Theoretische Informatik
WS 2024/2025

Kapitel 1

Formale Sprachen

1.2

Die Chomsky Hierarchie von Grammatiken

Prof. Dr. Robert Preis
Fachbereich Informatik
Fachhochschule Dortmund
Robert.Preis@fh-dortmund.de

Eine Hierarchie von Grammatiken und Sprachen

Die Regeln von Grammatiken können unterschiedlich kompliziert sein:

Typ 0-Grammatiken	Typ 0-Sprachen
Allgemeine Regeln z.B. SaT → bXbY	z.B. $a^{(2^n)}$ ={aa, aaaa, aaaaaaaa,…}

Typ 1-Grammatiken	Typ 1-Sprachen
Nur Kontextsensitive Regeln z.B. aSb → abXaYb	z.B. $a^nb^nc^n$ = {abc, aabbcc, …}

Typ 2-Grammatiken	Typ 2-Sprachen
Nur Kontextfreie Regeln z.B. S → aXbY	z.B. a^nb^n = {ab, aabb, aaabbb, …} oder $(^n)^n$ = {(), (()), ((())), …}

Typ 3-Grammatiken	Typ 3-Sprachen,
Nur Reguläre Regeln z.B. S → aX	z.B. $(ab)^n$ = {ab, abab, ababab, …}

Es gilt für Sprachen: Typ 0 ⊃ Typ 1 ⊃ Typ 2 ⊃ Typ 3 !

Typ 0: Allgemeine Grammatiken

Keine Einschränkung an den Regeln: $X \to Y$ ($X \in \Gamma^+ \setminus T^*$, $Y \in \Gamma^*$), d.h.

- Links: $X \in \Gamma^+ \setminus T^*$ Links stehen Variablen und Terminale, aber mindestens eine Variable muss dabei sein.
- Rechts: $Y \in \Gamma^*$ Rechts stehen Variablen und Terminale, es kann aber auch das leere Wort sein.

$G = (\{S,A,B,C,D,E\}, \{a\}, P, S)$ mit

$P = \{$	$S \to ACaB,$	1) generiere Grenzen A und B
	$Ca \to aaC,$	2) verdopple rechtes ‚a‘
	$CB \to DB,$	3) am rechten Ende angekommen
	$aD \to Da,$	4) gehe zurück nach links
	$AD \to AC,$	5) links angekommen, Neuanfang
	$CB \to E,$	6) Ende, lösche rechten Rand
	$aE \to Ea,$	7) gehe nach links um…
	$AE \to \varepsilon \}$	8) …auch linken Rand zu löschen

Sprache von G = L(G) = {aa,aaaa,aaaaaaaa,…} = $\{a^2, a^4, a^8, a^{16}, a^{32}, \ldots\}$

Was bedeutet „kontextsensitiv“?

Was ist eine „Bank“ ?

Beispiele:

Ein Mann geht in die Stadt zu seiner Bank und hebt Geld ab.

Ein Mann geht in den Park und setzt sich auf eine Bank.

Die Bedeutung von Bank kann unterschiedlich sein (Teekesselchen), es hängt davon ab, in welchem **Kontext** der Begriff verwendet wird!

Wir nennen das: ***kontextsensitiv***

Typ 1: Kontextsensitive Grammatiken

Nur Regeln der Form

- $XAY \to XZY$ $A \in V$, $X,Y \in \Gamma^*$, $Z \in \Gamma^+$ ($z \neq \varepsilon$, das Wort wird nie kürzer!) oder
- $S \to \varepsilon$ Das ist nur bei Startvariable möglich, die dann aber nirgends auf der rechten Seite sein darf),

d.h.

- Die Variable $A \in V$ wird ersetzt durch den (nicht leeren) Ausdruck $Z \in \Gamma^+$.
- Aber nur, wenn es kontextsensitiv ist, d.h.
  - links und rechts vom A müssen Ausdrücke $X \in \Gamma^*$ und $Y \in \Gamma^*$ stehen (Wir nennen es *Kontext*, falls es nicht das leere Wort ist).
  - Sowohl X als auch Y müssen auch rechts stehen bleiben!

