7 Die Chomsky-Hierarchie
Formale Sprachen, Grammatiken, Automaten

Σ Alphabet, $L \subseteq \Sigma^*$ formale Sprachen.

- Terme über Signatur (S, Sigma)
- Formeln
- While Programme
- Partielle Korrektheitsformeln
- Ausdrücke (primitiv rekursiv, μ-rekursiv)

Wie beschreibt man Sprachen ?

- Durch **Grammatiken** $G = (N, T, \Pi, Z)$ (spezielle Kalküle)
  N, T disjunkte Alphabete, Π Produktionen über $N \cup T$
  $Z \in N$ Startsymbol. Von G **erzeugte** Sprache:
  $L(G) = \{w \in T^* : Z \underset{\Pi}{\vdash} w\}$, d. h.
  $Z \overset{1}{\underset{\Pi}{\vdash}} w_1 \overset{1}{\underset{\Pi}{\vdash}} \cdots \overset{1}{\underset{\Pi}{\vdash}} w_n = w \qquad n \geq 1$

Problem: Wie entscheidet man $w \in L(G)$?

- Durch **Automaten** $A = (Q, N, T, \Pi, i, F)$
  Q endliche Zustandsmenge, Π Produktionen über $N \cup T$, die Übergang zwischen Konfigurationen beschreiben, i Initialkontext, F Finalkonfigurationen. Von A **akzeptierte** Sprache:
  $L(A) = \{w \in T^* : \exists f \in F \;\; i(w) \underset{\Pi}{\vdash} f\}$

Problem: Wie entscheidet man $w \in L(A)$?

7.1 Grammatiken

7.1 Definition Allgemeine Grammatiken

Eine Grammatik ist ein 4 Tupel

$$G = (N, T, \Pi, Z)$$

- Mit N endliche Menge **Nichtterminalsymbole**,
- T endliche Menge **Terminalsymbole**, $N \cap T = \varnothing$,
- Π endliche Menge von **Produktionen** $l \longrightarrow r$ mit $l, r \in (N \cup T)^*$, wobei l mindestens ein Zeichen aus N enthält und $Z \in N$ Startsymbol ist.

Die von G **erzeugte Sprache** ist die Menge

$$L(G) = \{w \in T^* : Z \underset{\Pi}{\vdash} w\}$$

D.h. es gibt eine Ableitung $\{Z, w_1, \ldots, w_n = w\}$ für w mit $Z \overset{1}{\underset{\Pi}{\vdash}} w_1 \overset{1}{\underset{\Pi}{\vdash}} w_2 \overset{1}{\underset{\Pi}{\vdash}} \cdots \overset{1}{\underset{\Pi}{\vdash}} w$, d. h. $Z \overset{n}{\underset{\Pi}{\vdash}} w$ im Wortersetzungssystem $(N \cup T, \Pi)$, für ein $n \geq 1$.

Zwei G_1, G_2 Grammatiken sind **äquivalent**, falls $L(G_1) = L(G_2)$.

Beispiele

7.2 Beispiel Schreibweisen

a) $G = (N, T, \Pi, Z)$, $N = \{Z, Z_1\}$, $T = \{a, b\}$
$\Pi :: Z \to aZ_1, Z_1 \to bZ_1 \mid a$ 3 Produktionen.
Behauptung: $L(G) = \{ab^n a : n \in \mathbb{N}\}$
Beweis: „$\supseteq$“ Gebe Ableitung an.
„$\subseteq$“ $L(Z_1, G) = \{w \in T^* : Z_1 \underset{\Pi}{\vdash} w\} = \{b^n a : n \in \mathbb{N}\}$
Induktion nach $i : Z_1 \overset{i}{\underset{\Pi}{\vdash}} w$, $w \in T^*$
$i = 1 \rightsquigarrow w = a$
$i \to i + 1 \quad Z_1 \overset{i}{\vdash} b^i Z_1 \vdash b^i a$

b) $G = (N, T, \Pi, Z)$, $N = \{Z\}$, $T = \{a, b\}$
$\Pi :: Z \to aZb \mid \varepsilon$
Behauptung: $L(G) = \{a^n b^n : n \in \mathbb{N}\}$
Sei $\alpha \in V^* = (N \cup T)^*$, $\alpha \notin T^*$, $Z \overset{n}{\underset{\Pi}{\vdash}} \alpha$, so $\alpha = a^n Z b^n$.
Induktion nach n.
Dann „$\subseteq$“ klar, „$\supseteq$“ Angabe einer Ableitung.

c) $N = \{Z, T, S, A, B\}$, $T = \{a, b\}$
$\Pi :: Z \to TS, T \to aTA \mid bTB \mid \varepsilon, S \to \varepsilon$
$Aa \to aA, \quad Ab \to bA, \quad AS \to aS$
$Ba \to aB, \quad Bb \to bB, \quad BS \to bS$

Beispiele (Fort.)

