
Rekursion in der Sprache

Formale Grammatiken und Rekursion

- Rekursion in hierarchischen Strukturen lässt sich formalisieren durch das Konzept der **generativen Grammatik**.
- Eine (generative) Grammatik ist ein Formalismus zum Generieren von Zeichenketten über einem endlichen Alphabet (Chomsky 1955, Chomsky 1963, Chomsky 1965).
- Die Menge der Zeichenketten, die von einer Grammatik G charakterisiert wird, nennt man die **Sprache** $L(G)$ der Grammatik.

Grammatiken 2

(1) *Grammatik:*

Ein Grammatik ist ein Quadrupel $\langle V_T, V_N, S, R \rangle$,

V_T = endliche Menge von **Terminalen**,

V_N = endliche Menge von **Nichtterminalen**,

S = **Startsymbol**, und

R = endliche Menge von **Regeln** der Form $\psi \rightarrow \omega$, wobei ψ und ω Ketten sind.

- **Regelinterpretation:** Wann immer ψ als Teilkette K einer anderen Kette K' auftaucht, kann diese Instanz von ψ durch ω in K' ersetzt werden, wodurch eine neue Kette K'' erzeugt wird.
- Mit anderen Worten: Für beliebige Ketten u, v aus Symbolen $\in V_T \cup V_N$ bedeutet $u \Rightarrow_G v$, dass es x, y und ein $A \in V_N$ gibt, so dass $u = xAy$ und $A \rightarrow_G v' \in R$ und $v = xv'y$.

Ableitungen

- Jede Sequenz der Form $w_0 \Rightarrow_G w_1 \Rightarrow_G \dots \Rightarrow_G w_n$ ist eine **Derivation** (oder **Ableitung**) von w_n in G von w_0 aus.
- \Rightarrow_G^* bezeichnet eine Derivation in beliebig vielen Schritten.
- $L(G)$ = die Sprache, die von G **generiert** wird = $\{w \text{ über dem Alphabet } \Sigma \mid S \Rightarrow_G^* w\}$. G generiert jede Kette in $L(G)$.
- Für eine Sprache L gilt $L = L(G)$, wenn es eine Grammatik G gibt, die es erlaubt, L zu generieren.

Grammatiken 3

(2) Grammatik G_1 (vgl. Partee et al. (1990, 437)):

a. $V_T = \{a, b\}$

b. $V_N = \{S, A, B\}$

c. S

d. $R = \left\{ \begin{array}{l} S \rightarrow ABS \\ S \rightarrow \epsilon \\ A \rightarrow B \\ B \rightarrow A \\ A \rightarrow a \\ B \rightarrow b \end{array} \right\}$

- (ϵ bezeichnet die **leere** Kette: S löst sich auf).

(3) *Beispielableitung 1:*

$$\begin{aligned} S &\Rightarrow ABS \Rightarrow ABABS \Rightarrow ABABABS \Rightarrow \\ &ABABAB \Rightarrow ABBBAB \Rightarrow ABBAAB \Rightarrow \\ &aBBAAB \Rightarrow abBAAB \Rightarrow abbAAB \Rightarrow \\ &abbaAB \Rightarrow abbaaB \Rightarrow abbaab \end{aligned}$$

Grammatiken 4

(2) Grammatik G_1 (wiederholt):

a. $V_T = \{a, b\}$

b. $V_N = \{S, A, B\}$

c. S

d. $R = \left\{ \begin{array}{l} S \rightarrow ABS \\ S \rightarrow \epsilon \\ A \rightarrow B \\ B \rightarrow A \\ A \rightarrow a \\ B \rightarrow b \end{array} \right\}$

(4) *Beispielableitung 2:*

$$\begin{aligned} S &\Rightarrow ABS \Rightarrow ABABS \Rightarrow ABAB \Rightarrow \\ &BBAB \Rightarrow BAAB \Rightarrow BABB \Rightarrow BABA \Rightarrow \\ &bABA \Rightarrow baBA \Rightarrow babA \Rightarrow baba \end{aligned}$$

Grammatiken 5

- G_1 in (2) charakterisiert die Sprache $L(G_1)$ der Ketten, die aus beliebigen Sequenzen von as und bs bestehen: $L(G_1) = \{a, b\}^*$
- (M^* bezeichnet die **Menge aller Ketten** aus Elementen der Menge M .)
- Diese Sprache (eine Menge) ist **unendlich**. Die Unendlichkeit entsteht durch **Rekursion**.
- Rekursion wird in G_1 dadurch ausgedrückt, dass dasselbe Nichtterminal auf beiden Seiten einer Regel auftaucht (z.B. $S \rightarrow ABS$ in (2)).

