal-sch, ...
2. Ein (Doppel-)Vokal darf nicht allein abgetrennt werden.
  - Bsp.: De-kolle-tee, nicht: De-kollet-ee, S-ee...

10

## Linguistisches Wissen: Graphematische Trennungsregeln

### Regel 4

1. Regeln 1 bis 3 beschreiben die Trennung einfacher Wörter. Die Trennung zusammengesetzter Wörter folgt den Syntax-Diagrammen.
  - Bsp: un-ver-letzt, an-ge-hei-tert, ...
2. Ausnahmen für die Trennung einfacher Wörter werden im Lexikon kodiert.
  - Bsp.: Pro-gramm statt Prog-ramm,
  - Ka-ta-stro-phe statt Ka-tast-ro-phe

11

## Linguistisches Wissen: Lexikon

Menge von Einträgen der Form  
**<Wortfragment>**, **<Typ>**, **<Trennung>**  
wobei

– **<Wortfragment>**

- ein Fragment der Wortform

– **<Typ>**

- Vorsilbe, Stamm, Endung, Fuge

– **<Trennung>**

- explizite Angabe einer Sondertrennung

lauf, **STAMM**, —

lief, **STAMM**, —

be, **VORSILBE**, —

lich, **ENDUNG**, —

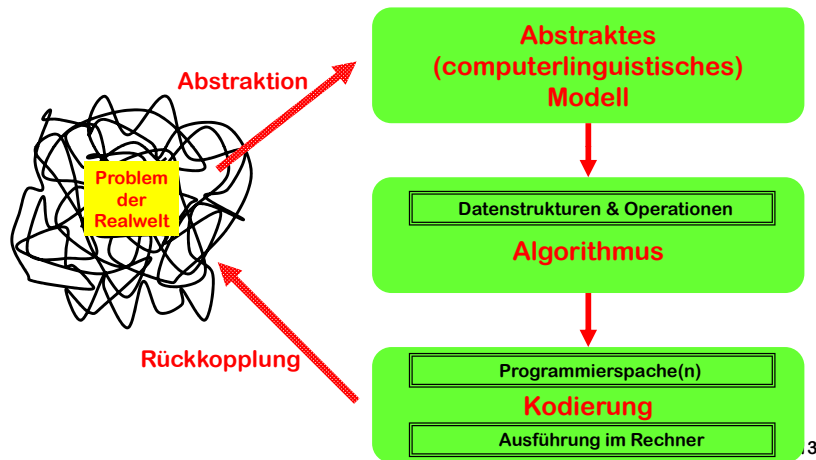
e, **ENDUNG**, —

Programm, **STAMM**,

Pro-gramm

12

## Informatischer Problemlösungszyklus



## Informatischer Problemlösungszyklus

- **Modellbildung**
  - **Abstraktion** von allen unwesentlichen Details der Problemstellung im Hinblick auf die algorithmische Lösung
  - Spezifikation der **logischen** Abhängigkeiten zwischen problemlösungsrelevanten Objekten
  - **CL: linguistisches Wissen**

14

## Informatischer Problemlösungszyklus

- **Algorithmisierung**
  - Übersetzung der modellbezogenen Spezifikation in
    - eine Menge von **Objekten** (Datenstrukturen) mit bestimmten Eigenschaften und Beziehungen zueinander
    - die erlaubten **Operationen** auf diesen Objekten
  - **Algorithmus**: (möglichst präzise) Beschreibung einer Folge zulässiger Operationen auf den Objekten, um das Problem zu lösen

15

## Informatischer Problemlösungszyklus

- **Kodierung (Programmierung)**
  - Übersetzung der algorithmischen Spezifikation in die Konstrukte und Syntax einer (geeigneten) Programmiersprache
- **Ausführung des Programms**
  - Hier erst Bezug auf konkrete Maschinen (Datenstrukturen und Algorithmen sind abstrakte Konstruktionen)
  - Test, Modifikation, Test, Modifikation, ...<sub>16</sub>

# Algorithmische Sprachkonstrukte

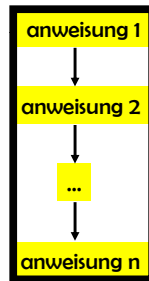
## Anweisungsfolge

PSEUDOCODE

FLUSSDIAGRAMM

STRUKTOGRAMM

anweisung 1;  
anweisung 2;  
...  
...  
anweisung n;



17

# Algorithmische Sprachkonstrukte

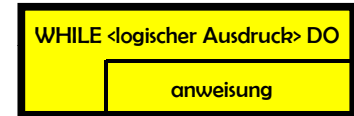
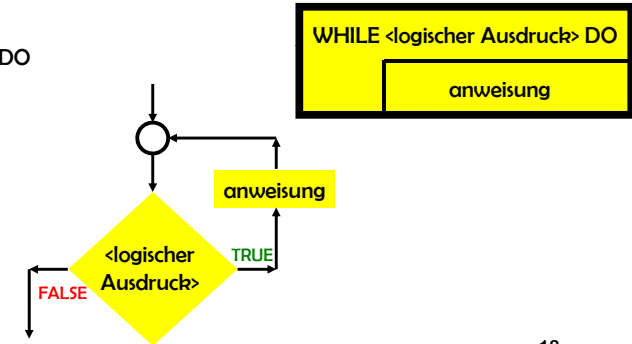
## Repetierte Anweisungen (WHILE)

PSEUDOCODE

FLUSSDIAGRAMM

STRUKTOGRAMM

WHILE <logischer Ausdruck> DO  
anweisung;



18

# Algorithmische Sprachkonstrukte

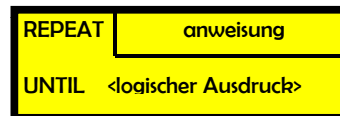
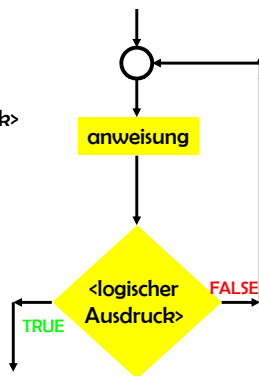
## Repetierte Anweisungen (REPEAT)

PSEUDOCODE

FLUSSDIAGRAMM

STRUKTOGRAMM

REPEAT anweisung;  
UNTIL <logischer Ausdruck>



19

# Algorithmische Sprachkonstrukte

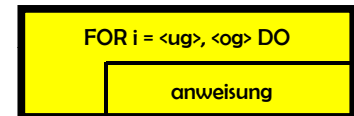
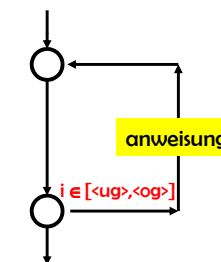
## Repetierte Anweisungen (FOR)

PSEUDOCODE

FLUSSDIAGRAMM

STRUKTOGRAMM

FOR i = <ug>, <og> DO  
anweisung;



20

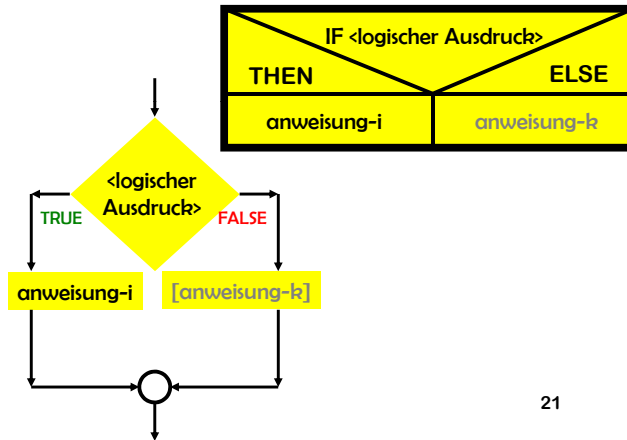
# Algorithmische Sprachkonstrukte

## Bedingte Anweisungen (IF)

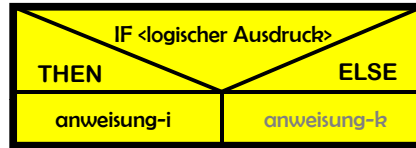
PSEUDOCODE

IF <logischer Ausdruck>  
 THEN anweisung-i;  
 (ELSE anweisung-k; )

