Die Chomsky–Hierarchie

Hans U. Simon (RUB)

Email: simon@lmi.rub.de

Homepage: http://www.ruhr-uni-bochum.de/lmi

Vorgeplänkel: Mathematische Grundlagen

Voraussetzung: Wissen aus mathematischen Grundvorlesungen

hier: weitere Grundlagen zu Wörtern, Sprachen und Relationen

Alphabet und Zeichen

Alphabet = endliche Menge

Die Elemente eines Alphabets Σ werden als Zeichen, Symbole oder Buchstaben bezeichnet.

Beispiele:

1. $\Sigma = \{a, \ldots, z\} \cup \{A, \ldots, Z\} \cup \{0, \ldots, 9\}$.
2. $\Sigma = \{a, b\}$.

Zeichenfolgen

$\Sigma^+ =$ Menge der nicht–leeren Zeichenfolgen über Alphabet Σ.

Die Elemente von Σ^+ werden auch als Wörter, Sätze oder Strings bezeichnet.

Die Länge eines Strings w ist gegeben durch

$$|w| = \text{Anzahl der Zeichen (mit Vielfachheit) in } w.$$

Für ein Zeichen $a \in \Sigma$ schreiben wir

$$|w|_a = \text{Anzahl der Vorkommen von } a \text{ in } w.$$

Beispiel:

- $\{a, b\}^+ = \{a, b, aa, ab, ba, bb, aaa, \ldots\}$.
- $|a| = |ba|_a = 1$, $|ba| = 2$, $|aaa| = 3$.

Zeichenfolgen (fortgesetzt)

Das leere Wort der Länge 0 wird mit ε bezeichnet.
(Spielt eine ähnliche Rolle wie die „Null“ beim Addieren.)

$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$ bezeichnet dann die Menge aller Strings (inklusive dem leeren) über Alphabet Σ.

Konkatenation von Wörtern

Wörter konkateniert (= aneinandergehängt) ergeben wieder Wörter. Es gilt:

$$|uv| = |u| + |v|$$

Beispiel: Für $u = ab$ und $v = aab$ gilt $uv = abaab$ und $|uv| = 2 + 3 = 5$.

Die n–fache Konkatenation eines Wortes w mit sich selbst notieren wir mit w^n. Es gilt $|w^n| = n \cdot |w|$.

Beispiel: Für $w = ab$ gilt $w^3 = ababab$ und $|w^3| = 3 \cdot 2 = 6$.

Formale Sprachen

Eine Menge $A \subseteq \Sigma^*$ heißt (formale) Sprache über Alphabet Σ. Neben den üblichen Mengenoperationen (Vereinigung, Durchschnitt, Komplement, Mengendifferenz,...) sind auf Sprachen die folgenden Operationen von Interesse:

Konkatenation $AB := \{uv |\ u \in A, v \in B\}$.

Potenz $A^n := \underbrace{A \cdots A}_{n-mal}$, wobei $A^0 = \{\varepsilon\}$ und $A^1 = A$.

Kleenescher Abschluss $A^+ = \cup_{n \geq 1} A^n$ und $A^* = \cup_{n \geq 0} A^n$.

Formale Sprachen (fortgesetzt)

Beispiel Betrachte die Sprachen

$$A = \{w \in \{a,b\}^* |\ |w|_a = 5\} \text{ und } B = \{w \in \{a,b\}^* |\ |w|_b = 5\}\ .$$

Dann gilt

$$w = \overbrace{baaabbbabbbbb}^{\in A}\overbrace{aaabaabbbbaabaaa}^{\in B} \in AB\ .$$

Weiter gilt:

$$\begin{aligned} A^{20} &= \{w \in \{a,b\}^* |\ |w|_a = 100\} \\ A^{+} &= \{w \in \{a,b\}^+ |\ 5 \text{ ist Teiler von } |w|_a\} \end{aligned}$$

Relationen

Wir betrachten Relationen R über einer Grundmenge M.

Formal: $R \subseteq M \times M$.