Indirekte Rekursion

- **Indirekte** Rekursion: dasselbe Nichtterminal steht einmal links und einmal rechts von \rightarrow in zwei Regeln R_1, R_2 , die Teil derselben Ableitung sind.
- G_2 in (5) enthält ausschließlich Ableitungen mit indirekter aber nicht mit direkter Rekursion; sie entstehen durch die Folgen $S \rightarrow aB, B \rightarrow bS$ und $S \rightarrow bA, A \rightarrow aS$.

(5) *Grammatik* G_2 :

a. $V_T = \{a, b\}$

b. $V_N = \{S, A, B\}$

c. S

d. $R = \left\{ \begin{array}{l} S \rightarrow aB \\ S \rightarrow bA \\ S \rightarrow \epsilon \\ A \rightarrow aS \\ B \rightarrow bS \\ A \rightarrow a \\ B \rightarrow b \end{array} \right\}$

Äquivalente Derivationen

(6) Grammatik G_3 :

a. $V_T = \{ (,) \}$

b. $V_N = \{ S \}$

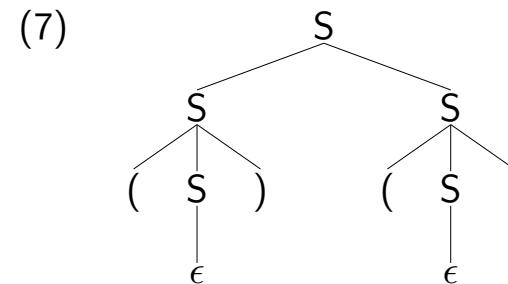
c. S

d. $R = \left\{ \begin{array}{l} S \rightarrow \epsilon \\ S \rightarrow SS \\ S \rightarrow (S) \end{array} \right\}$

- $L(G_3)$ ist die Sprache der **ausbalancierten Klammern**.
- Es gibt wenigstens zwei verschiedene, aber **äquivalente** Derivationen, die $()()$ ableiten.
 1. $S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$
 2. $S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$
- Die äquivalenten Derivationen sind demselben **Derivations-** oder **Ableitungsbaum** zugeordnet.

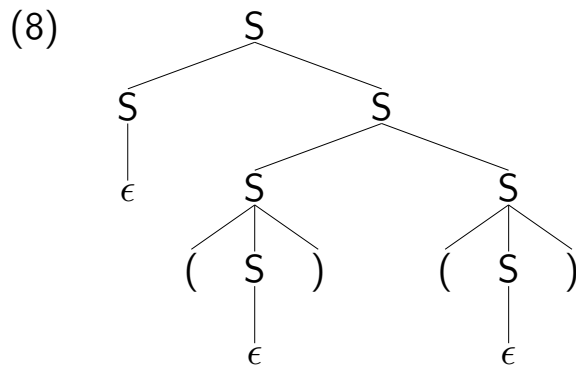
Ableitungsäume

- Der Ableitungsbaum für den Ausdruck $()()$, der diesen Derivationen zugeordnet ist, ist (7).



Ableitungsbäume 2

- Es gibt **unendlich** viele **nicht äquivalente** Ableitungen für den Ausdruck $()()$.
- Der Ableitungsbaum für eine dieser Ableitungen ist (8).



Schwache versus starke generative Kapazität

- Man unterscheidet **starke** und **schwache generative Kapazität** einer Grammatik.
 1. Die starke generative Kapazität bemisst sich nach der Menge der Ableitungsbäume, die eine Grammatik generiert.
 2. Die schwache generative Kapazität bemisst sich nach der Menge der Ketten, die eine Grammatik generiert.
- Meistens werden Grammatiken verglichen bezüglich ihrer schwachen generativen Kapazitäten.

Regelbeschränkungen

- Grammatikregeln können sich dadurch unterscheiden, welchen **Beschränkungen** ihre Form unterliegt.
- Eine Grammatik nimmt je nachdem, welche Form ihre Regeln haben, eine bestimmte Position in der sogenannten **Chomsky-Hierarchie** ein (siehe (Chomsky 1963)).

Chomsky-Hierarchie

- (9) *Chomsky-Hierarchie:*
- a. **Typ-0**-Grammatiken:
keine Beschränkungen
 - b. **Typ-1**-Grammatiken (“kontextsensitiv”):
alle Regeln haben die Form $\psi \rightarrow \omega$, wobei ω mindestens so lang sein muss wie ψ .
 - c. **Typ-2**-Grammatiken (“kontextfrei”):
alle Regeln haben die Form $A \rightarrow \psi$.
 - d. **Typ-3**-Grammatiken (“regulär”):
alle Regeln haben die Form $A \rightarrow xB$ oder $A \rightarrow x$.
- (α, β, ψ sind Ketten aus $(V_T \cup V_N)^*$. A und B sind Nichtterminale, x eine Terminalkette.)
 - Achtung: (9) beschreibt **keine** Teilmengenbeziehung zwischen den Mengen von Grammatiken, in dem Sinne, dass z.B. die Menge der Typ-1-Grammatiken die Menge der Typ-2-Grammatiken enthielte: Typ-2-Grammatiken erlauben z.B. Regeln der Form $A \rightarrow \epsilon$, Typ-1-Grammatiken nicht.