Beispiel einer Ableitung:
$Z \overset{1}{\vdash} TS \overset{1}{\vdash} aTAS \overset{1}{\vdash} abTBAS \overset{1}{\vdash} abBAS \overset{1}{\vdash} abBaS \overset{1}{\vdash} abaBS \overset{1}{\vdash} ababS \overset{1}{\vdash} abab$
Behauptung: $L(G) = \{ww : w \in T^*\}$
Für $w = w(a, b)$, sei $\hat{w} = w(A, B)$ das entsprechende Wort in den Großbuchstaben. Weiterhin sei ρ die Spiegelungsfunktion.
„$\supseteq$“ $Z \underset{\Pi}{\vdash} wT\rho(\hat{w})S \underset{T\to\varepsilon}{\vdash} w\rho(\hat{w})S \underset{\Pi}{\vdash} wwS \vdash ww$
„$\subseteq$“ Normierte Ableitungen: Erst T-Regeln bis $T \to \varepsilon$
$Z \vdash TS \underset{\Pi}{\vdash} wT\rho(\hat{w})S \vdash w\rho(\hat{w})S \underset{\Pi}{\vdash} ww$
Groß $\to$ klein, Vertauschregeln, mit $AS \to aS$, $BS \to bS$

d) $N = \{Z, A, B\}$, $T = \{a, b\}$
$\Pi :: Z \to \varepsilon \mid aAbZ \mid bBaZ, \quad A \to \varepsilon \mid aAbA, \quad B \to \varepsilon \mid bBaB$
Behauptung: $L(G) = \{w \in T^* : |w|_a = |w|_b\}$
$Z \underset{\Pi}{\vdash} \alpha \in (N \cup T)^*$, $|w|_a = |w|_b$ klar aus Regeln, also $L(G) \subseteq \{w \in T^* \mid |w|_a = |w|_b\}$
„$\supseteq$“ Ableitung angeben + Induktion $|w|_a = |w|_b$.

Eine andere Möglichkeit: $\Pi' : Z \to \varepsilon \mid aZb \mid bZa \mid ZZ$, dann $L(G') = L(G)$. Also sind G und G' äquivalent.
Frage: Einfachste Grammatik, die eine Sprache L erzeugt?

Beispiele (Forts.)

e) $N = \{Z, B, C\}$, $T = \{a, b, c\}$

$\Pi :: Z \to aZBC \mid aBC, \quad CB \to BC,$
$aB \to ab, bB \to bb,$
$bC \to bc, cC \to cc$

Behauptung: $L(G) = \{a^n b^n c^n : n \geq 1\}$

„$\supseteq$“ $Z \overset{n-1}{\vdash} a^{n-1}S(BC)^{n-1} \overset{1}{\underset{S \to aBC}{\vdash}} a^n(BC)^n \underset{\Pi}{\vdash}$
$a^n B^n C^n \underset{\Pi}{\vdash} a^n b^n C^n \underset{\Pi}{\vdash} a^n b^n c^n$

„$\subseteq$“ Jede Ableitung lässt sich „normieren“, erst alle Anwendungen von Z-Regeln (d. h. keine $CB \to BC$ Anwendung), dann die restlichen Regeln.

$Z \underset{\Pi}{\vdash} a^n ZW(B, C) \overset{1}{\vdash} a^{n+1}BCW(B, C) \vdash$
$a^{n+1}b^{n+1}c^{n+1}$
mit $|W(B, C)|_B = |W(B, C)|_C = n$

Aus $aW(B, C)$ mit $|W(B, C)|_B = |W(B, C)|_C$ lässt sich nur $ab^n c^n$ ableiten (als terminales Wort).

7.2 Chomsky Hierarchie

7.3 Definition Klassifikation nach Form der Produktionen

Sei $G = (N, T, \Pi, Z)$ Grammatik.

0) G ist vom **Typ 0**, falls keine Einschränkungen.
1) G ist vom **Typ 1 (kontext-sensitiv)**, falls $l \longrightarrow r \in \Pi$, so $l = xAy$, $r = xzy$ mit $x, y \in (N \cup T)^*$, mit $A \in N$, $z \in (N \cup T)^+$ (d. h. $|l| \leq |r|$).
   Ausnahme: $Z \longrightarrow \varepsilon$ (ε - Regel) erlaubt, falls Z in keiner rechten Seite einer Produktion vorkommt.
2) G ist vom **Typ 2 (kontext-frei)**, falls $l \longrightarrow r \in \Pi$, so $l = A$, $r = z$ mit $A \in N, z \in (N \cup T)^*$.
3) G ist vom **Typ 3 (rechts-linear)**, falls $l \longrightarrow r \in \Pi$, so $l = A, r = aB|a|\varepsilon$, $A, B \in N, a \in T$.