$G = (\{S,A\}, \{a, b\}, P, S)$ mit
$P = \{$ $S \to aAa \mid bAb,$
$aAa \to abAba \mid aba,$
$babA \to babaAa \mid baba\}$

Eine Ableitung wird niemals kürzer!

…es gibt auch andere, ähnliche Definitionen in der Literatur…

Typ 2: Kontextfreie Grammatiken

Nur Regeln der Form $A \to X$ ($A \in V$, $X \in \Gamma^*$), d.h.

- $A \in V$: Links steht genau eine Variable, nichts anderes.
- $X \in \Gamma^*$: Rechts stehen Variablen und Terminale oder das leere Wort.

Beispiele:

- $\{a^n b^n \mid n \in \mathbf{N}_+\}$:
  $G = (\{S\}, \{a, b\}, P, S)$ mit $P = \{S \to aSb \mid ab\}$

- {alle korrekten Klammerausdrücke}:
  $G = (\{S\}, \{(,)\}, P, S)$ mit $P = \{S \to SS \mid (S) \mid ()\}$

- {alle arithmetischen Ausdrücke über $\mathbf{N}_0$}:
  $G = (\{E, Z\}, \{0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,)\}, P, E)$ mit
  $P = \{E \to E+E|E-E|E*E|E/E|(E)|0|1|2|3|4|5|6|7|8|9|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z,$
  $Z \to 0|1|2|3|4|5|6|7|8|9|0Z|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z\}$

Mehrere Ableitungen

Kann es mehrere Ableitungen für dasselbe Wort geben?

G = ({E, Z}, {0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,)}, P, E) mit
P = {E → E+E|E-E|E∗E|E/E|(E)|0|1|2|3|4|5|6|7|8|9|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z,
Z → 0|1|2|3|4|5|6|7|8|9|0Z|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z}

Ableitungen für „2+3*4“?

- E→E*E→E+E*E→2+E*E→2+3*E→2+3*4
- E→E+E→E+E*E→2+E*E→2+3*E→2+3*4
- E→E+E→2+E→2+E*E→2+E*4 →2+3*4
- …

Macht es einen Unterschied?

Es kommt immer dasselbe Wort heraus …

Es wird aber manchmal anders hergestellt …

Es gibt Unterschiede, ob erst „+“ oder erst „“ …*

Kann dasselbe Wort eine unterschiedliche Bedeutung haben?

Mehrdeutigkeit

Was bedeutet der folgende Satz?

Der Mann sagt die Frau kann nicht Auto fahren.

Wer sagt etwas und wer kann etwas nicht?

Der Satz ist mehrdeutig !

Ableitungsbäume für kontextfreie Grammatiken

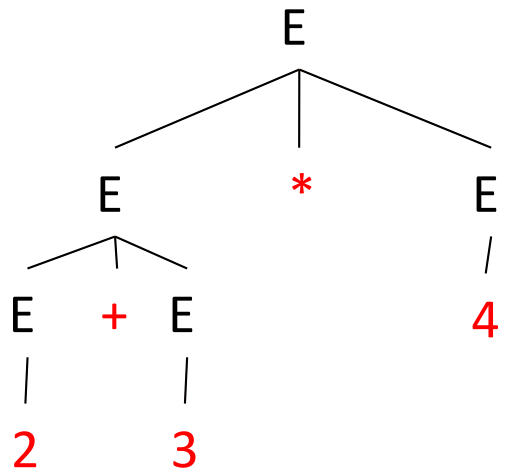
Bei kontextfreien Grammatiken, d.h. wenn nur Regeln der Form

$$A \rightarrow X \quad (A \in V, X \in \Gamma^*)$$

existieren, wird immer genau eine Variable ersetzt.