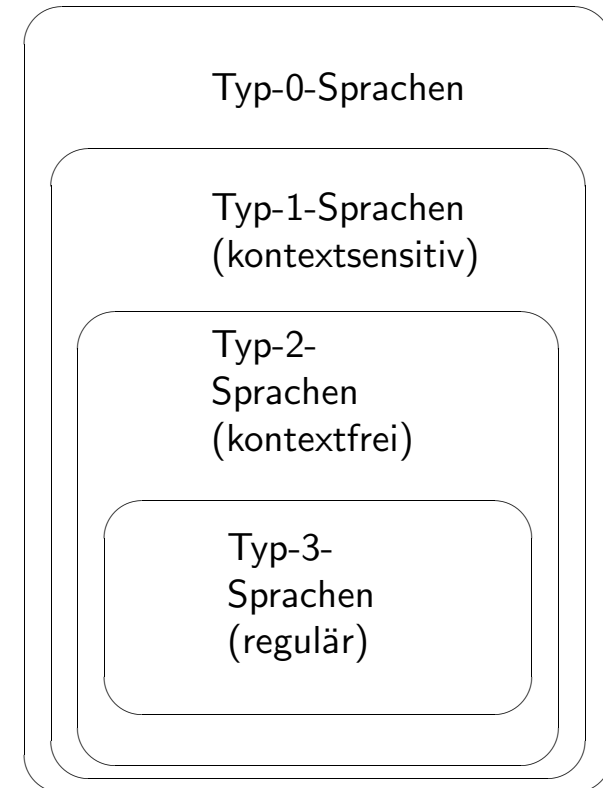
Chomsky-Hierarchie 2

- Frage: In welchem Sinne ist die Chomsky-Hierarchie denn dann eine Hierarchie?
- Antwort: Die Mengen der **Sprachen**, die von den Grammatikmengen der Chomsky-Hierarchie generiert werden, stehen in Teilmengenbeziehung zueinander, siehe (10). Eigentlich bilden also die Mengen der Sprachen die Hierarchie.

(10) Typ-0-Sprachen \supset Typ-1-Sprachen \supset Typ-2-Sprachen \supset Typ-3-Sprachen

- **Definition:** Ein **Sprache** L ist genau dann **Typ- n** , wenn es eine Typ- n -Grammatik G gibt, so dass $L = L(G)$.

Chomsky-Hierarchie 3



Typ- n -Grammatik $\not\Rightarrow$ Typ- n -Sprache

- Grammatik G_1 und Grammatik G_2 sind nicht vom gleichen Typ: G_1 ist Typ-2, G_2 ist Typ-3.
- Die Sprachen $L(G_1)$ und $L(G_2)$ gehören aber derselben Sprachklasse an: beide sind Typ-3-Sprachen.
 1. $L(G_1) = \{a, b\}^*$
 2. $L(G_2) = \{K \mid K \text{ besteht aus } ab\text{- und } ba\text{-Paaren.}\}$

(11) Regeln von G_1 : Regeln von G_2 :

$$R = \left\{ \begin{array}{l} S \rightarrow ABS \\ S \rightarrow \epsilon \\ A \rightarrow B \\ B \rightarrow A \\ A \rightarrow a \\ B \rightarrow b \end{array} \right\} \quad R = \left\{ \begin{array}{l} S \rightarrow aB \\ S \rightarrow bA \\ S \rightarrow \epsilon \\ A \rightarrow aS \\ B \rightarrow bS \\ A \rightarrow a \\ B \rightarrow b \end{array} \right\}$$

- Wieso ist $L(G_1)$ Typ-3, wenn doch G_1 Typ-2 ist? Weil es eine Typ-3-Grammatik G_4 gibt, so dass $L(G_4) = L(G_1)$ (siehe nächste Seite).