FLUSSDIAGRAMM



STRUKTOGRAMM



21

# Computerlinguistisches Wissen:

## Silbentrennungsalgorithmus (Idee)

- Rekursive Suche nach Zerlegungen von links nach rechts.
- Bei jedem Schritt wird (entsprechend den Bedingungen im Syntaxdiagramm) ein maximaler Wortteil abgetrennt, der intern auf weitere Trennbarkeit untersucht wird.
- Unterscheidung von Haupttrennstellen (=) und Nebentrennstellen *innerhalb* von Teilwörtern (-)

• DRUCK = UN-TER-PRO-GRAMM = AUF-RU-FE<sup>22</sup>

# Computerlinguistisches Wissen:

## Silbentrennungsalgorithmus (Idee)

- **Rekursive** Suche nach Zerlegungen von links nach rechts.
- Bei jedem Schritt wird (entsprechend den Bedingungen im Syntaxdiagramm) ein maximaler Wortteil abgetrennt, der intern auf weitere Trennbarkeit untersucht wird.
- Unterscheidung von Haupttrennstellen (=) und Nebentrennstellen *innerhalb* von Teilwörtern (-)

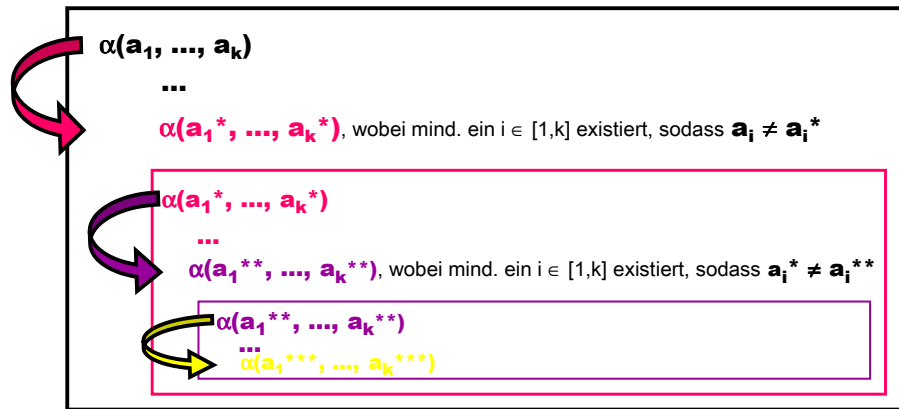
• DRUCK = UN-TER-PRO-GRAMM = AUF-RU-FE<sup>23</sup>

# Rekursion

- Ein Ausdruck  $\alpha$  (Programm, Prozedur, Funktion o.Ä.) heißt **rekursiv**, wenn seine Auswertung (Berechnung) mit einem Satz von Argumenten  $\delta$  die Auswertung eines Subausdrucks verlangt, die die erneute Auswertung von  $\alpha$  mit einem von  $\delta$  verschiedenen Satz von Argumenten  $\delta'$  verlangt.
- Es muss ein Terminierungskriterium für die Auswertung rekursiver Ausdrücke bestimmt und die **Terminierung** garantiert werden.

24

## Illustration der Rekursion



25

## Computerlinguistisches Wissen: Silbentrennungsalgorithmus (Idee)

- Rekursive Suche nach Zerlegungen von links nach rechts.
- Bei jedem Schritt wird (entsprechend den Bedingungen im Syntaxdiagramm) ein **maximaler** Wortteil abgetrennt, der intern auf weitere Trennbarkeit untersucht wird.
- Unterscheidung von Haupttrennstellen (=) und Nebentrennstellen *innerhalb* von Teilwörtern (-)

• DRUCK = UN-TER-PRO-GRAMM = AUF-RU-FE

26

## Maximalität eines Wortteils

- Ein Wortteil  $w' = c_1 \dots c_k$  eines Eingabeworts  $w = c_1 \dots c_k \dots c_n$ ,  $k \leq n$ ,
  - $c_i$ ,  $i=1..n$ , ist ein Buchstabe,
  - $n$  die Länge von  $w$

ist **maximal**, wenn es keinen längeren Wortteil  $w^* = c_1 \dots c_p$ ,  $p > k$ , im Lexikon gibt, der mit  $w$  in den ersten  $j$  Buchstaben,  $j=1, \dots, p$ , übereinstimmt.

27

## Prozedur LinksAb

- **Funktionalität:**  
 schneidet von EingabeWort ein maximales, im Lexikon auftretendes Wortfragment (Atom) linksbündig ab und erzeugt RestWort, das um das Präfix Atom reduzierte EingabeWort; liefert den Typ des Atoms (AtomTyp) und seine Länge (AtomLänge)

28

## Prozedur LinksAb

- **Hintergrundwissen:**

- Ein Lexikon, das Wortfragmente, ihren Typ und ggf. Angaben zu Ausnahmetrennungen enthält
- Syntax-Diagramme zur Beschreibung der morphologischen Struktur deutscher Wörter

- **Eingabeparameter:**

- EingabeWort das Eingabewort

- **Ausgabeparameter:**

- Atom Wortfragment aus Lexikon
- RestWort Atom – EingabeWort
- AtomTyp Typ des Atoms
- AtomLänge Länge des Atoms

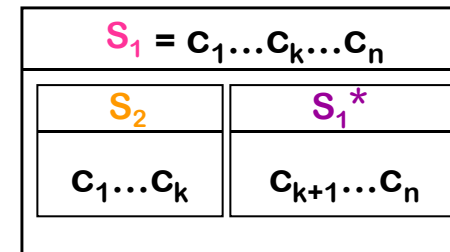
29

## Prozedur LinksAb

- **Verwendete Funktion:**

„-“: schneidet eine Teilkette aus Zeichenkette

Notation:  $S_2 - S_1 = S_1^*$



$||S_1|| = n$

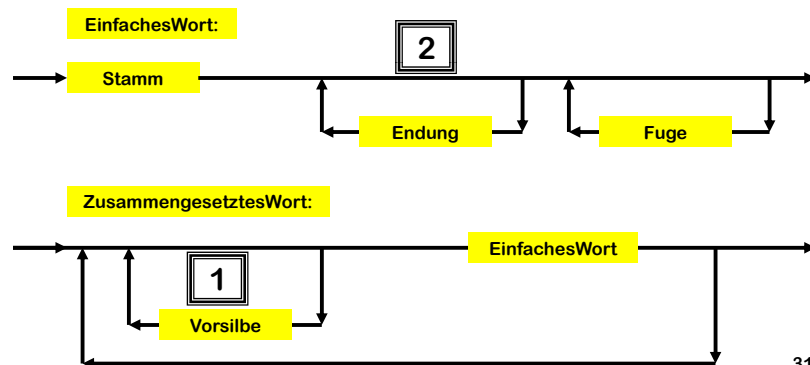
$||S_2|| = k$

$||S_1^*|| = n-k$

30

## Linguistisches Wissen: Morphologische Struktur von Wörtern

### Syntax-Diagramme



31

**Prozedur LinksAb**(↓EingabeWort, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

Suche als Präfix von **EingabeWort** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow \epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow 0$ ; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine **Vorsilbe** THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow ||\text{Atom}||$  ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein **Stamm** THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Stamm

**AtomLänge**  $\Leftarrow ||\text{Atom}||$  ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine **Endung** THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow ||\text{Atom}||$  ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

32

## Hauptprogramm Trennen

- **Funktionalität:**
  - s. Konzeption
- **Hintergrundwissen:**
  - Ein **Lexikon**, das Wortfragmente, ihren Typ und ggf. Angaben zu Ausnahmetrennungen enthält
  - **Syntax-Diagramme** zur Beschreibung der morphologischen Struktur deutscher Wörter
  - **Trennungsregeln** des Deutschen (R1-R4)