Dadurch können wir Ableitungen als Ableitungsbäume darstellen:

- Wurzel (Startvariable)
- Innere Knoten (Variablen)
- Blätter (Terminale)



Die Blätter repräsentieren Terminalwörter:

- Auslesen durch Tiefensuche von links nach rechts

Eine Ableitung ist die Frage nach dem Wort, ein Ableitungsbaum ist die Frage nach der Struktur des Wortes (bezüglich der Regeln):

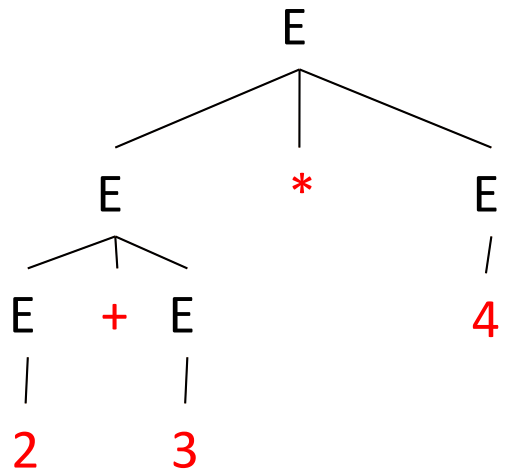
Ableitungsbäume geben daher nicht nur an, ob ein Wort von einer Grammatik erzeugt werden kann, sondern auch wie es strukturiert ist.

Die Ableitungen eines Ableitungsbaumes

G = ({E, Z}, {0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,)}, P, E) mit
P = {E → E+E|E-E|E∗E|E/E|(E)|0|1|2|3|4|5|6|7|8|9|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z,
Z → 0|1|2|3|4|5|6|7|8|9|0Z|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z}

Ein Ableitungsbaum kann meistens mehrere Ableitungen haben:

1. Immer die linke Variable zuerst:
   E→E*E→E+E*E→2+E*E →2+3*E→2+3*4
2. Immer die rechte Variable zuerst:
   E→E*E→E*4→E+E*4→E+3*4→2+3*4
3. Durcheinander:
   E→E*E→E*4→E+E*4→2+E*4→2+3*4



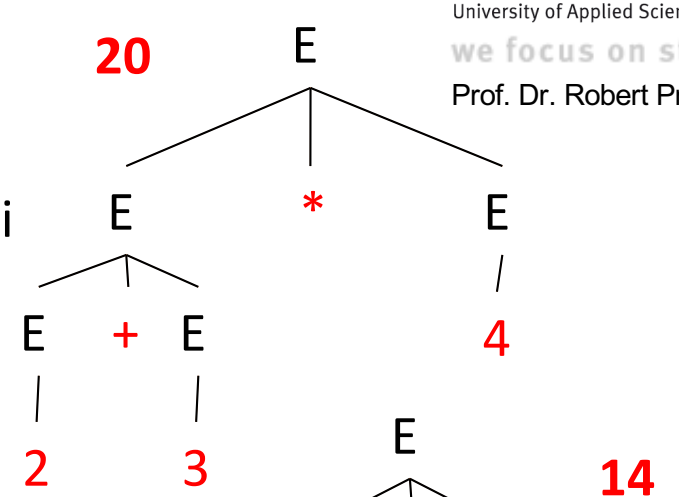
Die Ableitungen sind zwar unterschiedlich, aber von der Struktur her sind sie alle gleich:

Bei allen wird erst „+“-gerechnet und erst danach „“ gerechnet.*

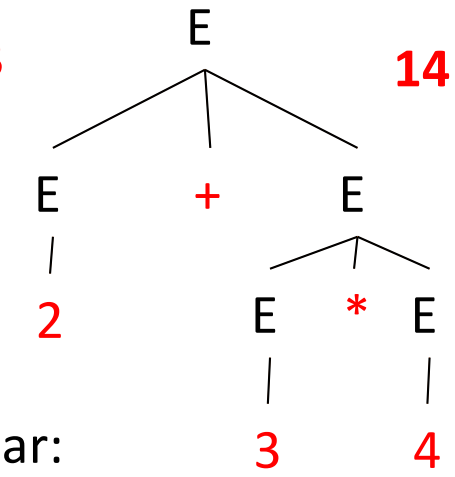
Gibt es immer nur einen Ableitungsbaum?