Typ- n -Grammatik $\not\Rightarrow$ Typ- n -Sprache 2

(12) Grammatik G_4 :

- a. $V_T = \{a, b\}$
- b. $V_N = \{S\}$
- c. S
- d. $R = \left\{ \begin{array}{l} S \rightarrow aS \\ S \rightarrow bS \\ S \rightarrow \epsilon \end{array} \right\}$

- Man kann also nicht schließen, dass eine Sprache L **nicht** Typ- n (z.B. Typ-3) ist, wenn es eine Typ- m -Grammatik mit $m < n$ (also Typ-2, -1 oder -0) gibt, die L erzeugt.
- Es könnte ja im Prinzip auch eine Typ- n -Grammatik geben, die L generiert, womit L dann eben doch eine Typ- n -Sprache ist.
- Allerdings gibt es eine solche Typ- n -Grammatik nicht für jede Sprache, die von einer Typ- m -Grammatik (mit $m < n$) generiert wird.

Typ- n -Grammatik $\not\Rightarrow$ Typ- n -Sprache 3

- Ein weiteres Beispiel:

(13) Grammatik G_5 :

- $V_T = \{a, b\}$
- $V_N = \{S, B\}$
- S

$$d. R = \left\{ \begin{array}{l} S \rightarrow aS \\ S \rightarrow bB \\ S \rightarrow \epsilon \\ B \rightarrow bSbS \\ B \rightarrow a \end{array} \right\}$$

- $L(G_5) = \{K \in \{a, b\}^* \mid K \text{ enthält eine ungerade Zahl an } bs\}$
- $L(G_5)$ ist Typ-3 (siehe nächste Seite), obwohl G_5 klar eine Typ-2-Grammatik ist.

Typ- n -Grammatik $\not\Rightarrow$ Typ- n -Sprache 4

- Es gibt nämlich eine Typ-3-Grammatik G'_5 , die $L(G_5)$ generiert, also $L(G_5) = L(G'_5)$.

(14) Grammatik G'_5 :

- $V_T = \{a, b\}$
- $V_N = \{S, B, C\}$
- S

$$d. R = \left\{ \begin{array}{l} S \rightarrow aS \\ S \rightarrow bB \\ S \rightarrow b \\ S \rightarrow \epsilon \\ B \rightarrow Cb \\ B \rightarrow bC \\ C \rightarrow bS \\ C \rightarrow Sb \\ B \rightarrow a \end{array} \right\}$$

Rand- versus Mittenrekursion

- Die Menge der Typ-3-Sprachen ist in der Menge der Typ-2-Sprachen echt enthalten.
- Intuitiv liegt das daran, dass bei Typ-3-Regeln ein Nichtterminalsymbol nur am Rand (am Anfang, $A \rightarrow Ax$, oder am Ende, $A \rightarrow xA$) auftauchen kann, bei Typ-2-Regeln aber in der Mitte.
- Das erste nennt man auch **Randrekursion** (oder **Endrekursion**), das zweite **Mittenrekursion** (oder **Zentraleinbettung**).

Rand- versus Mittenrekursion 2

- **Endliche** Sprachen können immer von Typ-3-Grammatiken generiert werden: Man macht einfach für jedes Wort der Sprache eine Typ-3-Regel!
- Für **unendliche** Sprachen gilt das nicht. Unendlichkeit kommt durch Rekursion zustande. Der Rekursionstyp wird dann entscheidend.
- Man braucht Mittenrekursion, um eine Sprache L erzeugen zu können, die nicht von einer Typ-3-Grammatik charakterisiert werden kann.

Rand- versus Mittenrekursion 3

(15) Grammatik G_6 :

a. $V_T = \{a, b\}$

b. $V_N = \{S\}$

c. S

d. $R = \left\{ \begin{array}{l} S \rightarrow aSb \\ S \rightarrow \epsilon \end{array} \right\}$

e. $L(G_6) = \{a^n b^n | n \geq 0\}$

- Grammatik G_6 involviert Mittenrekursion, ist also mindestens vom Typ-2.
- Behauptung: Keine Typ-3-Grammatik kann die von (15) erzeugte Sprache $L(G_6)$ erzeugen.
- Dies kann man beweisen. Der Beweis dafür wird üblicherweise über das **Pumping-Lemma** geführt (vgl. Partee et al. (1990, 472)).

Pumping-Lemma für Typ-3-Sprachen

(16) *Pumping-Lemma für Typ-3-Sprachen:*

Wenn L eine Typ-3-Sprache ist, dann gibt es eine Zahl p (die **Pumpzahl**), so dass jedes Wort $w \in L$, mit $|w| \geq p$ aufgeteilt werden kann in $w = xyz$, so dass a., b. und c. erfüllt sind.

a. $|xy| \leq p$ (Bedingung 1)

b. $|y| > 0$ (Bedingung 2)

c. für jedes $i \geq 0 : xy^i z \in L$ (Bedingung 3)

Pumping-Lemma für Typ-3-Sprachen 2

- Behauptung: $L(G_6)$ ist nicht Typ-3. Der Beweis erfolgt durch **Widerspruch**:

$$(17) \quad L(G_6) = \{a^n b^n \mid n \geq 0\}$$

- Schema Beweis durch Widerspruch:
 1. Angenommen $L(G_6)$ ist vom Typ-3.
 2. Dann gilt das Pumping-Lemma für $L(G_6)$.
 3. Man zeigt, dass das Pumping-Lemma nicht für $L(G_6)$ gilt (Widerspruch zur 1.).
 4. Man folgert (**Modus Tollens**) dass $L(G_6)$ nicht Typ-3 sein kann.