Statt $(x, y) \in R$ schreiben wir oft xRy.

Statt $(x, y) \in R$ und $(y, z) \in R$ schreiben wir mitunter $xRyRz$.

Verknüpfung von Relationen

Ähnlich wie bei formalen Sprachen betrachten wir folgende Operationen:

Konkatenation $RS := \{(x,y) \in M \times M | \ \exists z \in M : xRz \text{ und } zSy\}$.

Potenz $R^n := \underbrace{R \cdots R}_{n-mal}$, wobei $R^0 = \{(x,x) | \ x \in M\}$ und $R^1 = R$.

(reflexive) transitive Hülle $R^+ = \cup_{n \geq 1} R^n$ und $R^* = \cup_{n \geq 0} R^n$.

Es gilt:

1. $R^n = \{(x,y) \in M \times M | \ \exists z_1, \ldots, z_{n-1} : xRz_1Rz_2 \cdots Rz_{n-1}Ry\}$ für $n \geq 1$.
2. R^+ ist die kleinste transitive R umfassende Relation.
3. R^* ist die kleinste reflexive und transitive R umfassende Relation.

Beispiele

M = Menge von Personen

R = die Relation „ist Schwester von“

S = die Relation „ist Vater oder Mutter von“

Dann gilt:

RS = die Relation „ist Tante von“

S^2 = die Relation „ist Großvater oder Großmutter von“

S^+ = die Relation „ist (echter) Vorfahr von“

Jetzt gehts ...

... zur Sache !

Grammatiken

Eine Grammatik besteht aus vier Komponenten:

- V, die Menge der Variablen
- Σ, das Terminalalphabet
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$, das Regel– oder Produktionensystem
- $S \in V$, die Startvariable

Dabei sind V, Σ, P endliche Mengen und $V \cap \Sigma = \emptyset$. Ein Paar (y, y') aus P notieren wir meist in der Form $y \to y'$.

Intuition: In einer grammatischen Ableitung darf y durch y' ersetzt werden.

Strings über Σ heißen Sätze; Strings über $V \cup \Sigma$ heißen Satzformen.

Grammatische Ableitungen

Die Notation $u \Rightarrow_G v$ bedeutet, dass Satzform u unter **einer** Regelanwendung der Grammatik G in Satzform v übergehen kann:

u,v haben die Form $u = xyz$, $v = xy'z$, und $y \to y' \in P$.

Mit Hilfe von Relation „$\Rightarrow_G$“ können folgende Relationen gebildet werden:

$$\Rightarrow_G^n \quad = \quad n\text{–fache Potenz von } \Rightarrow_G$$

$$\Rightarrow_G^+ \quad = \quad \text{transitive Hülle von } \Rightarrow_G$$

$$\Rightarrow_G^* \quad = \quad \text{reflexive–transitive Hülle von } \Rightarrow_G$$

Grammatische Ableitungen (fortgesetzt)

- $u \Rightarrow_G^n v$ bedeutet, dass Satzform u unter n Anwendungen von Regeln der Grammatik G in Satzform v übergehen kann.
- $u \Rightarrow_G^+ v$ bedeutet, dass Satzform u unter einer iterierten Anwendung (mindestens einmal) von Regeln der Grammatik G in Satzform v übergehen kann.
- $u \Rightarrow_G^* v$ bedeutet, dass Satzform u unter einer iterierten Anwendung (auch null-mal) von Regeln der Grammatik G in Satzform v übergehen kann.

Grammatische Ableitungen (fortgesetzt)

Eine Folge

$$w_0, w_1, \ldots, w_n$$

von Satzformen mit

$$w_0 = S,\ w_n \in \Sigma^* \text{ und } w_i \Rightarrow_G w_{i+1}$$

heißt Ableitung (von w_n mit Regeln aus G).