33

## Computerlinguistisches Wissen: Silbentrennungsalgorithmus (Idee)

- Rekursive Suche nach Zerlegungen von links nach rechts.
- Bei jedem Schritt wird (entsprechend den Bedingungen im Syntaxdiagramm) ein maximaler Wortteil abgetrennt, der intern auf weitere Trennbarkeit untersucht wird.
- Unterscheidung von Haupttrennstellen (=) und Nebentrennstellen *innerhalb* von Teilwörtern (-)
  - **DRUCK = UN-TER-PRO-GRAMM = AUF-RU-~~FE~~**<sup>34</sup>

## Hauptprogramm Trennen

- **Eingabeparameter:**
  - Wort                das zu trennende Wort
  - Zustand            Position im Syntax-Diagramm  
                          Initialwert = 1
- **Ausgabeparameter:**
  - Opcode             Statusmeldung nach Trennung:
    - „Erfolg“:        Trennung von Wort erfolgreich durchgeführt
    - „Misserfolg“:   keine Trennung von Wort möglich
    - „Offen“:        Zwischenzustand beim Lauf
- **Verwendete Prozeduren:**
  - LinksAb            Teilstring abschneiden von links
- **Quelle:**
  - Barth & Nirsch (1985)

35

**Programm Trennen**(↓Wort, ↓Zustand, ↑Opcode )

Opcode  $\Leftarrow$  „offen“

REPEAT

**LinksAb**( Wort, Teil, Typ, Länge, Rest )

    IF Länge > 0 THEN

      IF Zustand = 1 THEN

        trage Nebentrennstelle vor gefundenem Teil ein;

        IF Typ = Vorsilbe THEN

          trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

        IF Typ = Stamm THEN

          trage im Lexikon spezifizierte Ausnahmetrennung ein;

        IF Rest =  $\epsilon$  THEN

          trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

          Opcode  $\Leftarrow$  „Erfolg“

        ELSE

          Zustand  $\Leftarrow$  2

**Trennen**( Rest, Zustand, Opcode )

      ELSE ... ▲ ...

    ... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

36

**Programm Trennen**(↓Wort, ↓Zustand, ↑Opcode ) ... **▲** ...

```
REPEAT
  LinksAb( Wort, Teil, Typ, Länge, Rest )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( Rest, Zustand, Opcode )
        IF ... ▲ ...
      ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

37

**Programm Trennen**(↓Wort, ↓Zustand, ↑Opcode ) ... **▲** ...

```
REPEAT
  LinksAb( Wort, Teil, Typ, Länge, Rest )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          ...
        IF Typ = Vorsilbe OR Typ = Stamm THEN
          markiere eine Haupttrennstelle vor dem aktuellen Teil;
          trenne zwischen der letzten Trennstelle und dem Ende des
          aktuellen Teils gemäß R1-R4 (falls keine Ausnahmen kodiert sind)
        IF Rest = ε THEN Opcode ← „Erfolg“
        ELSE
          IF Typ = Vorsilbe THEN Zustand ← 1
          IF Typ = Stamm THEN Zustand ← 2
          Trennen( Rest, Zustand, Opcode )
        ...
      ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