Pumping-Lemma für Typ-3-Sprachen 3

- Beweis: Sei p die Pumpzahl und $w = a^p b^p$ ($|w| > p$). Man zeigt nun, dass es kein y geben kann, mit $|y| > 0$, $w = xyz$ und $xy^i z \in L(G_6)$, für alle $i \geq 0$.
 1. Fall 1: Angenommen, y enthält nur as . Dann führte das Aufpumpen von y zu mehr as als bs in w (dann aber wäre $w \notin L(G_6)$).
 2. Fall 2: Angenommen, y enthält nur bs . Dann führte das Aufpumpen von y zu mehr bs als as in w (dann aber wäre $w \notin L(G_6)$).
 3. Fall 3: Angenommen, y besteht aus as und bs . Dann führte das Aufpumpen von y zu wiederholt abwechselndem Auftauchen von as als bs in w (dann gilt wieder: $w \notin L(G_6)$).
 4. y kann also nicht existieren. Das ist aber ein Widerspruch zum Pumping-Lemma. Daher kann $L(G_6)$ nicht regulär sein.

Pumping-Lemma für Typ-3-Sprachen 4

- Fortsetzung Beweis:
- Alle drei möglichen Belegungen für y wurden getestet. Keine war geeignet, die Bedingungen des Pumping-Lemmas zu erfüllen.
- Das Pumping-Lemma gilt also nicht für $L(G_6)$, entgegen der Annahme.
- Wenn aber das Pumping-Lemma für $L(G_6)$ nicht gilt, dann ist $L(G_6)$ nicht Typ-3.

Daumenregel

- Wenn in einer Sprache beliebig große **Abhängigkeiten** zwischen den Symbolen der Ketten bestehen, ist sie nicht vom Typ-3.
- In $L(G_6)$ ($\{a^n b^n | n \geq 0\}$) hängt die Zahl der bs von der Zahl der as ab, (oder umgekehrt) und zwar für beliebig viele as (bzw. bs).
- Eine Typ-3-Grammatik müsste sich die Zahl der as **merken** können, um dann die entsprechende Zahl an bs zu erzeugen.
- Das geht mit Randrekursion nicht, wohl aber mit Mittenrekursion, wie zum Beispiel mit G_6 in (15), wo as und bs gleichzeitig erzeugt werden.

Pumping-Lemma für Typ-3-Sprachen 5

- Beachte:
 1. Wenn man zeigen kann, dass eine Sprache L das Pumping-Lemma für Typ-3-Sprachen erfüllt, folgt daraus noch **nicht**, dass L selber Typ-3 ist.
 2. Aber **umgekehrt**: Wenn eine Sprache L das Pumping-Lemma **nicht** erfüllt, dann kann man schließen, dass L **nicht** Typ-3 ist.
- Beispiel: $L = \{K \in \{a, b\}^* \mid K \text{ enthält gleich viele } as \text{ wie } bs.\}$
 1. Wähle $x = \epsilon, y = ab, z = \epsilon$.
 2. Dann liegen $xy^0z = \epsilon, xy^1z = ab, xy^2z = abab,$ etc. alle in L (ϵ ist Teil jeder Sprache).
 3. Damit trifft das Pumping-Lemma auf L zu, aber L ist keine Typ-3-Sprache.
 4. Intuition: Eine Grammatik, die L generiert muss as und bs zählen können (oder sich die Zahl der as merken können, bevor er die bs generiert).

Natürliche Sprachen \supset Typ-3

- Behauptung: Natürliche Sprachen sind mindestens vom Typ-2: natürliche Sprachen \supseteq Typ-2 \supset Typ-3.
- Beweisidee (siehe, z.B., Partee et al. (1990, 480ff.):
 1. Unabhängige Erkenntnis: Der (mengentheoretische) **Schnitt** zweier Typ-3-Sprachen liefert wieder eine Typ-3-Sprache.
 2. Wenn z.B. Englisch Typ-3 wäre, dann sollte eine Typ-3-Sprache herauskommen, wenn man Englisch mit einer Typ-3-Sprache schneidet.
 3. Kommt keine Typ-3-Sprache heraus (überprüfbar durch das Pumping-Lemma), kann man umgekehrt schlussfolgern, dass Englisch nicht Typ-3 ist.