Nein, denn die Grammatik hat z.B. für 2+3*4 zwei verschiedene Ableitungsbäume:

- E→E*E→E+E*E→2+E*E→2+3*E→2+3*4



- E→E+E→E+E*E→2+E*E→2+3*E→2+3*4



Eine Ableitungsbaum stellt eine Menge von Ableitungen dar:

Linker Baum: Alle Ableitungen mit +- vor *-Rechnung

Rechter Baum: Alle Ableitungen mit *- vor +-Rechnung

Problem: Das Wort 2+3*4 und damit die Grammatik G sind **mehrdeutig**, d.h. dem Wort kann man die Reihenfolge der Auswertung nicht ansehen. Das erkennt man daran, dass es keinen eindeutigen Ableitungsbaum gibt.

Problem der Mehrdeutigkeit

Wortproblem:

Gehört das Wort x zu der Sprache L , d.h. x ∈ L?

Wortbedeutungsproblem:

Was bedeutet das Wort x, d.h. nach welcher Struktur ist es aufgebaut?

Eindeutige Grammatik G = (V , T , P, S)

- Jedes Wort w ∈ L(G) hat genau einen Ableitungsbaum.
- Andernfalls ist G mehrdeutig (mindestens ein w ∈L(G) hat mindestens zwei verschiedene Ableitungsbäume)

Eindeutige Sprache L

- Es gibt eine eindeutige Grammatik G mit L = L(G).
- Andernfalls ist L mehrdeutig (eine eindeutige Grammatik für L kann nicht angegeben werden).

Programmiersprachen müssen eindeutig sein !

Mehrdeutigkeit auflösen

Was bedeutet der folgende Satz?

Der Mann sagt die Frau kann nicht Auto fahren.

Und wie löst man es auf?

Durch Kommasetzung !

Variante 1:

Der Mann, sagt die Frau, kann nicht Auto fahren.

Variante 2:

Der Mann sagt, die Frau kann nicht Auto fahren.

Bei den arithmetischen Termen können wir durch Klammern die Eindeutigkeit erreichen...

Auflösung von Mehrdeutigkeiten

Bei der Grammatik

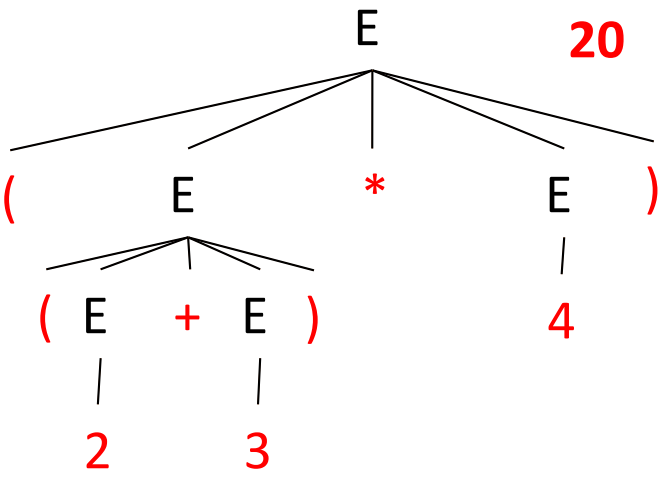
G = ({E, Z}, {0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,)}, P, E) mit

P = {E → E+E|E-E|E∗E|E/E|(E)|0|1|2|3|4|5|6|7|8|9|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z,
Z → 0|1|2|3|4|5|6|7|8|9|0Z|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z}

werden zwar alle korrekten arithmetischen Ausdrücke erstellt, es wird aber nicht die Punkt-vor-Strichrechnung bei der Abarbeitung erzwungen!

Mehrdeutig: 2+3*4

Eindeutig: ((2+3)*4) (2+(3*4))



Beispiel für eindeutige Grammatik:

G = ({E, Z}, {0,1,2,3,4,5,6,7,8,9,+,-,*,/,(,)}, P, E) mit

P = {E → (E+E)|(E-E)|(E∗E)|(E/E)|(E)|
0|1|2|3|4|5|6|7|8|9|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z,
Z → 0|1|2|3|4|5|6|7|8|9|0Z|1Z|2Z|3Z|4Z|5Z|6Z|7Z|8Z|9Z}

Typ 3: Reguläre Grammatiken

Regulär, d.h. entweder

1. Rechtslinear: nur Regeln der Form $A \to \varepsilon$ oder $A \to aB$ ($A,B \in V$, $a \in T$)
2. Linkslinear: nur Regeln der Form $A \to \varepsilon$ oder $A \to Ba$ ($A,B \in V$, $a \in T$)

d.h.