38

**Programm Trennen**(↓Wort, ↓Zustand, ↑Opcode ) ... **B** ...

```
REPEAT
  LinksAb( Wort, Teil, Typ, Länge, Rest )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          ...
        IF Typ = Vorsilbe OR Typ = Stamm THEN
          ...
        IF Typ = „keine erlaubte Teilkette“ THEN
          Opcode ← „Misserfolg“
        IF Länge = 0 OR Opcode = „Misserfolg“ THEN
          lösche alle Trennstellen, die unmittelbar vor und innerhalb von
          Teil eingetragen wurden
      ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

39

## Trace für „Lampen“

40

**Programm Trennen**(↓Lampen, ↓1, ↑Opcode )

Opcode  $\Leftarrow$  „offen“

REPEAT

**LinksAb**( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\varepsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

41

**Programm Trennen**(↓Lampen, ↓1, ↑Opcode◀„offen“)

Opcode  $\Leftarrow$  „offen“

REPEAT

**LinksAb**( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\varepsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

42

**Programm Trennen**(↓Lampen, ↓1, ↑Opcode◀„offen“)

Opcode  $\Leftarrow$  „offen“

REPEAT

**LinksAb**( Lampen, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\varepsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

43

**Prozedur LinksAb**(↓Lampen, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

Suche als Präfix von EingabeWort nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

Atom  $\Leftarrow$   $\varepsilon$ ; AtomTyp  $\Leftarrow$  „keine erlaubte Teilkette“

AtomLänge  $\Leftarrow$  0; RestWort  $\Leftarrow$  EingabeWort

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

Atom  $\Leftarrow$  maximale Teilkette ; AtomTyp  $\Leftarrow$  Vorsilbe

AtomLänge  $\Leftarrow$  || Atom || ; RestWort  $\Leftarrow$  Atom – EingabeWort

IF maximale Teilkette ist ein Stamm THEN

Atom  $\Leftarrow$  maximale Teilkette ; AtomTyp  $\Leftarrow$  Stamm

AtomLänge  $\Leftarrow$  || Atom || ; RestWort  $\Leftarrow$  Atom – EingabeWort

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

Atom  $\Leftarrow$  maximale Teilkette ; AtomTyp  $\Leftarrow$  Endung

AtomLänge  $\Leftarrow$  || Atom || ; RestWort  $\Leftarrow$  Atom – EingabeWort

44

### Prozedur LinksAb(↓Lampen, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampen** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Stamm

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

45

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

### Prozedur LinksAb(↓Lampen, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampen** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  **Lampe** ; **AtomTyp**  $\Leftarrow$  **Stamm**

**AtomLänge**  $\Leftarrow$  || **Lampe** || ; **RestWort**  $\Leftarrow$  **Lampe** – **Lampen**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

46

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

### Prozedur LinksAb(↓Lampen, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampen** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  **Lampe** ; **AtomTyp**  $\Leftarrow$  **Stamm**

**AtomLänge**  $\Leftarrow$  5 ; **RestWort**  $\Leftarrow$  n

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

47

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

## Trace für „Lampen“

- **Trennen**(↓Lampen, ↓1, ↑Opcode)

Wort = **Lampen**; Zustand = 1; Opcode = „offen“

**LinksAb**(↓Lampen, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

– **Atom** = **Lampe**

**AtomTyp** = **Stamm**

– **AtomLänge** = 5

**Restwort** = n

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

48

Programm Trennen(↓Lampen, ↓1, ↑Opcode◀„offen“)

Opcode ⇐ „offen“

REPEAT

LinksAb( Lampen, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀n )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ⇐ „Erfolg“

ELSE

Zustand ⇐ 2

Trennen( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

49

Programm Trennen(↓Lampen, ↓1, ↑Opcode◀„offen“)

-Lampe

Opcode ⇐ „offen“

REPEAT

LinksAb( Lampen, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀n )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ⇐ „Erfolg“

ELSE

Zustand ⇐ 2

Trennen( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

50

## Trace für „Lampen“

• Trennen(↓Lampen, ↓1, ↑Opcode )

-Lampe

Wort = Lampen; Zustand = 1; Opcode = „offen“

LinksAb(↓Lampen, ↑Lampe, ↑Stamm, ↑5, ↑n )

– Teil = Lampe      Typ = Stamm

– Länge = 5      Rest = n

LEXIKON  
[ Lampe, Stamm, — ]  
[ n, Endung, — ]  
51

Programm Trennen(↓Lampen, ↓1, ↑Opcode◀„offen“)

-Lampe

Opcode ⇐ „offen“

REPEAT

LinksAb( Lampen, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀n )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ⇐ „Erfolg“

ELSE

Zustand ⇐ 2

Trennen( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

LEXIKON  
[ Lampe, Stamm, — ]  
[ n, Endung, — ]

52

Programm Trennen( $\downarrow$ Lampen,  $\downarrow$ 1,  $\uparrow$ Opcode  $\leftarrow$  „offen“ )

-Lampe

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Lampen, Teil  $\leftarrow$  Lampe, Typ  $\leftarrow$  Stamm, Länge  $\leftarrow$  5, Rest  $\leftarrow$  n )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil  $\leftarrow$  Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( n, 2, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

53

## Trace für „Lampen“

• Trennen( $\downarrow$ Lampen,  $\downarrow$ 1,  $\uparrow$ Opcode )

-Lampe

Wort = Lampen; Zustand = 1; Opcode = „offen“

LinksAb( $\downarrow$ Lampen,  $\uparrow$ Lampe,  $\uparrow$ Stamm,  $\uparrow$ 5,  $\uparrow$ n )

- Teil = Lampe                      Typ = Stamm

- Länge = 5                              Rest = n

- Trennen( $\downarrow$ n,  $\downarrow$ 2,  $\uparrow$ Opcode )

- Wort = n; Zustand = 2;

LEXIKON

[ Lampe, Stamm, — ]  
[ n, Endung, — ]

54

Programm Trennen( $\downarrow$ n,  $\downarrow$ 2,  $\uparrow$ Opcode )

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

55

Programm Trennen( $\downarrow$ n,  $\downarrow$ 2,  $\uparrow$ Opcode  $\leftarrow$  „offen“ )

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

56

**Programm Trennen**(↓n, ↓2, ↑Opcode ◀ „offen“ )

**Opcode** ⇐ „offen“

REPEAT

**LinksAb**( n, Teil, Typ, Länge, Rest )

IF **Länge** > 0 THEN

IF **Zustand** = 1 THEN

trage Nebentrennstelle vor gefundenem **Teil** ein;

IF **Typ** = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF **Typ** = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF **Rest** = ε THEN

trenne **Teil** gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode** ⇐ „Erfolg“

ELSE

**Zustand** ⇐ 2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

57

**Prozedur LinksAb**(↓n, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

Suche als Präfix von **EingabeWort** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ⇐ ε; **AtomTyp** ⇐ „keine erlaubte Teilkette“

**AtomLänge** ⇐ 0; **RestWort** ⇐ **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Vorsilbe

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Stamm

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Endung

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

58

**Prozedur LinksAb**(↓n, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

Suche als Präfix von **n** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ⇐ ε; **AtomTyp** ⇐ „keine erlaubte Teilkette“

**AtomLänge** ⇐ 0; **RestWort** ⇐ **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Vorsilbe

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Stamm

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Endung

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

59

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

**Prozedur LinksAb**(↓n, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

Suche als Präfix von **n** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ⇐ ε; **AtomTyp** ⇐ „keine erlaubte Teilkette“

**AtomLänge** ⇐ 0; **RestWort** ⇐ **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Vorsilbe

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Stamm

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ⇐ **n** ; **AtomTyp** ⇐ Endung

**AtomLänge** ⇐ || **n** || ; **RestWort** ⇐ **n** – **n**

60

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

### Prozedur LinksAb( $\downarrow n$ , $\uparrow$ Atom, $\uparrow$ AtomTyp, $\uparrow$ AtomLänge, $\uparrow$ RestWort)

Suche als Präfix von  $n$  nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

Atom  $\Leftarrow \varepsilon$ ; AtomTyp  $\Leftarrow$  „keine erlaubte Teilkette“

AtomLänge  $\Leftarrow 0$ ; RestWort  $\Leftarrow$  EingabeWort

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

Atom  $\Leftarrow$  maximale Teilkette; AtomTyp  $\Leftarrow$  Vorsilbe

AtomLänge  $\Leftarrow$  || Atom ||; RestWort  $\Leftarrow$  Atom – EingabeWort

IF maximale Teilkette ist ein Stamm THEN

Atom  $\Leftarrow$  maximale Teilkette; AtomTyp  $\Leftarrow$  Stamm

AtomLänge  $\Leftarrow$  || Atom ||; RestWort  $\Leftarrow$  Atom – EingabeWort

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

Atom  $\Leftarrow n$ ; AtomTyp  $\Leftarrow$  Endung

AtomLänge  $\Leftarrow 1$ ; RestWort  $\Leftarrow \varepsilon$

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

61

## Trace für „Lampen“

• Trennen( $\downarrow$ Lampen,  $\downarrow 1$ ,  $\uparrow$ Opcode)

–Lampe
--------

Wort = Lampen; Zustand = 1; Opcode = „offen“

LinksAb( $\downarrow$ Lampen,  $\uparrow$ Lampe,  $\uparrow$ Stamm,  $\uparrow 5$ ,  $\uparrow n$ )

– Teil = Lampe Typ = Stamm

– Länge = 5 Rest = n

– Trennen( $\downarrow n$ ,  $\downarrow 2$ ,  $\uparrow$ Opcode)

– Wort = n; Zustand = 2; Opcode = „offen“

– LinksAb( $\downarrow n$ ,  $\uparrow n$ ,  $\uparrow$ Endung,  $\uparrow 1$ ,  $\uparrow \varepsilon$ )

• Teil = n Typ = Endung

• Länge = 1 Rest =  $\varepsilon$

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]

62

### Programm Trennen ( $\downarrow n$ , $\downarrow 2$ , $\uparrow$ Opcode $\Leftarrow$ „offen“)

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( n, Teil  $\Leftarrow$  n, Typ  $\Leftarrow$  Endung, Länge  $\Leftarrow$  1, Rest  $\Leftarrow$   $\varepsilon$ )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\varepsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacktriangleright$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

63

### Programm Trennen ( $\downarrow n$ , $\downarrow 2$ , $\uparrow$ Opcode $\Leftarrow$ „offen“)

...  $\blacktriangle$  ...

REPEAT

LinksAb( n, Teil  $\Leftarrow$  n, Typ  $\Leftarrow$  Endung, Länge  $\Leftarrow$  1, Rest  $\Leftarrow$   $\varepsilon$ )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkateniere aktuelle Endung mit zugehörigem Stamm

(ggf. bereits um andere Endungen erweitert);

trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest =  $\varepsilon$  THEN

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Trennen( Rest, Zustand, Opcode )

IF ...  $\blacktriangle$  ...

...  $\blacktriangleright$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

64

Programm Trennen ( $\downarrow n, \downarrow 2, \uparrow \text{Opcode} \leftarrow \text{„offen“}$ ) ...  $\Delta$  ...

REPEAT

LinksAb(  $n, \text{Teil} \leftarrow n, \text{Typ} \leftarrow \text{Endung}, \text{Länge} \leftarrow 1, \text{Rest} \leftarrow \varepsilon$  )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkatiniere aktuelle Endung mit zugehörigem Stamm  
(ggf. bereits um andere Endungen erweitert);  
trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest =  $\varepsilon$  THEN

Opcode  $\leftarrow$  „Erfolg“

ELSE

Trennen( Rest, Zustand, Opcode )

IF ...  $\Delta$  ...

...  $B$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

65

-Lampe

Programm Trennen ( $\downarrow n, \downarrow 2, \uparrow \text{Opcode} \leftarrow \text{„offen“}$ ) ...  $\Delta$  ...

REPEAT

LinksAb(  $n, \text{Teil} \leftarrow n, \text{Typ} \leftarrow \text{Endung}, \text{Länge} \leftarrow 1, \text{Rest} \leftarrow \varepsilon$  )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkatiniere aktuelle Endung mit zugehörigem Stamm  
(ggf. bereits um andere Endungen erweitert);  
trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest =  $\varepsilon$  THEN

Opcode  $\leftarrow$  „Erfolg“

ELSE

Trennen( Rest, Zustand, Opcode )

IF ...  $\Delta$  ...

...  $B$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

66

-Lampen

Programm Trennen ( $\downarrow n, \downarrow 2, \uparrow \text{Opcode} \leftarrow \text{„offen“}$ ) ...  $\Delta$  ...

REPEAT

LinksAb(  $n, \text{Teil} \leftarrow n, \text{Typ} \leftarrow \text{Endung}, \text{Länge} \leftarrow 1, \text{Rest} \leftarrow \varepsilon$  )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkatiniere aktuelle Endung mit zugehörigem Stamm  
(ggf. bereits um andere Endungen erweitert);  
trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest =  $\varepsilon$  THEN