Natürliche Sprachen \supset Typ-3 2

(18) *Englische Relativsätze:*

- a. The cat died
- b. The cat the dog chased died
- c. The cat the dog the rat bit chased died
- d. The cat the dog the rat the elephant admired bit chased died

- Die Sätze in (18) folgen alle dem folgenden Muster: $(the\ N)^n (V_{trans})^{n-1} V_{intrans}$. Eine Sprache, die nur aus solchen Sätzen besteht, ist (19-b).
- L (siehe (19-b)) ergibt sich aus dem Schnitt von Englisch mit L' (siehe (19-a)), einer Typ-3-Sprache.

- (19) a. $L' = A^*B^*\{\text{died}\}$, wobei
- (i) $A = \{\text{the cat, the dog, the rat, the elephant, the bee, } \dots\}$ und
 - (ii) $B = \{\text{chased, bit, kicked, admired, kissed, } \dots\}$
- b. $L = \{a^n b^{n-1} \text{died} \mid a \in A, b \in B\}$

Natürliche Sprachen \supset Typ-3 3

- Mit dem Pumping-Lemma lässt sich zeigen: L ist nicht Typ-3.
- Der Beweis läuft analog zum Beweis, dass $L(G_6)$ nicht Typ-3 ist: Es kann kein $y \neq \epsilon$ gefunden werden, so dass y sich **aufpumpen** lässt und dabei sowohl
 1. das Verhältnis von n as zu $(n - 1)$ bs erhalten bleibt als auch
 2. die relative Reihenfolge von as und bs erhalten bleibt.

Natürliche Sprachen \supset Typ-3 4

- Ein ähnlicher Punkt lässt sich mit Beispielen aus der Morphologie machen (Lasnik (2000, 15); siehe auch Pinker (1995, 130)).

(20) *Englische Komposita:*

- a. missiles
- b. [[anti missile] missiles]
- c. [[anti [anti missile] missile] missiles]
- d. [[anti [anti [anti missile] missile] missile] missiles]
- e. *anti-missiles
- f. *anti-anti-missiles

- Die Komposita in (20) folgen alle dem Muster $\text{anti}^n \text{missile}^{n+1}$
- Argument: Auch in (20) hängt die Anzahl n der Instanzen von *anti-* ab von der Anzahl $n + 1$ der Instanzen von *missile* (und zwar für beliebig große n). Typ-3-Grammatiken können das nicht ausdrücken.

Pumping-Lemma für Typ-2-Sprachen

(21) *Pumping-Lemma für Typ-2-Sprachen:*

Wenn L eine Typ-2-Sprache ist, dann gibt es eine Pumpzahl p , so dass jedes $w \in L, |w| > p$, zerlegt werden kann in $w = uvxyz$, so dass a., b. und c. erfüllt sind.

- a. $|vxy| \leq p$ (Bedingung 1)
- b. $|vy| > 0$ (nicht: v **und** y leer) (Bedingung 2)
- c. Für alle $i \geq 0 : uv^i xy^i z \in L$ (Bedingung 3)

- So, wie man (16) benutzen kann, um zu zeigen, dass eine Sprache nicht Typ-3 ist, kann man (21) benutzen, um zu zeigen, dass eine Sprache nicht Typ-2 ist.
- Das geschieht wieder über Beweis durch Widerspruch.

Pumping-Lemma für Typ-2-Sprachen 2

- Behauptung: L ist nicht Typ-2.

$$(22) \quad L = \{a^n b^n c^n \mid n \geq 0\}$$

- Beweis: Angenommen L ist Typ-2. Dann muss das Pumping-Lemma gelten. Sei p Pumpzahl und $w = uvxyz = a^p b^p c^p \in L$ (d.h. es gilt $|w| \geq p$).
- Angenommen v und y bestehen aus verschiedenen Symbolen.
 1. Dann können das wegen Bedingung 1 ($vxy \leq p$) nur benachbarte sein (as und bs oder bs und cs).
 2. v kann nicht aus as und bs bestehen, weil das Aufpumpen von v dann Ketten erzeugt, bei denen bs vor as stehen ($\notin L$).
 3. Das gleiche gilt, wenn v aus bs und cs besteht. Und dasselbe gilt auch für y .

Pumping-Lemma für Typ-2-Sprachen 3

- Fortsetzung Beweis:
- Angenommen v und y bestehen nur aus einem Symbol.
 1. Besteht v nur aus as und y nur aus bs , dann führt Pumpen zu einem w mit mehr as und bs , aber die cs bleiben gleich ($w \notin L$).
 2. Besteht v aus bs und y aus cs , dann führt Pumpen zu einem w mit mehr bs und cs , aber die as bleiben gleich (wieder $w \notin L$).
- Dies erschöpft die Möglichkeiten für v und y .
- Keine der Belegungen konnte die Bedingungen des Pumping-Lemmas (21) erfüllen.
- Wenn L aber das Pumping-Lemma nicht erfüllt, kann L nicht Typ-2 sein.