Die von G erzeugte Sprache ist die Menge aller aus Startsymbol S ableitbaren Wörter über Terminalalphabet Σ:

$$L(G) := \{w \in \Sigma^* |\ S \Rightarrow_G^* w\}$$

Beispiel: Korrekt geklammerte Ausdrücke

Betrachte die Grammatik G mit den Komponenten

- $V = \{E, T, F\}$.
- $\Sigma = \{(,), a, +, *\}$.
- P enthalte die Regeln

$$E \to T \ , \ E \to E + T \ , \ T \to F \ , \ T \to T * F \ , \ F \to a \ , \ F \to (E) \ .$$

- $S = E$, d.h., E ist die Startvariable.

Intuition: E steht für EXPRESSION (arithmetischer Ausdruck), T für TERM und F für FACTOR. Terminalzeichen a repräsentiert einen atomaren Ausdruck (etwa eine Konstante oder eine Variable). Die Grammatik soll gerade die korrekt geklammerten Ausdrücke erzeugen.

Beispiel (fortgesetzt)

Es gilt

$$a * (a + a) + a \in L(G)$$

wie folgende Ableitung zeigt:

$$\begin{array}{lllll}
E & \Rightarrow & E + T & \Rightarrow & T + T & \Rightarrow & T * F + T \\
& \Rightarrow & F * F + T & \Rightarrow & a * F + T & \Rightarrow & a * (E) + T \\
& \Rightarrow & a * (E + T) + T & \Rightarrow & a * (T + T) + T & \Rightarrow & a * (F + T) + T \\
& \Rightarrow & a * (a + T) + T & \Rightarrow & a * (a + F) + T & \Rightarrow & a * (a + a) + T \\
& \Rightarrow & a * (a + a) + F & \Rightarrow & a * (a + a) + a & &
\end{array}$$

Es wurde in jedem Schritt die am weitesten links stehende Variable ersetzt: wir sprechen von einer Linksableitung.

Beispiel (fortgesetzt)