Eine Sprache $L \subseteq T^*$ heißt **vom Typ i**, falls es eine Grammatik G vom Typ i gibt mit $L = L(G)$.

Im **Beispiel** 7.2: a) Typ 3, b) Typ 2, c) Typ 0, d) Typ 2, e) Typ 0.

Beachte: G rechts-linear, so G kontext-frei, G kontext-frei ohne ε-Regeln, so G kontext-sensitiv.

Normierungen für Grammatiken

7.4 Bemerkung Normierte Grammatiken - Eigenschaften

- Es gibt stets eine äquivalente Grammatik vom gleichen Typ, für die das Startsymbol in keiner rechten Seite einer Produktion vorkommt.
  $\Pi_1 = \Pi \cup \{Z_1 \to Z\}$
  Für Typ 3 $\{Z_1 \to \alpha$: für $Z \to \alpha \in \Pi\}$
- Für eine kontext-freie Grammatik G und Wörter $x, y, z, u, v \in (N \cup T)^*$ gilt
  $x \underset{\Pi}{\vdash} y$ so $uxv \underset{\Pi}{\vdash} uyv$ (gilt sogar für beliebige G)
  $xy \overset{n}{\underset{\Pi}{\vdash}} z$, so gibt es $z_1, z_2 \in (N \cup T)^*$ mit $z = z_1 z_2$ und
  $x \overset{\leq n}{\underset{\Pi}{\vdash}} z_1, \quad y \overset{\leq n}{\underset{\Pi}{\vdash}} z_2$ (Ind. nach n).
- Für jede kontext-freie Grammatik G gibt es eine ε-freie kontext-freie Grammatik G_1 mit $L(G_1) = L(G) - \{\varepsilon\}$.
  Ist $\varepsilon \in L(G)$, dann gibt es eine kontext-freie Grammatik G' mit $L(G') = L(G)$, wobei die einzige Regel in G', die ε als rechte Seite hat, $Z' \to \varepsilon$ ist. Hierbei ist Z' Startsymbol von G', und Z' kommt in keiner rechten Seite einer Regel vor.

Normierungen - Abschlusseigenschaften

Beweisidee:

Sei $U_1 = \{X : X \to \varepsilon \in \Pi\}$ und
$U_{i+1} = U_i \cup \{X : X \to \alpha \in \Pi, \alpha \in U_i^*\}$.

Offenbar $U_i \subseteq N$, $U_i \subseteq U_{i+1}$. D. h. es gibt k mit $U_k = U_{k+1}$ und somit $U_k = U_{k+v}$, für $v = 0, 1, 2, 3 \ldots$

Behauptung: $X \underset{\Pi}{\vdash} \varepsilon$ gdw $X \in U_k$. (Beweis: Übung).
Insbesondere: $\varepsilon \in L(G)$ gdw $Z \in U_k$.

Definiere: $G_1 = (N, T, \Pi_1, Z)$ mit
$X \to \alpha' \in \Pi_1$ gdw es gibt $X \to \alpha \in \Pi, \alpha' \neq \varepsilon$ entsteht durch Streichen von Buchstaben in U_k (kein Streichen erlaubt).

7.5 Lemma Abschlusseigenschaften von $\mathcal{L}_i$

$\mathcal{L}_i$ ist abgeschlossen bzgl. $\cup, \circ, *$ für $i = 0, 1, 2, 3$.

Beweis:

$L_1 \circ L_2 = \{uv : u \in L_1, v \in L_2\}$

$L^* = \{u_1 \ldots u_n : n \in \mathbb{N}, u_i \in L\} = \bigcup_{n \geq 0} L^n \quad (L^0 = \{\varepsilon\})$

Sei L_j erzeugt von $G_j = (N_j, T_j, \Pi_j, Z_j)$. G_j vom Typ i $(i = 0, 1, 2, 3)$, $j = 1, 2$.