- A: Links steht genau eine Variable, nichts anderes.
- ε, aB (Ba): rechts steht entweder ε oder genau zwei Zeichen (immer Terminal/Variable oder immer Variable/Terminal).

Rechtslinear:

$G_R = (\{S,T\}, \{a,b\}, P_R, S)$ mit

$P_R = \{ \quad S \to aT \mid \varepsilon,$
$\quad\quad\quad T \to bS \}$

Linkslinear:

$G_L = (\{S,T\}, \{a,b\}, P_L, S)$ mit

$P_L = \{ \quad S \to Tb \mid \varepsilon,$
$\quad\quad\quad T \to Sa \}$

Sprache von G_R = Sprache von G_L = $\{(ab)^n \mid n \in \mathbf{N}_0\}$

…wir werden immer **rechtslineare** Grammatiken behandeln!

Das Schema von regulären (rechtslinearen) Grammatiken

Rechtslinear: $G_R = (\{S,T\}, \{a,b\}, P_R, S)$ mit

$P_R = \{ \quad S \rightarrow aT \mid \varepsilon,$

$\quad\quad\quad\quad T \rightarrow bS \}$

Die Ableitung von rechtslinearen Grammatiken folgt immer einem Schema:

- Anfang mit dem Startsymbol S
- 2 Möglichkeiten:
  1. A→aB: ein Buchstabe wird links erzeugt und es bleibt immer rechts eine Variable
  2. A→ε: die Variable verschwindet und Ende.

Beispiel mit obiger Grammatik:

$\underline{S} \rightarrow a\underline{T} \rightarrow ab\underline{S} \rightarrow aba\underline{T} \rightarrow abab\underline{S} \rightarrow abab$

Der Ablauf ist relativ einfach, es gibt immer genau eine Variable und wir müssen uns nichts Zusätzliches merken (kein Speicher nötig).

Ableitungsbäume von regulären Grammatiken

Rechtslinear (Linkslinear): Während der Ableitung gibt es immer genau eine Variable und diese steht immer rechts (links).

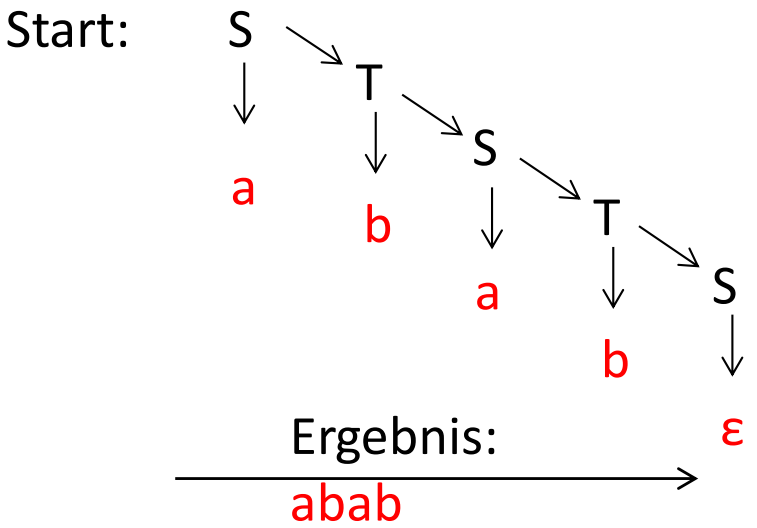
Rechtslinear:

$G_R = (\{S,T\}, \{a,b\}, P, S)$ mit

$P_R = \{ \quad S \rightarrow aT \mid \varepsilon,$

$\qquad\qquad T \rightarrow bS \}$

$S \rightarrow aT \rightarrow abS \rightarrow abaT \rightarrow ababS \rightarrow abab$