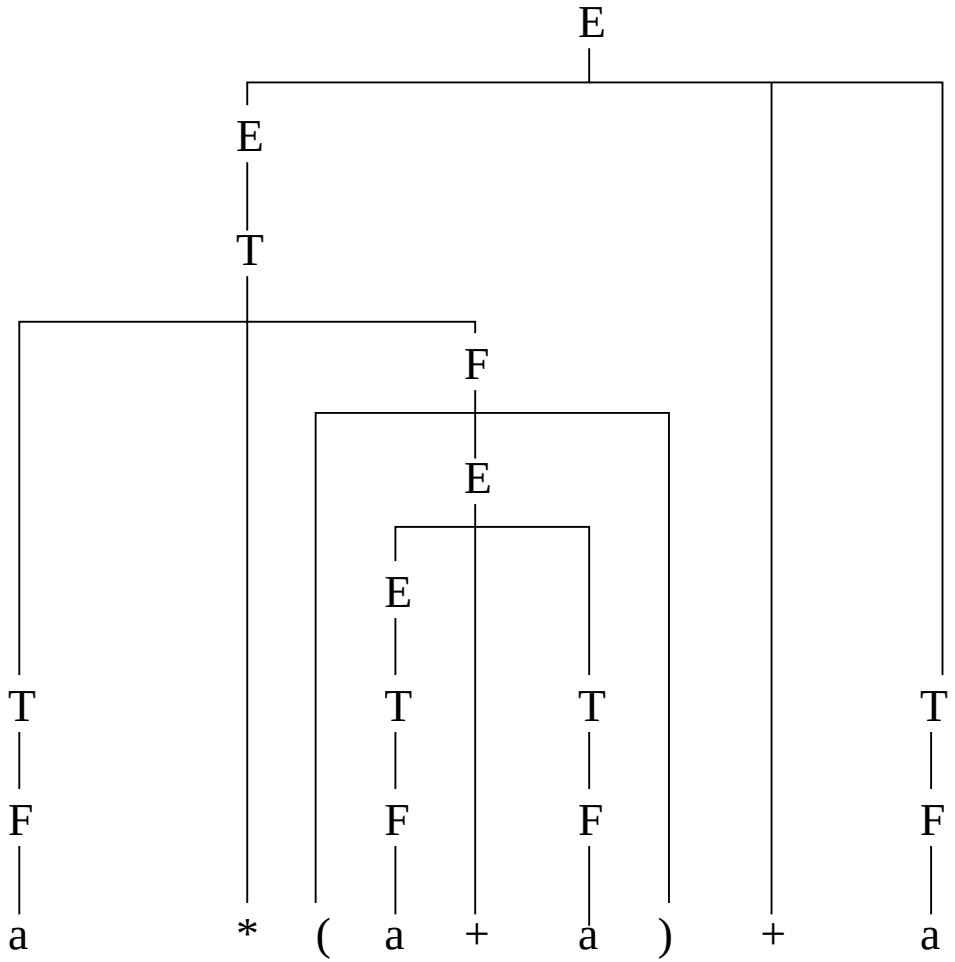

Abbildung 1: Der Syntaxbaum zur Ableitung von $a * (a + a) + a$.

Chomsky–Hierarchie der Grammatiken

Typ 0: Eine Grammatik mit Regeln der allgemeinen Form $w \to w'$ mit $w \in (V \cup \Sigma)^+$ und $w' \in (V \cup \Sigma)^*$ heißt Grammatik vom Typ 0.

Typ 1: Eine Grammatik mit Regeln der Form $w \to w'$ mit $w, w' \in (V \cup \Sigma)^+$ und $|w| \leq |w'|$ heißt kontextsensitive (oder auch Typ 1) Grammatik.

Typ 2: Eine Grammatik mit Regeln der Form $X \to w$ mit $X \in V$ und $w \in (V \cup \Sigma)^+$ heißt kontextfreie (oder auch Typ 2) Grammatik.

Typ 3: Eine Grammatik mit Regeln der Form $X \to a$ oder $X \to aY$ mit $X, Y \in V$ und $a \in \Sigma$ heißt reguläre (oder auch Typ 3) Grammatik.

Wir übertragen diese Bezeichnungen auch auf die von den Grammatiken generierten Sprachen.

Offensichtlich gilt:

$$\text{regulär} \Rightarrow \text{kontextfrei} \Rightarrow \text{kontextsensitiv} \Rightarrow \text{Typ 0} \Rightarrow \text{formale Sprache}$$

Echtheit der Chomsky–Hierarchie

1. Die Sprache $\{(a^n b^n | \ n \geq 1\}$ ist kontextfrei aber nicht regulär.
2. Die Sprache $\{a^n b^n c^n | \ n \geq 1\}$ ist kontextsensitiv aber nicht kontextfrei.
3. Es gibt überabzählbar viele formale Sprachen, aber nur abzählbar viele vom Typ 0.

Da es auch nicht kontextsensitive Sprachen vom Typ 0 gibt, sind alle Inklusionen der Chomsky–Hierarchie echt.

Echtheit der Chomsky–Hierarchie (fortgesetzt)

Die kontextfreien Regeln

$$S \to ab,\ S \to aSb$$

erzeugen die Sprache

$$\{a^n b^n |\ n \geq 1\}\ .$$

Wir werden später sehen, dass diese Sprache nicht regulär ist.

Echtheit der Chomsky–Hierarchie (fortgesetzt)

Satz: Die kontextsensitive Grammatik mit den Regeln

$$S \to aSBC,\ S \to aBC,\ CB \to BC$$
$$aB \to ab,\ bB \to bb,\ bC \to bc,\ cC \to cc$$

erzeugt die Sprache $\{a^n b^n c^n |\ n \geq 1\}$. (Beweis folgt!)

Wir werden später sehen, dass diese Sprache nicht kontextfrei ist.

Warnung

Subtile Beweisführung im Anmarsch !

Techniken

- des Erkennens von Invarianten
- der (einfachen) vollständigen Induktion
- der strukturellen vollständigen Induktion

kommen dabei zum Einsatz !