Abschlusseigenschaften

O.B.d.A. auf linken Seiten von Produktionen kommen keine terminalen Buchstaben vor. (Für $a \in T$ Platzhalter $A_a \in N$, ersetze Vorkommen von a in linker Seite durch A_a. Hinzunahme von Produktionen $A_a \to a$). $N_1 \cap N_2 = \varnothing$.

a) $\cup$: $G =$
$(N_1 \cup N_2 \cup \{Z\}, T_1 \cup T_2, \Pi_1 \cup \Pi_2 \cup \{Z \to Z_1 \mid Z_2\})$
Für Typ (3): $Z \to \alpha$ für $Z_1 \to \alpha \in \Pi_1$ oder $Z_2 \to \alpha \in \Pi_2$.
G ist vom Typ i und $L(G) = L(G_1) \cup L(G_2)$.

b) $\circ$: $G = (N_1 \cup N_2 \cup \{Z\}, T_1 \cup T_2, \Pi_1 \cup \Pi_2 \cup \{Z \to Z_1 Z_2\})$
G ist vom Typ i für $i = 0, 1, 2$.
Behauptung: $L(G) = L(G_1) \circ L(G_2)$.
„$\supseteq$“ $Z \underset{\Pi}{\overset{1}{\vdash}} Z_1 Z_2 \underset{\Pi}{\vdash} u Z_2 \vdash uv$ für $u \in L(G_1), v \in L(G_2)$.
„$\subseteq$“ $Z \underset{\Pi}{\overset{1}{\vdash}} Z_1 Z_2 \underset{\Pi}{\vdash} X$ und $X \in (T_1 \cup T_2)^*$.
Dann $Z_1 \underset{\Pi_1}{\vdash} X_1$ und $Z_2 \underset{\Pi_2}{\vdash} X_2$, $X = X_1 X_2$.
Da linke Seiten nur aus nichtterminalen Buchstaben und
$N_1 \cap N_2 = \varnothing$, d. h. keine Vermischungen.
Für Typ 3 - Grammatiken:
Π_1' entstehe aus Π_1 durch Ersetzen von jeder Produktion $X \to a|\varepsilon$ durch $X \to a Z_2$ bzw. $X \to \alpha$ für $Z_2 \to \alpha \in \Pi_2$.
$G = (N_1 \cup N_2, T_1 \cup T_2, \Pi' \cup \Pi_2, Z_1)$ erfüllt Forderung.

Abschlusseigenschaften (Fort.)

c) $* : L^* = \{w : \exists n \in \mathbb{N}, w \in L^n, w = v_1 \dots v_n, v_i \in L\}$
Sei $G =$
$(N_1 \cup \{Z\}, T_1, \Pi_1 \cup \{Z \to \varepsilon, Z \to Z_1, Z_1 \to Z_1 Z_1\})$.
Dann ist G vom Typ i für $i = 0, 1, 2$ und $L(G) = L(G_1)^*$.
Für Typ 3 Grammatiken: Übung.

7.6 Folgerung

- Jede endliche Sprache ist vom Typ 3:
  $w = a_1 \dots a_n \qquad a_i \in T \qquad n \geq 0$
  $Z \to a_1 A_1, A_1 \to a_2 A_2, \dots, A_{n-1} \to a_n A_n, A_n \to \varepsilon$
  $N = \{Z, A_1, \dots, A_n\}$
- $\mathcal{L}_{\text{endl}} \subsetneq \mathcal{L}_{T_3} \subsetneq \mathcal{L}_{T_2} \subsetneq \mathcal{L}_{T_1} \subsetneq \mathcal{L}_{T_0}$
- Wie ordnen sich die Sprachklassen in Hierarchie ein?

$$\mathcal{L}_{\text{endl}} \subsetneq \mathcal{L}_{\text{prim-rek}} \subsetneq \mathcal{L}_{\text{rek-entsch.}} \subsetneq \mathcal{L}_{\text{rek-aufzb.}}$$

Ist $L(G)$ entscheidbar für beliebiges G ?

7.7 Lemma

Sei $G = (N, T, \Pi, Z)$ Grammatik, dann ist $L(G)$ rekursiv aufzählbar.

Idee: Führe systematisch alle Ableitungen aus Z der Länge nach durch.

Ableitbare Wörter aus $(N \cup T)^*$ in $1, 2, 3 \dots$ Ableitungsschritte.

$L(G)$ ist rekursiv aufzählbar

- Verfahren hält mit Eingabe w gdw $s \overset{i}{\underset{\Pi}{\vdash}} w$ für ein i d. h. w kommt in Stufe i vor.
- Verfahren ist effektiv und hält bei Eingabe w gdw $w \in L(G)$.

Formal: Sei $\Sigma = N \dot\cup T \dot\cup \{\vdash\}, \Pi = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ und

$M = \{w \in \Sigma^*$: Es gibt $w_1, \ldots, w_m \in (N \cup T)^*$ mit
$w = \ \vdash Z \vdash w_1 \vdash \cdots \vdash w_m \vdash$ und
$Z \overset{1}{\underset{\Pi}{\vdash}} w_1, w_i \overset{1}{\underset{\Pi}{\vdash}} w_{i+1}$ für $i \geq 1\}$

M ist die Menge der Ableitungen in G.