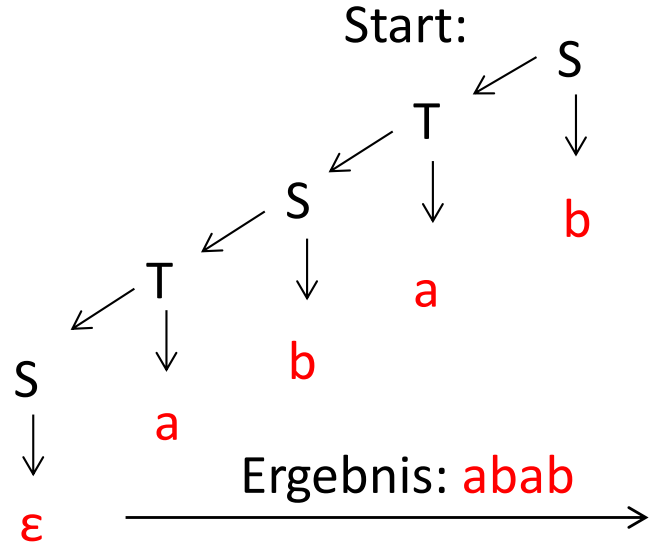
Linkslinear:

$G_L = (\{S,T\}, \{a,b\}, P, S)$ mit

$P_L = \{ \quad S \rightarrow Tb \mid \varepsilon,$

$\qquad\qquad T \rightarrow Sa \}$

$S \rightarrow Tb \rightarrow Sab \rightarrow Tbab \rightarrow Sabab \rightarrow abab$

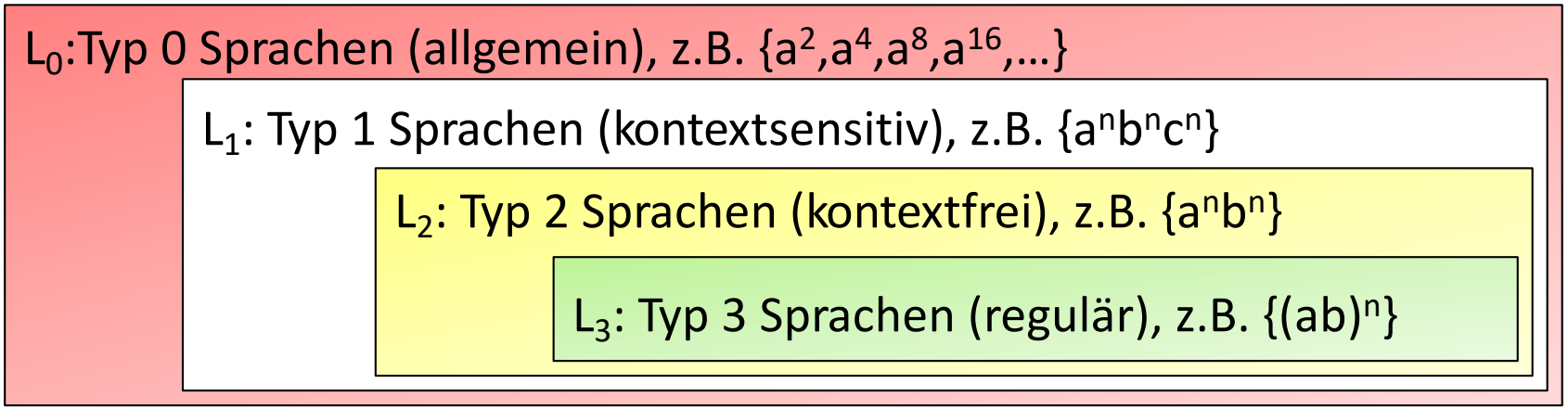


Beide Grammatiken erzeugen dieselbe Sprache!