1. Beweisrichtung (vollständige Induktion)

Behauptung 1 (Beweis an der Tafel skizziert): Die Regeln

$$S \to aSBC,\ S \to aBC,\ CB \to BC$$
$$aB \to ab,\ bB \to bb,\ bC \to bc,\ cC \to cc$$

erlauben für alle $n \geq 0$ die Erzeugung von

$$a^n S \overbrace{BCBC\cdots BC}^{n\text{-mal}}$$
$$a^n \overbrace{BCBC\cdots BC}^{n\text{-mal}}$$
$$a^n B^n C^n$$
$$a^n b^n c^n \quad .$$

Formaler Beweis durch vollständige Induktion nach n (oder durch Überlegung vom Typ "..."!

2. Beweisrichtung (strukturelle vollständige Induktion)

$$S \to aSBC,\ S \to aBC,\ CB \to BC$$
$$aB \to ab,\ bB \to bb,\ bC \to bc,\ cC \to cc$$

Behauptung 2 (Beweis an der Tafel skizziert): Für jede von diesen Regeln erzeugte Satzform w gelten die folgenden Bedingungen (sogenannte Invarianzbedingungen):

1. $|w|_a = |w|_B + |w|_b = |w|_C + |w_c|$.
2. Ein Großbuchstabe steht niemals links von einem Kleinbuchstaben.
3. Symbole b, c stehen niemals links von einem a.
4. Ein Symbol c steht niemals links von einem b

Daher sind alle erzeugten Terminalstrings von der Form $a^n b^n c^n$ (mit $n \geq 1$).

Formaler Beweis durch vollständige Induktion über die Länge bzw. den Aufbau einer grammatischen Ableitung !

Sonderregelung für das leere Wort

Jetzige Definition von Typ 1,2,3 Grammatiken G erlaubt nicht die Generierung des leeren Wortes ε. Das ist schlecht, falls $\varepsilon \in L(G)$ erwünscht ist:

Erweiterung der alten Definition (eingeschränkte Verwendung von ε–Regeln):
Bei Typ 1,2,3 Sprachen mit Startsymbol S erlauben wir auch die Regel $S \to \varepsilon$.
In diesem Falle darf aber S auf keiner rechten Seite einer Regel auftreten.

Allgemeine ε–Regeln bei Typ 2,3 Grammatiken

Bei kontextfreien (oder regulären) Sprachen treiben wir es noch bunter und erlauben wir i.A. auch Regeln der Form $A \to \varepsilon$ mit $A \in V$ (sogenannte ε-Regeln).

Frage: Wird dadurch die Klasse der kontextfreien (bzw. regulären) Sprachen erweitert?

Antwort: Nein!

Begründung: Eine kontextfreie (bzw. reguläre) Grammatik G mit ε-Regeln kann stets in eine äquivalente kontextfreie (bzw. reguläre) Grammatik G' verwandelt werden, die ε-Regeln nicht (oder nur in der eingeschränkten Form) verwendet.

Beweis wird später (in Verbindung mit der Chomsky-Normalform) nachgeliefert.

Das Wortproblem

Das Wortproblem für eine Sprache $L \subseteq \Sigma^*$ ist folgendes Problem:

Eingabe: $w \in \Sigma^*$

Frage: $w \in L$?

Es heißt entscheidbar, wenn ein Algorithmus existiert, der für jede Eingabe w die richtige Antwort liefert.

Satz: Das Wortproblem für jede kontextsensitive Sprache L ist entscheidbar.

Idee: Verwende eine kontextsensitive Grammatik G, die L generiert.

Wegen der Monotonie–Eigenschaft

„rechte Seite einer Regel ist nicht kürzer als die linke Seite“

sind nur die endlich vielen Satzformen mit einer Maximallänge von $n = |w|$ für die Ableitung von w relevant.