Natürliche Sprachen \supset Typ-2

(23) *Überkreuzende Abhängigkeit, Schweizerdeutsch*

- a. Jan säit das mer em Hans es Huus
Jan sagte dass wir dem Hans das Haus
hälfed aastriiche
helfen anzustreichen
- b. Jan säit das mer d'Chind em Hans
Jan sagte dass wir die=Kinder dem Hans
es Huus lönd hälfe aastriiche
das Haus lassen helfen anzustreichen

- Beobachtungen:
 1. *d'Chind* in (23-b) ist Objekt von *lönd*, *em Hans* Objekt von *hälfe* und *es Huus* Objekt von *aastriiche*.
 2. Die Objekte sind den Verben eindeutig durch Kasus zugeordnet (*hälfe* regiert Dativ, die anderen beiden Verben Akkusativ).
 3. Diese Abhängigkeiten können im Prinzip noch komplizierter werden.

Natürliche Sprachen \supset Typ-2 2

- Idee (Shieber 1985): Schneide Schweizerdeutsch mit der Typ-3-Sprache R in (24-a).
- Das Resultat dieses Schnittes ist die Sprache L in (24-b).

- (24) a. $R =$ Jan säit das mer (d'Chind)^{*} (em Hans)^{*}
es Huus händ wele (laa)^{*} (hälfe)^{*} aastriiche
b. $L =$ Jan säit das mer (d'Chind)ⁿ (em Hans)^m
es Huus händ wele (laa)ⁿ (hälfe)^m aastriiche

- In L stimmt die Zahl der Akkusativobjekte mit der Zahl der Verben überein, die Akkusativ zuweisen; dasselbe für Dativobjekte und ihre Verben.
- Die Wörter von L sind von der abstrakten Form $qa^n b^m r c^n d^m s$. Die **Abhängigkeiten** zwischen a und c auf der einen und b und d auf der anderen Seite **überkreuzen** sich (vgl. (Bresnan, Kaplan, Peters & Zaenen 1982)).

Natürliche Sprachen \supset Typ-2 3

- Unabhängige Erkenntnis: Typ-2-Sprachen geschnitten mit Typ-3-Sprachen ergeben immer Typ-2-Sprachen.
- Behauptung: L ist nicht Typ-2. Dies kann man durch das Pumping-Lemma für Typ-2-Sprachen (siehe (21)) zeigen.
- Dann aber kann auch das Schweizerdeutsche nicht vom Typ-2 sein (wegen der Erkenntnis oben). Daraus wird dann gefolgert, dass natürliche Sprachen im allgemeinen nicht vom Typ-2 sind.

Natürliche Sprachen \supset Typ-2 4

- Beweis: $L = \{qa^n b^m r c^n d^m s \mid m, n \geq 0\}$. Angenommen L ist Typ-2. Sei p die Pumpzahl und $w = uvxyz = qa^p b^p r c^p d^p s$ ($|w| > p$).

 1. Wegen $|vxy| \leq p$ kann vy höchstens aus zwei verschiedenen Symbolen (a, b, c oder d) bestehen; diese müssen aufeinanderfolgend sein.
 2. Fall 1: Besteht vy nur aus as , dann ist uxz (v^i und y^i mit $i = 0$) nicht in L , da es weniger as als cs gibt. Dasselbe gilt, wenn vy nur aus bs , cs und ds besteht.
 3. Fall 2: Besteht vy aus as und bs , dann ist $uxz \notin L$: weniger as als cs und weniger bs als ds . Dasselbe gilt, wenn vy aus bs und cs oder cs und ds besteht.
 4. Da dies alle Möglichkeiten sind, entsteht ein Widerspruch zum Pumping-Lemma. Also kann L nicht kontextfrei sein.

Natürliche Sprachen ⊃ Typ-2 5

- Wieder lässt sich ein ähnliches Argument aus der Morphologie anführen (Culy 1985).
 1. In Bambara (Mande, Mali), können zwei Nomen mit -o- komponiert werden, (25).
 2. Die Nomen müssen identisch sein, siehe (26).

(25) *Wortbildung Nr. 1 in Bambara:*

- a. wulu-o-wulu
Hund-o-Hund
“was immer für ein Hund”
- b. malo-o-malo
Reis-o-Reis
“was immer für Reis”

(26) *Identitätsbeschränkung für Wortbildung Nr. 1:*

- a. *wulu-o-malo
Hund-o-Reis
- b. *malo-o-wulo
Reis-o-Hund

Natürliche Sprachen ⊃ Typ-2 6

- Es gibt einen Prozess Nr. 2 in Bambara der Nomen + transitives Verb + “jemand” kombiniert zu einem Wort, (27-a,b).