Sprachklassen

- L_0: Typ-0 Sprachen (allgemeine Sprachen)
  Sprachen der Form L = L(G) für eine beliebige Grammatik G
- L_1: Typ-1 Sprachen (kontextsensitive Sprachen)
  Sprachen der Form L = L(G) für eine kontextsensitive Grammatik G
- L_2: Typ-2 Sprachen (kontextfreie Sprachen)
  Sprachen der Form L = L(G) für eine kontextfreie Grammatik G
- L_3: Typ-3 Sprachen (reguläre Sprachen)
  Sprachen der Form L = L(G) für eine rechtslineare Grammatik G

$L_i \equiv \{ L \mid L \text{ ist Sprache vom Typ i} \}$



Kaskade der Sprachklassen

Grammatiken:

- Typ-0: $X \rightarrow Y$ $(x \in \Gamma^+ \backslash T^*, Y \in \Gamma^*)$
- Typ-1: $XAY \rightarrow XZY$, $S \rightarrow \varepsilon$ $(A \in V, X,Y \in \Gamma^*, Z \in \Gamma^+)$
- Typ-2: $A \rightarrow X$ $(A \in V, X \in \Gamma^*)$
- Typ-3: $A \rightarrow \varepsilon$, $A \rightarrow aB$ $(A,B \in V, a \in T)$

Deshalb gilt für **Grammatiken**:

$$\text{Typ-3} \subset \text{Typ-2} \quad \text{und} \quad \text{Typ-1} \subset \text{Typ-0}$$

Es gilt nicht unbedingt Typ-2 $\subset$ Typ-1 !

Aber: *Man kann jede Typ-2 Grammatik in eine Typ-1 Grammatik umwandeln, ohne dass die Sprache verändert wird ! D.h. auch* $L_2 \subset L_1$!

Achtung: das gilt nicht unbedingt anders herum, werden wir später sehen…

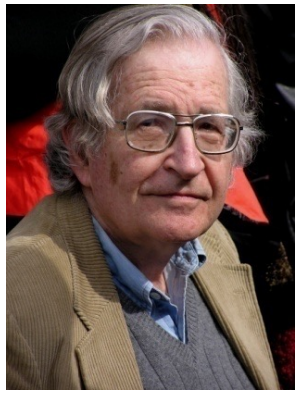
Deshalb gilt für **Sprachen**: $L_3 \subset L_2 \subset L_1 \subset L_0$

Typ2-Grammatik in Typ1-Grammatik umwandeln

1. Falls es eine Regel S-> ε (S die Startvariable) gibt und S irgendwo auf der rechten Seite auftaucht, dann (zu beliebiger Zeit)
   - neue Startvariable S´ einführen und
   - S´-> S hinzufügen.
2. Falls es eine Regel X-> ε (X keine Startvariable) gibt, dann
   - diese Regel löschen und
   - alle Regeln mit X auf der rechten Seite: doppeln und dabei das X weglassen

Typ2:	**Fall1:**	**Fall2 S->ε:**	**Fall2 A->ε:**	**Fall2 B->ε:**
	S´-> S	S´->S|ε	S´-> S|ε	S´-> S|ε
S->ABC|Sc|ε	S->ABC|Sc|ε	S->ABC|Sc|c	S->ABC|Sc|c|BC	S->ABC|Sc|c|BC|AC|C
A->SBab|ε	A->SBab|ε	A->SBab|ε|Bab	A->SBab|Bab	A->SBab|Bab|Sab|ab
B->b|ab|S|ε	B->b|ab|S|ε	B->b|ab|S|ε	B->b|ab|S|ε	B->b|ab|S
C->bc	C->bc	C->bc	C->bc	C->bc

Die Chomsky-Hierarchie zur Klassifizierung von Grammatiken



Avram Noam Chomsky

geb. 7.12.1928 in Philadelphia, USA

Professor für Linguistik am

Massachusetts Institute of Technology (MIT)

→ **http://de.wikipedia.org/wiki/Noam_Chomsky**

Typ	Typ-3 ⊂	Typ-2 ⊂	Typ-1 ⊂	Typ-0
Grammatik	$X \rightarrow aY$	$X \rightarrow aYXbY$	$aXb \rightarrow aYb$	$aXYb \rightarrow aXb$
Sprache	regulär	kontextfrei	kontextsensitiv	allgemein
Beispiel	$(ab)^n$	$a^n b^n$	$a^n b^n c^n$	$\{a^2, a^4, a^8, a^{16} \ldots\}$
Maschine	Endlicher Automat	Nichtdet. Kellerautomat	Linear beschränkter Automat	Turing-maschine

dieses Semester

dieses Semester

Einführung dieses Semester

Vorausblick Sprachklassen (später)