Das Wortproblem (fortgesetzt)

Implementierung der Idee:

1. Setze $n := |w|$.
2. Berechne die Menge Abl_n aller Satzformen der Maximallänge n, die sich aus S ableiten lassen:

$$\text{Abl}_n := \{w \in (V \cup \Sigma)^* | \ |w| \leq n \text{ und } S \Rightarrow_G^* w\}$$

3. Falls $w \in \text{Abl}_n$, dann akzeptiere w; andernfalls verwerfe w.

Dabei kann Abl_n iterativ berechnet werden wie folgt:

Initialisierung $\text{Abl} := \{S\}$.

Iteration Solange ein Wort $w \notin \text{Abl}$ mit

$$|w| \leq n, \exists u \in \text{Abl} : u \Rightarrow_G w$$

existiert, nimm auch w in Abl auf.

Syntaxbäume

Betrachte wieder eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$. Jede Regel $B \to A_1 \cdots A_k$ kann durch eine Verzweigung visualisiert werden:

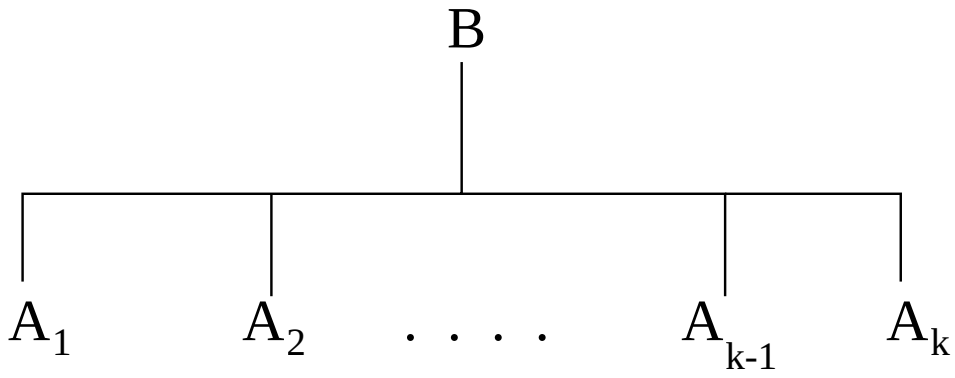

Abbildung 2: Visualisierung einer Regel $B \to A_1 \cdots A_k$.

Syntaxbäume (fortgesetzt)

Ein Syntaxbaum zur kontextfreien Grammatik G ist ein geordneter Wurzelbaum mit folgenden Eigenschaften:

1. Die Wurzel ist mit dem Startsymbol S markiert.
2. Die inneren Knoten sind mit Variablen aus V markiert.
3. Die Blätter sind Terminalzeichen aus Σ (oder mit ε) markiert.
4. Jede Verzweigung entspricht einer Regel aus P.

Als Beschriftung eines Syntaxbaumes T bezeichnen wir das Wort aus Σ^*, das sich aus den Markierungen der Blätter, gelesen von links nach rechts, zusammensetzt.

Syntaxbäume (fortgesetzt)

Folgende Aussagen sind äquivalent:

1. Es gibt eine Ableitung von w mit Regeln aus G.
2. Es gibt einen Syntaxbaum zu G mit Beschriftung w.
3. Es gibt eine Linksableitung von w mit Regeln aus G.

Ein formaler Beweis könnte durch einen Ringschluss

$$1. \Rightarrow 2. \Rightarrow 3. \Rightarrow 1.$$

erfolgen.

(Illustration an der Tafel)

Backus–Naur–Form

Backus–Naur–Form (BNF) erlaubt flexiblere Formen von kontextfreien Regeln:

1. Die Metaregel

$$A \to \beta_1|\beta_2|\cdots|\beta_n$$

(unter Verwendung des „Metasymbols" |) steht für

$$\begin{array}{lcl} A & \to & \beta_1 \\ A & \to & \beta_2 \\ \cdots & & \cdots \\ A & \to & \beta_n \end{array}$$

Dadurch lassen sich alle A-Regeln in einer Zeile zusammenfassen.

2. $A \to \alpha[\beta]\gamma$ steht für $A \to \alpha\gamma|\alpha\beta\gamma$:
man darf β zwischen α und γ einfügen, muss es aber nicht.