Opcode  $\leftarrow$  „Erfolg“

ELSE

Trennen( Rest, Zustand, Opcode )

IF ...  $\Delta$  ...

...  $B$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

67

-Lam-pen

Programm Trennen ( $\downarrow n, \downarrow 2, \uparrow \text{Opcode} \leftarrow \text{„Erfolg“}$ ) ...  $\Delta$  ...

REPEAT

LinksAb(  $n, \text{Teil} \leftarrow n, \text{Typ} \leftarrow \text{Endung}, \text{Länge} \leftarrow 1, \text{Rest} \leftarrow \varepsilon$  )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkatiniere aktuelle Endung mit zugehörigem Stamm  
(ggf. bereits um andere Endungen erweitert);  
trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest =  $\varepsilon$  THEN

Opcode  $\leftarrow$  „Erfolg“

ELSE

Trennen( Rest, Zustand, Opcode )

IF ...  $\Delta$  ...

...  $B$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

68

-Lam-pen

# Trace für „Lampen“

• **Trennen**(↓Lampe, ↓1, ↑Erfolg )

-Lam-pen (R1.1,R3.1)

Wort = Lampe; Zustand = 1; Opcode = „offen“ „Erfolg“

**LinksAb**(↓Lampe, ↑Lampe, ↑Stamm, ↑5, ↑n )

– Teil = Lampe            Typ = Stamm

– Länge = 5                Rest = n

– **Trennen**(↓n, ↓2, ↑Erfolg )

– Wort = n; Zustand = 2; Opcode = „offen“ „Erfolg“

– **LinksAb**(↓n, ↑n, ↑Endung, ↑1, ↑ε)

• Teil = n                    Typ = Endung

• Länge = 1                 Rest = ε

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
69

**Programm Trennen** (↓n, ↓2, ↑Opcode◀„Erfolg“ )

... B ...

REPEAT

**LinksAb**( n, Teil◀n, Typ◀Endung, Länge◀1, Rest◀ε )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

...

IF Typ = „keine erlaubte Teilkette“ THEN

**Opcode** ◀ „Misserfolg“

IF Länge = 0 OR **Opcode** = „Misserfolg“ THEN

  lösche alle Trennstellen, die unmittelbar vor und innerhalb von Teil eingetragen wurden

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

70

**Programm Trennen**(↓Lampen, ↓1, ↑Opcode◀„Erfolg“)

-Lam-pen

**Opcode** ◀ „offen“

REPEAT

**LinksAb**( Lampen, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀n )

IF Länge > 0 THEN

  IF Zustand = 1 THEN

    trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;

    IF Typ = Vorsilbe THEN

      trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

    IF Typ = Stamm THEN

      trage im Lexikon spezifizierte Ausnahmetrennung ein;

    IF Rest = ε THEN

      trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode** ◀ „Erfolg“

  ELSE

**Zustand** ◀ 2

**Trennen**( n, 2, Opcode◀„Erfolg“ )

  ELSE ... ▲ ...

... B ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

71

**Programm Trennen** (↓Lampen, ↓1, ↑Opcode◀„Erfolg“)

... B ...

REPEAT

**LinksAb**( Lampen, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀n )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

...

IF Typ = „keine erlaubte Teilkette“ THEN

**Opcode** ◀ „Misserfolg“

IF Länge = 0 OR **Opcode** = „Misserfolg“ THEN

  lösche alle Trennstellen, die unmittelbar vor und innerhalb von Teil eingetragen wurden

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

72

# Trace für „Lampenschirm“

73

Programm Trennen( $\downarrow$ Lampenschirm,  $\downarrow$ 1,  $\uparrow$ Opcode)

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

74

Programm Trennen( $\downarrow$ Lampenschirm,  $\downarrow$ 1,  $\uparrow$ Opcode  $\Leftarrow$  „offen“)

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

75

Programm Trennen( $\downarrow$ Lampenschirm,  $\downarrow$ 1,  $\uparrow$ Opcode  $\Leftarrow$  „offen“)

Opcode  $\Leftarrow$  „offen“

REPEAT

LinksAb( Lampenschirm, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest =  $\epsilon$  THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode  $\Leftarrow$  „Erfolg“

ELSE

Zustand  $\Leftarrow$  2

Trennen( Rest, Zustand, Opcode )

ELSE ...  $\blacktriangle$  ...

...  $\blacklozenge$  ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

76

**Prozedur LinksAb**(↓Lampenschirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **EingabeWort** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Stamm

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

**Prozedur LinksAb**(↓Lampenschirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampenschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Stamm

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

**Prozedur LinksAb**(↓Lampenschirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampenschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein **Stamm** THEN

**Atom**  $\Leftarrow$  **Lampe** ; **AtomTyp**  $\Leftarrow$  **Stamm**

**AtomLänge**  $\Leftarrow$  || **Lampe** || ; **RestWort**  $\Leftarrow$  **Lampe** – **Lampenschirm**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

**Prozedur LinksAb**(↓Lampenschirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **Lampenschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\epsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein **Stamm** THEN

**Atom**  $\Leftarrow$  **Lampe** ; **AtomTyp**  $\Leftarrow$  **Stamm**

**AtomLänge**  $\Leftarrow$  **5** ; **RestWort**  $\Leftarrow$  **nschirm**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  || **Atom** || ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

# Trace für „Lampenschirm“

• **Trennen**(↓Lampenschirm, ↓1, ↑Opcode )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

**LinksAb**(↓Lampenschirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort )

- Atom = Lampe            AtomTyp = Stamm
- AtomLänge = 5        Restwort = nschirm

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

**Programm Trennen**(↓Lampenschirm, ↓1, ↑Opcode◀„offen“ )

Opcode ← „offen“

REPEAT

**LinksAb**( Lampenschirm, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀nschirm )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

82

**Programm Trennen**(↓Lampenschirm, ↓1, ↑Opcode◀„offen“ )

Opcode ← „offen“

REPEAT

**LinksAb**( Lampenschirm, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀nschirm )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

83

**Programm Trennen**(↓Lampenschirm, ↓1, ↑Opcode◀„offen“ )

Opcode ← „offen“

REPEAT

**LinksAb**( Lampenschirm, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀nschirm )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

**Trennen**( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

84

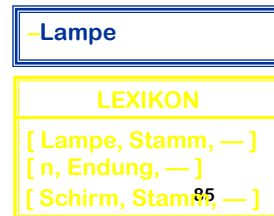
# Trace für „Lampenschirm“

• Trennen(↓Lampenschirm, ↓1, ↑Opcode )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

LinksAb(↓Lampenschirm, ↑Lampe, ↑Stamm, ↑5, ↑nschirm)

- Teil = Lampe            Typ = Stamm
- Länge = 5                Rest = nschirm



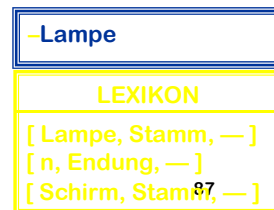
# Trace für „Lampenschirm“

• Trennen(↓Lampenschirm, ↓1, ↑Opcode )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

LinksAb(↓Lampenschirm, ↑Lampe, ↑Stamm, ↑5, ↑nschirm)

- Teil = Lampe            Typ = Stamm
- Länge = 5                Rest = nschirm
- Trennen(↓nschirm, ↓2, ↑Opcode )
- Wort = nschirm; Zustand = 2;



Programm Trennen(↓Lampenschirm, ↓1, ↑Opcode ← „offen“ )

Opcode ← „offen“

REPEAT

LinksAb( Lampenschirm, Teil ← Lampe, Typ ← Stamm, Länge ← 5, Rest ← nschirm )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ← Lampe ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

Trennen( nschirm, 2, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

86

Programm Trennen(↓nschirm, ↓2, ↑Opcode )

Opcode ← „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

Trennen( Rest, Zustand, Opcode )

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

Trennen( Rest, Zustand, Opcode )

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

88

**Programm Trennen**(↓**nschirm**, ↓**2**, ↑**Opcode**◀„offen“)

**Opcode** ⇐ „offen“

REPEAT

**LinksAb**( **Wort**, **Teil**, **Typ**, **Länge**, **Rest** )

IF **Länge** > 0 THEN

IF **Zustand** = 1 THEN

trage Nebentrennstelle vor gefundenem **Teil** ein;

IF **Typ** = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( **Rest**, **Zustand**, **Opcode** )

IF **Typ** = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF **Rest** = ε THEN

trenne **Teil** gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode** ⇐ „Erfolg“

ELSE

**Zustand** ⇐ 2

**Trennen**( **Rest**, **Zustand**, **Opcode** )

ELSE ... ▲ ...

... B ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

89

**Programm Trennen**(↓**nschirm**, ↓**2**, ↑**Opcode**◀„offen“)

**Opcode** ⇐ „offen“

REPEAT

**LinksAb**( **nschirm**, **Teil**, **Typ**, **Länge**, **Rest** )

IF **Länge** > 0 THEN

IF **Zustand** = 1 THEN

trage Nebentrennstelle vor gefundenem **Teil** ein;

IF **Typ** = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( **Rest**, **Zustand**, **Opcode** )

IF **Typ** = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF **Rest** = ε THEN

trenne **Teil** gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode** ⇐ „Erfolg“

ELSE

**Zustand** ⇐ 2

**Trennen**( **Rest**, **Zustand**, **Opcode** )

ELSE ... ▲ ...

... B ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

90

**Prozedur LinksAb**(↓**nschirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort** )

Suche als Präfix von **EingabeWort** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ⇐ ε; **AtomTyp** ⇐ „keine erlaubte Teilkette“

**AtomLänge** ⇐ 0; **RestWort** ⇐ **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Vorsilbe

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Stamm

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Endung

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

91

**Prozedur LinksAb**(↓**nschirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort** )

Suche als Präfix von **nschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ⇐ ε; **AtomTyp** ⇐ „keine erlaubte Teilkette“

**AtomLänge** ⇐ 0; **RestWort** ⇐ **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Vorsilbe

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Stamm

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ⇐ maximale Teilkette ; **AtomTyp** ⇐ Endung

**AtomLänge** ⇐ || **Atom** || ; **RestWort** ⇐ **Atom** – **EingabeWort**

92

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

**Prozedur LinksAb**(↓**nschirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort** )

Suche als Präfix von **nschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ← ε; **AtomTyp** ← „keine erlaubte Teilkette“

**AtomLänge** ← 0; **RestWort** ← **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Vorsilbe

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Stamm

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine **Endung** THEN

**Atom** ← n ; **AtomTyp** ← **Endung**

**AtomLänge** ← || n || ; **RestWort** ← n – **nschirm**

LEXIKON		
[ Lampe, Stamm, — ]		
[ n, Endung, — ]		
[ Schirm, Stamm, — ]		

**Prozedur LinksAb**(↓**nschirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort** )

Suche als Präfix von **nschirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ← ε; **AtomTyp** ← „keine erlaubte Teilkette“

**AtomLänge** ← 0; **RestWort** ← **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Vorsilbe

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Stamm

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine **Endung** THEN

**Atom** ← n ; **AtomTyp** ← **Endung**

**AtomLänge** ← 1 ; **RestWort** ← **schirm**

# Trace für „Lampenschirm“

• **Trennen**(↓**Lampenschirm**, ↓1, ↑**Opcode** )

**Wort** = Lampenschirm; **Zustand** = 1; **Opcode** = „offen“

**LinksAb**(↓**Lampenschirm**, ↑**Lampe**, ↑**Stamm**, ↑5, ↑**nschirm** )

– **Teil** = Lampe            **Typ** = Stamm

– **Länge** = 5              **Rest** = nschirm

– **Trennen**(↓**nschirm**, ↓2, ↑**Opcode** )

– **Wort** = nschirm; **Zustand** = 2; **Opcode** = „offen“

– **LinksAb**(↓**nschirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort** )

• **Atom** = n              **AtomTyp** = Endung

• **AtomLänge** = 1      **Restwort** = schirm

LEXIKON		
[ Lampe, Stamm, — ]		
[ n, Endung, — ]		
[ Schirm, Stamm, — ]		

**Programm Trennen** (↓**nschirm**, ↓2, ↑**Opcode** ← „offen“ )

**Opcode** ← „offen“

REPEAT

**LinksAb**( **nschirm**, **Teil** ← n, **Typ** ← **Endung**, **Länge** ← 1, **Rest** ← **schirm** )

IF **Länge** > 0 THEN

IF **Zustand** = 1 THEN

trage Nebentrennstelle vor gefundenem **Teil** ein;

IF **Typ** = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( **Rest**, **Zustand**, **Opcode** )

IF **Typ** = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF **Rest** = ε THEN

trenne **Teil** gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode** ← „Erfolg“

ELSE

**Zustand** ← 2

**Trennen**( **Rest**, **Zustand**, **Opcode** )

ELSE ... ▲ ...

... B ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

Programm Trennen (↓nschirm, ↓2, ↑Opcode◀„offen“) ... ▲ ...