Für $\alpha, \beta \in V^*$ sei

$Q_i(\alpha, \beta)$ gdw $\alpha \underset{l_i \to r_i}{\vdash} \beta$ gdw
$\exists\, \alpha', \alpha'' \leq \alpha.\ \alpha = \alpha' l_i \alpha'' \wedge \beta = \alpha' r_i \alpha''$

$Q(\alpha, \beta)$ gdw $\alpha \overset{}{\underset{\Pi}{\vdash}} \beta$ gdw $Q_1(\alpha, \beta) \vee \cdots \vee Q_n(\alpha, \beta)$.

Offenbar $Q_1, \ldots, Q_n \in \mathcal{P}(\Sigma), Q \in \mathcal{P}(\Sigma)$.

M ist primitiv rekursiv (verwende Anfangswort, Teil- und Endwort).

$\bullet x \in L(G)$ gdw $\exists w.\ w \in M \wedge$ Endwort$(\vdash x \vdash, w)$.

Umkehrung

7.8 Lemma

$L \subset \Sigma^*$ rekursiv aufzählbar, dann gibt es eine Grammatik $G = (N, \Sigma, \Pi, Z)$ mit $L = L(G)$.

Beweisidee: Simuliere mit der Grammatik die TM-Schritte einer TM die L akzeptiert rückwärts.

Sei o.B.d.A. T eine TM, die L akzeptiert mit nur einem Haltezustand q. D. h. $F = \{q\}$. $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$

Die Konfigurationen von T werden in Klammern eingeschlossen: $[uq_iv]$.

Produktionen von G bewirken:

1-Gruppe: $Z \underset{G}{\vdash} [uqv] \quad u, v \in \Gamma^* \;\; (u, v \text{ lang genug})$.

2-Gruppe: $[k_{i+1}] \underset{G}{\vdash} [k_i]$, falls $k_i \underset{T}{\vdash} k_{i+1}$, dabei ist $|k_i| = |k_{i+1}|$.

Dann gilt: $[uqv] \underset{G}{\vdash} [\square^s q_0 \square x \square^t]$, falls $q_0 \square x \overset{*}{\underset{T}{\vdash}} uqv \quad (x \in \Sigma^*)$.

3-Gruppe: $[\square^s q_0 \square x \square^t] \underset{G}{\vdash} x$ für alle $s, t \in \mathbb{N}, x \in \Sigma^*$.

Wählt man s, t genügend groß, so verlässt die TM bei ihrer Berechnung nie den Block $\square^s x \square^t$.

Produktionen (Forts.)

Produktionen 1-Gruppe:
$Z \to [Z_0], Z_0 \to Z_0 b \mid b Z_0 \mid q \ (b \in \Gamma)$.

Produktionen 2-Gruppe: z. B. aus Turing Programm

$$
\begin{array}{llll}
q_i : a & \rightsquigarrow & q_{i+1} a \to q_i b & b \in \Gamma \\
q_i : R & \rightsquigarrow & b q_{i+1} \to q_i b & b \in \Gamma \\
q_i : L & \rightsquigarrow & q_{i+1} b \to b q_i & b \in \Gamma \\
q_i : q_k & \rightsquigarrow & q_k \to q_i & \\
q_i : a, q_k & \rightsquigarrow & q_k a \to q_i a & \text{und } q_{i+1} b \to q_i b \ (b \neq a)
\end{array}
$$

Produktionen 3-Gruppe:

$q_0 \to T_1$, $\square T_1 \to T_1$, $[T_1 \square \to T_2$

$T_2 b \to b T_2$, $b \in \Sigma$, $T_2 \to T_3$,

$T_3 \square \to T_3$, $T_3] \to \varepsilon$.

G **ist Typ-0 Grammatik**!
Hierbei ist $N = \{Z, Z_0, T_1, T_2, T_3, [,]\} \cup Q \cup (\Gamma - \Sigma)$
Es gilt $Z \underset{G}{\vdash} x \in \Sigma^*$ gdw T akzeptiert x, d. h. $L(G) = L$.

7.9 Satz

$L \subseteq \Sigma^*$ ist rekursiv aufzählbar gdw es gibt eine Typ-0 Grammatik $G = (N, \Sigma, \Pi, Z)$ mit $L = L(G)$.

Insbesondere sind Typ-0-Sprachen abgeschlossen gegenüber $\cap$ aber nicht gegen $\neg$ (Komplement) und es gibt nicht entscheidbare Typ-0-Sprachen.