3. $A \to \alpha\{\beta\}\gamma$ steht für $A \to \alpha\gamma|\alpha B\gamma, B \to \beta|\beta B$: das Wort β kann zwischen α und γ iteriert (auch null–mal) eingefügt werden.

Beispiel zur Backus–Naur–Form

Die Regeln

$$E \to T \ , \ E \to E+T \ , \ T \to F \ , \ T \to T*F \ , \ F \to a \ , \ F \to (E)$$

für korrekt geklammerte arithmetische Ausdrücke können in BNF kompakt notiert werden wie folgt:

$$E \to T|E+T \ , \ T \to F|T*F \ , \ F \to a|(E)$$

Oder noch kompakter:

$$E \to [E+]T \ , \ T \to [T*]F \ , \ F \to a|(E)$$

Eindeutige und mehrdeutige Grammatiken

Eine kontextfreie Grammatik G heißt mehrdeutig, wenn sie verschiedene Syntaxbäume mit der gleichen Beschriftung zulässt; andernfalls heißt sie eindeutig.

Eine kontextfreie Sprache L heißt eindeutig, wenn eine L generierende eindeutige kontextfreie Grammatik G existiert; andernfalls heißt sie inhärent mehrdeutig.

Programmiersprachen sollten eindeutig sein, damit jedes Programm eindeutig interpretiert werden kann.

Mehrdeutige Beispiel-Grammatik

Die Regeln

$$K \to () \mid KK \mid (K)$$

erzeugen die Sprache der (nichtleeren) korrekten Klammerausdrücke.

Korrekt geklammert: () , (()())()