```

REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( Rest, Zustand, Opcode )
        IF ... ▲ ...
      ... B ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
  
```

Programm Trennen (↓nschirm, ↓2, ↑Opcode◀„offen“) ... ▲ ...

```

REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( Rest, Zustand, Opcode )
        IF ... ▲ ...
      ... B ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
  
```

-Lampe

Programm Trennen (↓nschirm, ↓2, ↑Opcode◀„offen“) ... ▲ ...

```

REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( Rest, Zustand, Opcode )
        IF ... ▲ ...
      ... B ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
  
```

-Lampen

Programm Trennen (↓nschirm, ↓2, ↑Opcode◀„offen“) ... ▲ ...

```

REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( Rest, Zustand, Opcode )
        IF ... ▲ ...
      ... B ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
  
```

-Lam-pen

# Trace für „Lampenschirm“

## • Trennen(↓Lampenschirm, ↓1, ↑Opcode )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

LinksAb(↓Lampenschirm, ↑Lampe, ↑Stamm, ↑5, ↑nschirm )

- Teil = Lampe            Typ = Stamm
  - Länge = 5              Rest = nschirm
- Lam-pen (R1.1,R3.1)
- Trennen(↓nschirm, ↓2, ↑Opcode )
  - Wort = nschirm; Zustand = 2; Opcode = „offen“
  - LinksAb(↓nschirm, ↑n, ↑Endung, ↑1, ↑schirm)
    - Teil = n                    Typ = Endung
    - Länge = 1                Rest = schirm

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

Programm Trennen (↓nschirm, ↓2, ↑Opcode ◀ „offen“ )                    ... ▲ ...

```

REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      IF Zustand = 2 THEN
        IF Typ = Endung THEN
          -Lam-pen
          konkateniere aktuelle Endung mit zugehörigem Stamm
          (ggf. bereits um andere Endungen erweitert);
          trenne die entstandene Zeichenkette gemäß R1-R4
        IF Rest = ε THEN
          Opcode ← „Erfolg“
        ELSE
          Trennen( schirm, 2, Opcode )
        IF ... ▲ ...
      ... B ...
    UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
  
```

Programm Trennen(↓schirm, ↓2, ↑Opcode )

Opcode ← „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

```

IF Länge > 0 THEN
  IF Zustand = 1 THEN
    trage Nebentrennstelle vor gefundenem Teil ein;
    IF Typ = Vorsilbe THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
      Trennen( Rest, Zustand, Opcode )
    IF Typ = Stamm THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
      IF Rest = ε THEN
        trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);
        Opcode ← „Erfolg“
      ELSE
        Zustand ← 2
        Trennen( Rest, Zustand, Opcode )
    ELSE ... ▲ ...
  ... B ...

```

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

Programm Trennen(↓schirm, ↓2, ↑Opcode ◀ „offen“ )

Opcode ← „offen“

REPEAT

LinksAb( Wort, Teil, Typ, Länge, Rest )