	Typ-3	Typ-2	Typ-1	Typ-0
Frage 1: L={} ?	√	√	↯	↯
Frage 2: w∈L ?	√	√	√	↯
Frage 3: $L_1=L_2$?	√	↯	↯	↯
Frage 4: minimal ?	√	↯	↯	↯
$L_1 \cup L_2$	√	√	√	√
$L_1 \cap L_2$	√	↯	√	√
$L_1 \setminus L_2$	√	↯	√	↯
L^c	√	↯	√	↯
$L_1 \circ L_2$	√	√	√	√
L^*	√	√	√	√

Typ-0 Sprachen können (fast) alles, sind aber sehr schwer zu analysieren.

Typ-3 Sprachen können nicht alles, sind aber sehr gut zu analysieren.

Probleme bei regulären (rechtslinearen) Grammatiken (Typ 3)

Bei regulären (rechtslinearen) Grammatiken wie z.B.

G = ({S,T}, {a,b}, P, S) mit P = { S → aS | aT | ε, T → bS }

gibt es 2 Probleme:

1. Auswahl: Wenn es Regeln A→aB und A→aC gibt, dann gibt es immer zwei mögliche Regeln, die ich als nächstes benutzen kann.
2. Sackgasse: Wenn man bei Variable A das Terminal a als nächstes erzeugen will, es aber keine Regel A→aB gibt.

Wir machen es uns jetzt noch einmal einfacher und sagen,
dass es die beiden Probleme nicht geben darf!

1. Regeln A→aB und A→aC sind nicht gleichzeitig erlaubt, d.h. es ist immer klar, welche Regel als nächstes angewendet wird !
2. Für jede Variable A und jedes Terminal a gibt es eine Regel A→aB, d.h. es gibt immer eine Regel, die man anwenden kann !

Ergo: Es gibt für das nächste Zeichen (oder beim Ende)
*immer **genau** eine Regel, die man anwenden kann!*

Typ 3+: Deterministische reguläre (rechtslineare) Grammatiken

Bei einer deterministischen rechtslinearen Grammatik gibt es immer genau eine Regel (ohne ε), die man anwenden kann:

G = ({S,A,B,C}, {a,b}, P, S) mit P = { S → aA | bB,
A → aA | bC | ε,
B → aC | bB | ε,
C → aA | bC}

Ableitung z.B. S→aA→abC→abbC→abbaA→abba

Frage: Kann man das Wort „bbabbaaa“ ableiten?

Deterministische reguläre Grammatiken sind sehr einfach, weil

1. man das Wort immer von links nach rechts abarbeitet,
2. dabei immer genau einen Buchstaben weitergeht,
3. es nach jedem Schritt immer ganz rechts genau eine Variable gibt,
4. es dabei immer genau eine Regel gibt, die man anwenden kann,
5. man zum Schluss einfach nur schauen muss, ob es eine ε-Regel gibt.

Zusammenfassung

- Durch die **Chomsky-Hierarchie** werden Grammatiken bzw. Sprachen in vier ineinander geschachtelten Typen klassifiziert.
- Bei den **kontextfreien Sprachen** wird immer eine Variable ersetzt.
- Ein **Ableitungsbaum** repräsentiert Ableitungen von kontextfreien Grammatiken.
- Compiler benötigt einen Ableitungsbaum:
  - Die Rekonstruktion ist nur möglich für **eindeutige Grammatiken**.
  - **Mehrdeutige Grammatiken** können evtl. eindeutig gemacht werden.
- Die **regulären Grammatiken** haben entweder nur rechtslineare oder nur linkslineare Regeln. Bei den Ableitungen existiert immer nur eine Variable und das Wort bildet sich dabei Buchstabe um Buchstabe zur anderen Seite.
- Bei den **deterministischen regulären Grammatiken** sind die Ableitungen sehr einfach durchzuführen.
- Wir werden im nächsten Kapitel direkt die Typ-3 Sprachen, d.h. die rechtslinearen Sprachen und deren Automaten behandeln.