Falsch geklammert: (() ,)(

Demonstration der Mehrdeutigkeit

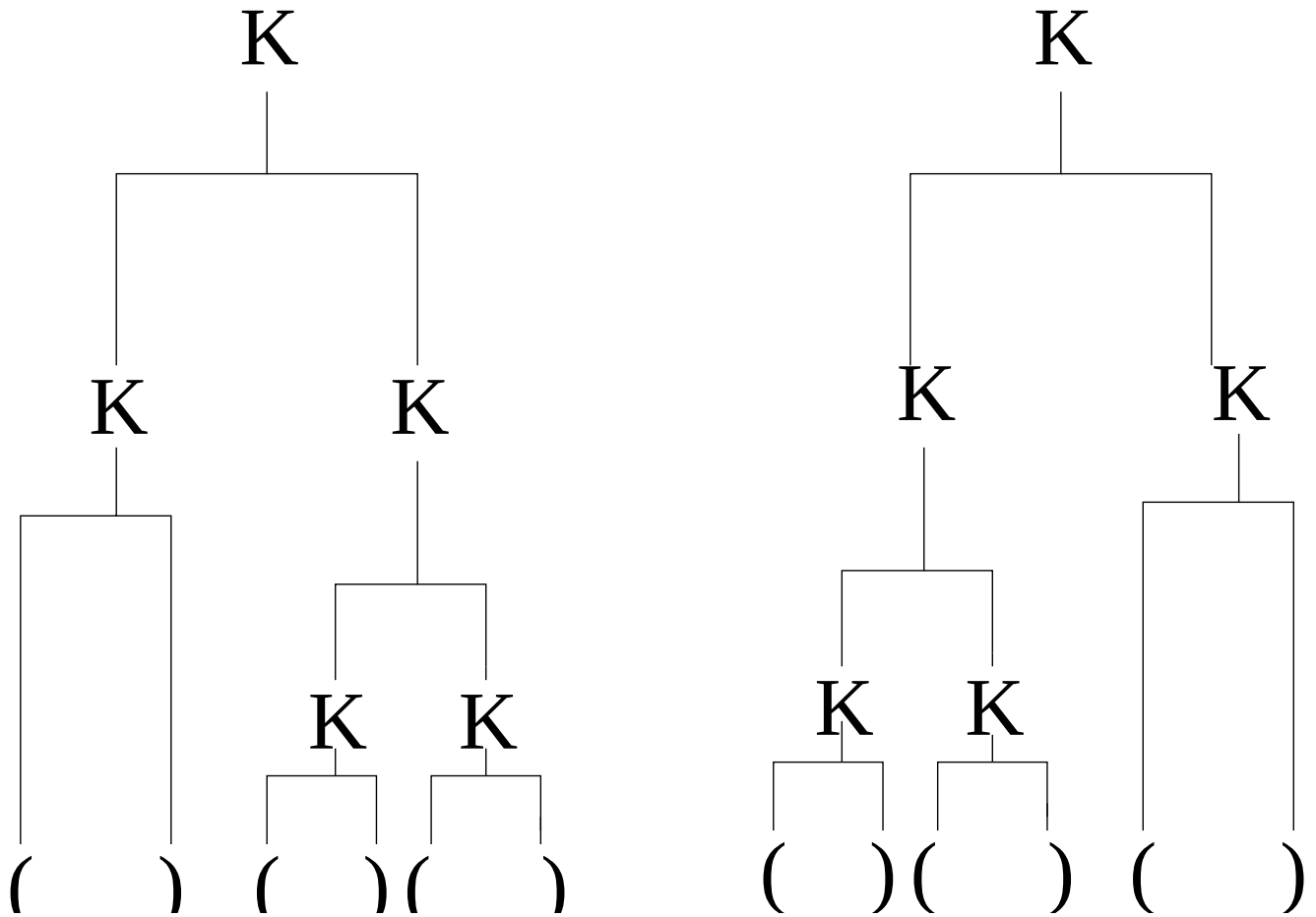

Abbildung 3: Zwei verschiedene Syntaxbäume für den Klammerausdruck ()()().

Eindeutige Grammatik für dieselbe Sprache

$$K \to A \mid AK \ , \ A \to () \mid (K)$$

Intuition: K = Klammerausdruck, A = atomarer Klammerausdruck.

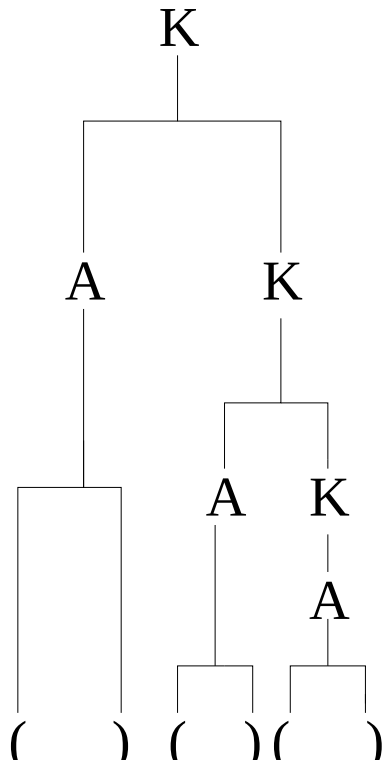

Abbildung 4: Eindeutiger Syntaxbaum für den Klammerausdruck ()()().

Beispiel einer inhärent mehrdeutigen Sprache

Die Sprache

$$L = \{a^i b^j c^k | \ i = j \text{ oder } j = k\}$$

ist inhärent mehrdeutig (ohne Beweis).

Exemplarische Lernziele zur Chomsky–Hierarchie

- Grundbegriffe kennen (Vokabeln lernen!) und intellektuell beherrschen
- bei Operationen auf Sprachen die Ergebnissprache beschreiben
- bei Operationen auf Relationen die Ergebnisrelation beschreiben
- zu einer gegebenen Sprache eine passende Grammatik finden
- zu einer gegebenen Grammatik die davon erzeugte Sprache „intelligent erraten“
- zu einem Wort aus einer Sprache die passende grammatische Ableitung (im kontextfreien Fall auch den Syntaxbaum) finden
- Mehrdeutigkeit einer kontextfreien Grammatik erkennen und, sofern möglich, vermeiden können
- Invarianten erkennen und mit vollständiger (evtl. struktureller) Induktion beweisen können