```

IF Länge > 0 THEN
  IF Zustand = 1 THEN
    trage Nebentrennstelle vor gefundenem Teil ein;
    IF Typ = Vorsilbe THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
      Trennen( Rest, Zustand, Opcode )
    IF Typ = Stamm THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
      IF Rest = ε THEN
        trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);
        Opcode ← „Erfolg“
      ELSE
        Zustand ← 2
        Trennen( Rest, Zustand, Opcode )
    ELSE ... ▲ ...
  ... B ...

```

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

**Programm Trennen**(↓schirm, ↓2, ↑Opcode◀„offen“)

Opcode ← „offen“

REPEAT

**LinksAb**(schirm, Teil, Typ, Länge, Rest)

IF Länge > 0 THEN

IF Zustand = 1 THEN

trage Nebentrennstelle vor gefundenem Teil ein;

IF Typ = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**(Rest, Zustand, Opcode)

IF Typ = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF Rest = ε THEN

trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

Opcode ← „Erfolg“

ELSE

Zustand ← 2

**Trennen**(Rest, Zustand, Opcode)

ELSE ... ▲ ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

105

**Prozedur LinksAb**(↓schirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **EingabeWort** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ← ε; **AtomTyp** ← „keine erlaubte Teilkette“

**AtomLänge** ← 0; **RestWort** ← **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Vorsilbe

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Stamm

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Endung

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

106

**Prozedur LinksAb**(↓schirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **schirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ← ε; **AtomTyp** ← „keine erlaubte Teilkette“

**AtomLänge** ← 0; **RestWort** ← **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Vorsilbe

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Stamm

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Endung

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

107

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

**Prozedur LinksAb**(↓schirm, ↑Atom, ↑AtomTyp, ↑AtomLänge, ↑RestWort)

Suche als Präfix von **schirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom** ← ε; **AtomTyp** ← „keine erlaubte Teilkette“

**AtomLänge** ← 0; **RestWort** ← **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Vorsilbe

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom** ← **schirm**; **AtomTyp** ← **Stamm**

**AtomLänge** ← || **schirm** || ; **RestWort** ← **schirm** – **schirm**

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom** ← maximale Teilkette ; **AtomTyp** ← Endung

**AtomLänge** ← || **Atom** || ; **RestWort** ← **Atom** – **EingabeWort**

108

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

### Prozedur LinksAb( $\downarrow$ schirm, $\uparrow$ Atom, $\uparrow$ AtomTyp, $\uparrow$ AtomLänge, $\uparrow$ RestWort)

Suche als Präfix von **schirm** nach einer maximalen Teilkette im Lexikon

IF Suche erfolglos THEN

**Atom**  $\Leftarrow$   $\varepsilon$ ; **AtomTyp**  $\Leftarrow$  „keine erlaubte Teilkette“

**AtomLänge**  $\Leftarrow$  0; **RestWort**  $\Leftarrow$  **EingabeWort**

ELSE

IF Syntaxdiagramm.Zustand = „1“ OR Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Vorsilbe THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Vorsilbe

**AtomLänge**  $\Leftarrow$  | **Atom** | ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

IF maximale Teilkette ist ein Stamm THEN

**Atom**  $\Leftarrow$  **schirm**; **AtomTyp**  $\Leftarrow$  **Stamm**

**AtomLänge**  $\Leftarrow$  **6** ; **RestWort**  $\Leftarrow$   $\varepsilon$

IF Syntaxdiagramm.Zustand = „2“ THEN

IF maximale Teilkette ist eine Endung THEN

**Atom**  $\Leftarrow$  maximale Teilkette ; **AtomTyp**  $\Leftarrow$  Endung

**AtomLänge**  $\Leftarrow$  | **Atom** | ; **RestWort**  $\Leftarrow$  **Atom** – **EingabeWort**

109

### Programm Trennen( $\downarrow$ schirm, $\downarrow$ 2, $\uparrow$ Opcode $\Leftarrow$ „offen“)

**Opcode**  $\Leftarrow$  „offen“

REPEAT

**LinksAb**( **schirm**, **Teil**  $\Leftarrow$  **schirm**, **Typ**  $\Leftarrow$  **Stamm**, **Länge**  $\Leftarrow$  **6**, **Rest**  $\Leftarrow$   $\varepsilon$  )

IF **Länge** > 0 THEN

IF **Zustand** = 1 THEN

trage Nebentrennstelle vor gefundenem **Teil** ein;

IF **Typ** = Vorsilbe THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

**Trennen**( **Rest**, **Zustand**, **Opcode** )

IF **Typ** = Stamm THEN

trage im Lexikon spezifizierte Ausnahmetrennung ein;

IF **Rest** =  $\varepsilon$  THEN

trenne **Teil** gemäß R1-R4 (falls keine Ausnahmen kodiert sind);

**Opcode**  $\Leftarrow$  „Erfolg“

ELSE

**Zustand**  $\Leftarrow$  2

**Trennen**( **Rest**, **Zustand**, **Opcode** )

ELSE ...  $\blacktriangle$  ...

... **B** ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

110

### Programm Trennen( $\downarrow$ schirm, $\downarrow$ 2, $\uparrow$ Opcode $\Leftarrow$ „offen“)

...  $\blacktriangle$  ...

REPEAT

**LinksAb**( **schirm**, **Teil**  $\Leftarrow$  **schirm**, **Typ**  $\Leftarrow$  **Stamm**, **Länge**  $\Leftarrow$  **6**, **Rest**  $\Leftarrow$   $\varepsilon$  )

IF **Länge** > 0 THEN

...

IF **Zustand** = 1 THEN

...

ELSE

IF **Zustand** = 2 THEN

IF **Typ** = Endung THEN

konkatenerie aktuelle Endung mit zugehörigem Stamm

(ggf. bereits um andere Endungen erweitert);

trenne die entstandene Zeichenkette gemäß R1-R4

IF **Rest** =  $\varepsilon$  THEN

**Opcode**  $\Leftarrow$  „Erfolg“

ELSE

**Trennen**( **Rest**, **Zustand**, **Opcode** )

IF ...  $\blacktriangle$  ...

... **B** ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

111

### Programm Trennen( $\downarrow$ schirm, $\downarrow$ 2, $\uparrow$ Opcode $\Leftarrow$ „offen“)

...  $\blacktriangle$  ...

REPEAT

**LinksAb**( **schirm**, **Teil**  $\Leftarrow$  **schirm**, **Typ**  $\Leftarrow$  **Stamm**, **Länge**  $\Leftarrow$  **6**, **Rest**  $\Leftarrow$   $\varepsilon$  )

IF **Länge** > 0 THEN

...

IF **Zustand** = 1 THEN

...

ELSE

IF **Zustand** = 2 THEN

IF **Typ** = Endung THEN

...

IF **Typ** = Vorsilbe OR **Typ** = Stamm THEN

markiere eine Haupttrennstelle vor dem aktuellen **Teil**;

trenne zwischen der letzten Trennstelle und dem Ende des

aktuellen **Teils** gemäß R1-R4 (falls keine Ausnahmen kodiert sind)

IF **Rest** =  $\varepsilon$  THEN **Opcode**  $\Leftarrow$  „Erfolg“

ELSE

IF **Typ** = Vorsilbe THEN **Zustand**  $\Leftarrow$  1

IF **Typ** = Stamm THEN **Zustand**  $\Leftarrow$  2

**Trennen**( **Rest**, **Zustand**, **Opcode** )

... **B** ...

UNTIL **Opcode** = „Erfolg“ OR **Opcode** = „Misserfolg“

112

Programm Trennen(↓schirm, ↓2, ↑Opcode◀„offen“) ... ▲1 ...

REPEAT

LinksAb( scherm, Teil◀schirm, Typ◀Stamm, Länge◀6, Rest◀ε )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

...

IF Typ = Vorsilbe OR Typ = Stamm THEN

markiere eine Haupttrennstelle vor dem aktuellen Teil;  
trenne zwischen der letzten Trennstelle und dem Ende des  
aktuellen Teils gemäß R1-R4 (falls keine Ausnahmen kodiert sind)

IF Rest = ε THEN Opcode ∈ „Erfolg“

ELSE

IF Typ = Vorsilbe THEN Zustand ∈ 1

IF Typ = Stamm THEN Zustand ∈ 2

Trennen( Rest, Zustand, Opcode )

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

-Lam-pen=schirm

113

Programm Trennen(↓schirm, ↓2, ↑Opcode◀„Erfolg“) ... ▲1 ...

REPEAT

LinksAb( scherm, Teil◀schirm, Typ◀Stamm, Länge◀6, Rest◀ε )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

...

IF Typ = Vorsilbe OR Typ = Stamm THEN

markiere eine Haupttrennstelle vor dem aktuellen Teil;  
trenne zwischen der letzten Trennstelle und dem Ende des  
aktuellen Teils gemäß R1-R4 (falls keine Ausnahmen kodiert sind)

IF Rest = ε THEN Opcode ∈ „Erfolg“

ELSE

IF Typ = Vorsilbe THEN Zustand ∈ 1

IF Typ = Stamm THEN Zustand ∈ 2

Trennen( Rest, Zustand, Opcode )

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

-Lam-pen=schirm

114

Programm Trennen(↓schirm, ↓2, ↑Opcode◀„Erfolg“) ... B ...

REPEAT

LinksAb( scherm, Teil◀schirm, Typ◀Stamm, Länge◀6, Rest◀ε )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

...

IF Typ = Vorsilbe OR Typ = Stamm THEN

...

IF Typ = „keine erlaubte Teilkette“ THEN

Opcode ∈ „Misserfolg“

IF Länge = 0 OR Opcode = „Misserfolg“ THEN

lösche alle Trennstellen, die unmittelbar vor und innerhalb von  
Teil eingetragen wurden

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

115

Programm Trennen(↓nschirm, ↓2, ↑Opcode◀„Erfolg“) ... ▲1 ...

REPEAT

LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )

IF Länge > 0 THEN

...

IF Zustand = 1 THEN

...

ELSE

IF Zustand = 2 THEN

IF Typ = Endung THEN

konkatiniere aktuelle Endung mit zugehörigem Stamm  
(ggf. bereits um andere Endungen erweitert);  
trenne die entstandene Zeichenkette gemäß R1-R4

IF Rest = ε THEN

Opcode ∈ „Erfolg“

ELSE

Trennen( scherm, 2, Opcode◀„Erfolg“ )

IF ... ▲1 ...

... B ...

UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“

116

Programm Trennen (↓**nschirm**, ↓**2**, ↑Opcode◀**„Erfolg“**) ... **B** ...