(27) *Wortbildung Nr. 2 in Bambara:*

- a. wulu-nyinin-na
Hund-such-jemand
“jemand, der Hunde sucht”
- b. malo-filè-la
Reis-bewach-jemand
“jemand der Reis bewacht”

Natürliche Sprachen ⊃ Typ-2 7

- Dieser Prozess Nr. 2 kann sogar rekursiv applizieren, siehe (28-a-d).

(28) *Rekursive Applikation von Prozess Nr. 2:*

- wulunyinina-nyini-na
Hundesucher-such-jemand
“jemand der nach Hundesuchern sucht”
- wulunyinina-filè-la
Hundesucher-bewach-jemand
“jemand der Hundesucher bewacht”
- malonyinina-nyini-na
Reissucher-such-jemand
“jemand der Reissucher sucht”
- malonyinina-filè-la
Reissucher-bewach-jemand
“jemand der Reissucher bewacht”

Natürliche Sprachen ⊃ Typ-2 8

- Die Prozesse Nr. 1 und 2 können kombiniert werden, siehe (29-a-d).

(29) *Kombination von Nr 1. und Nr 2.:*

- wulunyinina-o-wulunyinina
Hundesucher-o-Hundesucher
“was immer für ein Hundesucher”
- malofilèla-o-malofilèla
Reisbewacher-o-Reisbewacher
“was immer für ein Reisbewacher”
- wulunyininafilèla-o-wulunyininafilèla
H.-sucherbewacher-o-H.-sucherbewacher
“wer auch immer Hundesucher bewacht”
- wulunfilèlafilèla-o-wulunfilèlafilèla
H.-bew.-bewacher-o-H.-bew.bewacher
“wer auch immer Hundebewacher bewacht”

Natürliche Sprachen \supset Typ-2 9

- Culy's Idee: Schneide die reguläre Sprache R in (30-a) mit dem Vokabular von Bambara B . Das Ergebnis ist B' in (30-b).

$$(30) \quad \begin{array}{l} \text{a. } R = \{ \text{wulu}(\text{filèla})^h(\text{nyinina})^i\text{-o-} \\ \quad \quad \quad \text{wulu}(\text{filèla})^j(\text{nyinina})^k \mid h, i, j, k \geq 1 \} \\ \text{b. } B' = \{ \text{wulu}(\text{filèla})^m(\text{nyinina})^n\text{-o-} \\ \quad \quad \quad \text{wulu}(\text{filèla})^m(\text{nyinina})^n \mid m, n \geq 1 \} \end{array}$$

- Wenn man das o ignoriert, dann ist B' von der Form $\{a^m b^n a^m b^n \mid n \geq 1\}$
- Argument:
 1. Eine Typ-2-Sprache geschnitten mit einer Typ-3-Sprache ergibt immer eine Typ-2-Sprache.
 2. Wenn das Vokabular von Bambara B Typ-2 wäre, dann müsste B' Typ-2 sein.
 3. Man kann mit dem Pumping-Lemma zeigen, dass B nicht Typ-2 ist.
 4. Es folgt, dass B auch nicht Typ-2 sein kann.

Literatur

Bresnan, Joan, Ron Kaplan, Stanley Peters & Annie Zaenen (1982): 'Cross-Serial Dependencies in Dutch', *Linguistic Inquiry* **13**, 613–635.

Chomsky, Noam (1955): *The Logical Structure of Linguistic Theory*. published (in part) 1975 by Plenum Press.

Chomsky, Noam (1963): Formal Properties of Grammars. In: R. D. Luce, R. Bush & E. Galanter, eds, *Handbook of Mathematical Psychology*. Wiley, New York, pp. 323–418.

Chomsky, Noam (1965): *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Massachusetts.

Culy, Christopher (1985): 'The Complexity of the Vocabulary in Bambara', *Linguistics and Philosophy* **8**, 345–351.

Lasnik, Howard (2000): *Syntactic Structures Revisited*. MIT Press, Cambridge Massachusetts.

- Partee, Barbara H., Alice ter Meulen & Robert E. Wall (1990): *Mathematical Methods in Linguistics*. Vol. 30 of *Studies in Linguistics and Philosophy*, Kluwer, Dordrecht.
- Pinker, Steven (1995): *The Language Instinct*. Harper-Perennial, New York.
- Shieber, Stuart (1985): 'Evidence Against the Context-freeness of Natural Language', *Linguistics and Philosophy* **8**, 333–343.