Wortprobleme

7.10 Definition Wortproblem, uniformes Wortproblem

Sei $G = (N, \Sigma, \Pi, Z)$. Das **Wortproblem** für G ist definiert:

$$WP(x) \text{ gdw } x \in L(G) \quad (x \in \Sigma^*)$$

Ist $\mathcal{G}$ eine Klasse von Grammatiken, so ist das **uniforme Wortproblem** für $\mathcal{G}$ definiert durch

$$UWP(G, x) \text{ gdw } x \in L(G) \quad (G \in \mathcal{G}, x \in T_G^*)$$

7.11 Folgerung

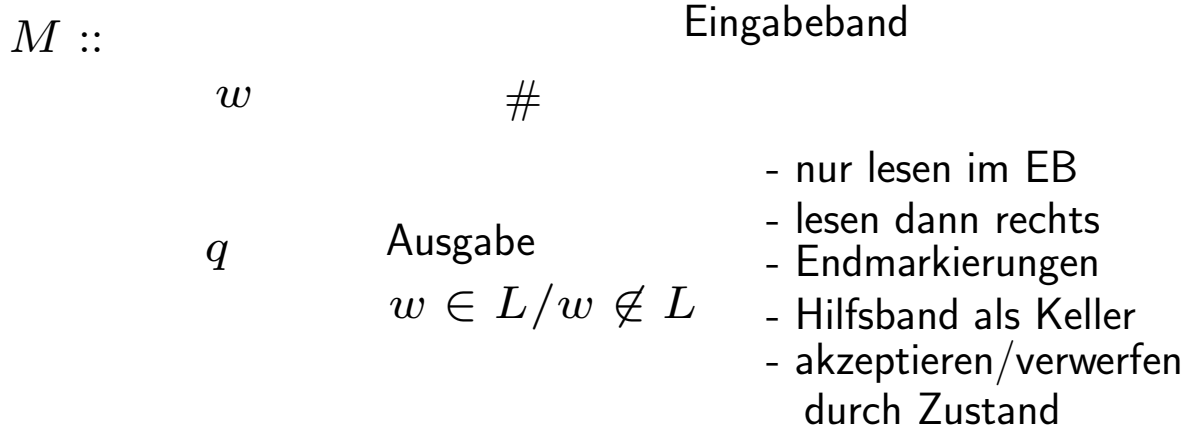
- UWP ist nicht entscheidbar für Typ-0 Grammatik.
- Es gibt Typ-0 Grammatik mit unentscheidbaren WP.
- Das uniforme WP für Typ 1 Grammatiken ist primitiv rekursiv.

$$\mathcal{L}_{\text{endl.}} \subsetneq \mathcal{L}_{\text{Typ}-3} \subseteq \mathcal{L}_{\text{Typ}-2} \subseteq \mathcal{L}_{\text{Typ}-1} \subseteq \mathcal{L}_{\text{prim-rek}} \subsetneq \mathcal{L}_{\text{Typ}-0} = \mathcal{L}_{\text{rek-aufzb.}}$$

endliche Automaten EA	Keller-automaten PDA	linear beschränkte Automaten LBA	TM als akzeptierende Automaten

Formale Sprachen und akzeptierende Automaten

Einschränkungen der Turing-Maschinen: Möglichkeiten



Konfigurationen: $uqw \underset{M}{\vdash} u'q'w'$ mit Hilfe von Produktionen.

7.12 Definition Automaten für Sprachen

Ein **Automat** (oder **Akzeptor**) $A = (Q, N, T, \Pi, i, F)$ mit endlicher **Zustandsmenge** Q, endlicher Menge N von **Hilfssymbolen** und endlichem **Eingabealphabet** T, so dass Q, N, T paarweise disjunkt sind, $i : T^* \to (N \cup T)^* \cdot Q \cdot (N \cup T)^*$: **Initialkonfiguration** zur Eingabe $w \in I^*$, einer endlichen Menge von **Finalkonfigurationen** F der Form $lqr \in (N \cup T)^* q (N \cup T)^*$ und einer endlichen Menge Π von **Produktionen** der Form $lqr \to l'q'r'$ $(l, l', r, r' \in (N \cup T)^* q, q' \in Q)$.

$L(A) = \{w \in T^* : \exists f \in F \quad i(w) \underset{\Pi}{\vdash} f\}$ die von A **akzeptierte Sprache**.

7.3 Endliche Automaten - reguläre Sprachen - Typ 3-Sprachen

Typ-3 Grammatik: $G = (N, T, \Pi, \Sigma)$, Π mit Produktionen der Form $A \longrightarrow aB|a|\varepsilon$, $A, B \in N, a \in T$

7.13 Definition Endliche Automaten

a) Ein (deterministischer) **endlicher Automat (DEA)** ist ein 5-Tupel $A = (Q, \Sigma, \Pi, q_0, F)$ mit $q_0 \in Q$ **Startzustand**, $F \subset Q$ Menge der **Finalzustände** (akzeptierende Zustände).
$\Pi = \{qa \longrightarrow q' : q, q' \in Q, a \in \Sigma\}$: Für jedes Paar $(q, a) \in Q \times \Sigma$ gibt es genau eine Produktion $qa \longrightarrow q'$.