```
REPEAT
  LinksAb( nschirm, Teil◀n, Typ◀Endung, Länge◀1, Rest◀schirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      ...
  IF Typ = „keine erlaubte Teilkette“ THEN
    Opcode ⇐ „Misserfolg“
  IF Länge = 0 OR Opcode = „Misserfolg“ THEN
    lösche alle Trennstellen, die unmittelbar vor und innerhalb von
    Teil eingetragen wurden
  UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

117

Programm Trennen(↓**Lampenschirm**, ↓**1**, ↑Opcode◀**„Erfolg“**)

```
Opcode ⇐ „offen“
REPEAT
  LinksAb( Lampenschirm, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀nschirm )
  IF Länge > 0 THEN
    IF Zustand = 1 THEN
      trage Nebentrennstelle vor gefundenem Teil◀Lampe ein;
    IF Typ = Vorsilbe THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
      Trennen( Rest, Zustand, Opcode )
    IF Typ = Stamm THEN
      trage im Lexikon spezifizierte Ausnahmetrennung ein;
    IF Rest = ε THEN
      trenne Teil gemäß R1-R4 (falls keine Ausnahmen kodiert sind);
      Opcode ⇐ „Erfolg“
    ELSE
      Zustand ⇐ 2
      Trennen( nschirm, 2, Opcode◀„Erfolg“ )
  ELSE ... A ...
  ... B ...
  UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

118

Programm Trennen (↓**Lampenschirm**, ↓**1**, ↑Opcode◀**„Erfolg“**) ... **B** ...

```
REPEAT
  LinksAb( Lampenschirm, Teil◀Lampe, Typ◀Stamm, Länge◀5, Rest◀nschirm )
  IF Länge > 0 THEN
    ...
    IF Zustand = 1 THEN
      ...
    ELSE
      ...
  IF Typ = „keine erlaubte Teilkette“ THEN
    Opcode ⇐ „Misserfolg“
  IF Länge = 0 OR Opcode = „Misserfolg“ THEN
    lösche alle Trennstellen, die unmittelbar vor und innerhalb von
    Teil eingetragen wurden
  UNTIL Opcode = „Erfolg“ OR Opcode = „Misserfolg“
```

119

• Trennen(↓**Lampenschirm**, ↓**1**, ↑Opcode)

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

LinksAb(↓**Lampenschirm**, ↑**Lampe**, ↑**Stamm**, ↑**5**, ↑**nschirm**)

- Teil = Lampe            Typ = Stamm
- Länge = 5              Rest = nschirm
- Trennen(↓**nschirm**, ↓**2**, ↑Opcode)
- Wort = nschirm; Zustand = 2; Opcode = „offen“
- LinksAb(↓**nschirm**, ↑**n**, ↑**Endung**, ↑**1**, ↑**schirm**)

- Teil = n                    Typ = Endung
- Länge = 1                  Rest = schirm
- Trennen(↓**schirm**, ↓**2**, ↑Opcode)
- Wort = schirm; Zustand = 2; Opcode = „offen“
- LinksAb(↓**schirm**, ↑**Atom**, ↑**AtomTyp**, ↑**AtomLänge**, ↑**RestWort**)
  - Atom = schirm            AtomTyp = Stamm
  - AtomLänge = 6            Restwort = ε

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

120

- Trennen(↓Lampenschirm, ↓1, ↑Opcode )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“

LinksAb(↓Lampenschirm, ↑Lampe, ↑Stamm, ↑5, ↑nschirm)

- Teil = Lampe            Typ = Stamm
- Länge = 5              Rest = nschirm

- Trennen(↓nschirm, ↓2, ↑Opcode )

- Wort = nschirm; Zustand = 2; Opcode = „offen“

- LinksAb(↓nschirm, ↑n, ↑Endung, ↑1, ↑schirm)

- Teil = n                  Typ = Endung
- Länge = 1              Rest = schirm
- Trennen(↓schirm, ↓2, ↑Opcode )
- Wort = schirm; Zustand = 2; Opcode = „offen“

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

- LinksAb(↓schirm, ↑schirm, ↑Stamm, ↑6, ↑ε)

- Teil = schirm            Typ = Stamm
- Länge = 6              Rest = ε

-Lam-pen (R1.1, R3.1)

-Lam-pen=schirm (R1.1, R1.2<sup>121</sup>, R3.1)

- Trennen(↓Lampenschirm, ↓1, ↑Erfolg )

Wort = Lampenschirm; Zustand = 1; Opcode = „offen“ „Erfolg“

LinksAb(↓Lampenschirm, ↑Lampe, ↑Stamm, ↑5, ↑nschirm)

- Teil = Lampe            Typ = Stamm
- Länge = 5              Rest = nschirm

- Trennen(↓nschirm, ↓2, ↑Erfolg )

- Wort = nschirm; Zustand = 2; Opcode = „offen“ „Erfolg“

- LinksAb(↓nschirm, ↑n, ↑Endung, ↑1, ↑schirm)

- Teil = n                  Typ = Endung
- Länge = 1              Rest = schirm
- Trennen(↓schirm, ↓2, ↑Erfolg )
- Wort = schirm; Zustand = 2; Opcode = „offen“ „Erfolg“

LEXIKON
[ Lampe, Stamm, — ]
[ n, Endung, — ]
[ Schirm, Stamm, — ]

- LinksAb(↓schirm, ↑schirm, ↑Stamm, ↑6, ↑ε)

- Teil = schirm            Typ = Stamm
- Länge = 6              Rest = ε

-Lam-pen=schirm (R1.1, R1.2<sup>122</sup>, R3.1)