b) Ein **indeterministischer endlicher Automat (NEA)** ist ebenfalls ein 5-Tupel A wie eben mit dem Unterschied, dass es für jedes Paar $(q, a) \in Q \times \Sigma$ eine endliche (eventuell leere) Menge von Produktionen der Form $qa \longrightarrow q'$ sowie Produktionen der Form $q \longrightarrow q'$ (Spontanübergänge, ε-**Übergänge**) gibt.

c) **Initialkonfiguration** bei Eingabe $w \in \Sigma^* : q_0 w$,
d. h. $i(w) = q_0 w$ für $w \in \Sigma^*$.
Finalkonfigurationen: F.

d) Die von A **akzeptierte Sprache** ist die Menge
$L(A) = \{w \in \Sigma^* : q_0 w \underset{\Pi}{\vdash} f \text{ für ein } f \in F\}$.
Schreibe auch $q_0 w \underset{A}{\vdash} f$.

Beispiele - Darstellungsarten
Zustandsgraph oder Automatendiagramme

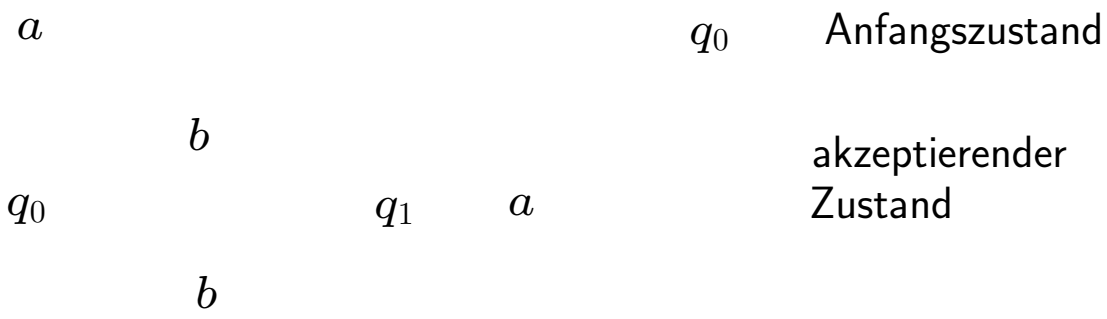
7.14 Beispiel

1. $A = (\{q_0, q_1\}, \{a, b\}, \Pi, q_0, \{q_0\})$
   $\Pi :: q_0a \to q_0, q_0b \to q_1, q_1a \to q_1, q_1b \to q_0$

   Behauptung: $q_0w \underset{A}{\vdash} q_0$ gdw $|w|_b$ gerade.

   Beweis: Induktion nach $|w|_b$,
   d. h. $L(A) = \{w \in \{a, b\}^* : |w|_b$ gerade$\}$.

   Diagramm: Knoten $\leftrightarrow$ Zustand, gerichtete Kante $\leftrightarrow$ Produktion



   Matrix-Tabelle:

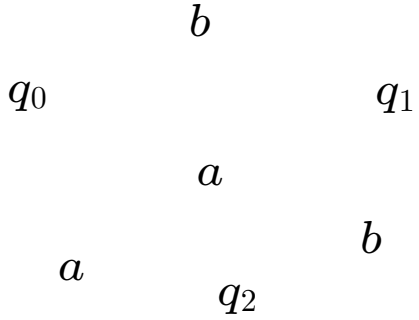
	a	b
q_0	q_0	q_1
q_1	q_1	q_0

Beispiele - Darstellungsarten Zustandsgraph oder Automatendiagramme (Forts.)

Bei indeterminierten Automaten: mehrere Kanten aus Zustand können mit Buchstaben a oder ε markiert sein.

Tabellendarstellung: Zustandsmengen + ε-Spalte.

2. Betrachte



Behauptung: $L(A) = \{ab, aba\}^*$

„$\supseteq$“ klar. „$\subseteq$“ Es gelte: $q_0 w \underset{A}{\vdash} q_0$.

Dann $w = \varepsilon$ oder w fängt mit a an.

$q_0 a w' \vdash q_1 w' \vdash q_0 \rightsquigarrow w'$ fängt mit b an.

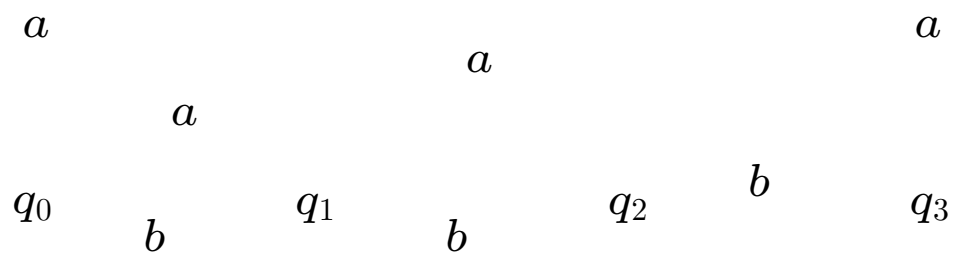
$q_0 w''$ Induktion

$q_1 b w''$

$q_2 w''$ w'' mit a +Ind.

Beispiele (Fort.)

3. $L = \{w \in \{a,b\}^* : w$ enthält nicht bbb als TW$\}$.

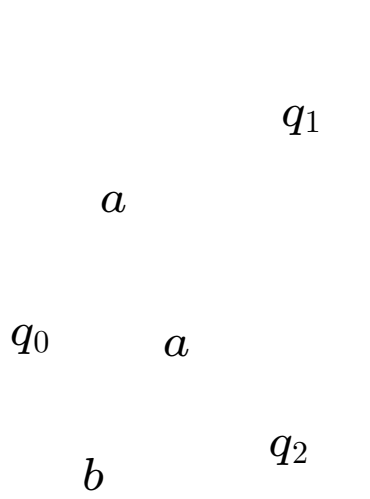


Beschreibung der Wege, die von q_0 nach q_i führen.

$q_0 \rightsquigarrow q_0 : \varepsilon, \{a\}^*, \{a\}^*\{ba\}^*\{a\}^*, a^*bbaa^*, \ldots$

Reguläre Ausdrücke zur Beschreibung von Sprachen.

4. Betrachte



$$F = \{q_0\} \rightsquigarrow L(A) = \{ba\}^*$$
$$F = \{q_2\} \rightsquigarrow L(A) = \{ba\}^*b$$
$$F = \{q_1\} \rightsquigarrow L(A) = (ba)^*(a + bb)\Sigma^*$$
$$F = \{q_0, q_2\} = (ba)^* + (ba)^*b$$
$$= (ba)^*(\varepsilon + b) \quad \text{Vereinigung}$$

Operationen: Verkettung, Vereinigung, Iteration $(*)$.

Beispiele (Fort.)

5. Dezimalzahlen, die durch 5 teilbar sind.

	0	1	2	3	4	5	6	7	8	9
q_0	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_1	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_2	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_3	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_4	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4

$q_0 w \vdash q_i$ gdw $w \equiv i (\mathrm{mod}\ 5)$, $F = \{q_0\}$
Automat mit 2 Zustände genügt!
$\rightsquigarrow$ **Äquivalente Automaten, minimale Automaten**.

Endliche Automaten und Typ-3-Grammatiken

7.15 Lemma Charakterisierungssatz

Ist $A = (Q, \Sigma, \Pi, q_0, F)$ EA, so ist $L(A)$ eine Typ-3 (rechtslineare) Sprache.

Beweis:

Definiere rl-Grammatik $G = (N, \Sigma, \Pi_G, Z)$ mit $N = Q$, $Z = q_0$, so dass für alle $x \in \Sigma^*$ gilt:

$$(*) \qquad q_0 x \underset{A}{\vdash} q_i \text{ gdw } Z \underset{G}{\vdash} x q_i$$

Endliche Automaten und Typ-3-Grammatiken

Definiere:

$$\begin{aligned}\Pi_G = \quad & \{q_i \to aq_j : q_i a \to q_j \in \Pi\} \\ & \cup \{q_i \to a : q_i a \to q \in \Pi \wedge q \in F\} \\ & \cup \{Z \to \varepsilon : \text{ falls } q_0 \in F\}\end{aligned}$$

G ist rechts-lineare Grammatik.

Behauptung: $(*)$ gilt für G:
Beweis: Induktion nach $|x|$.

„$\Rightarrow$“ $x = \varepsilon$, $q_0\varepsilon \underset{A}{\vdash} q_0$, $Z = q_0 \underset{G}{\vdash} q_0$

$x \rightsquigarrow xa$, $q_0 x \underset{A}{\vdash} q_i$, Ind. Vor $Z \underset{G}{\vdash} xq_i$

Sei $q_i a \to q_j \in \Pi$, dann $q_0 xa \underset{A}{\vdash} q_i a \underset{A}{\vdash} q_j$

Da $q_i \to aq_j \in \Pi_G$ folgt $Z \underset{G}{\vdash} xq_i \overset{1}{\vdash} xaq_j$

„$\Leftarrow$“ $x = \varepsilon$, $Z \underset{G}{\vdash} q_i$, dann $q_i = q_0$

$x \rightsquigarrow xa$, $Z \underset{G}{\vdash} xaq_j$. Da Π_G rechts-linear ist, folgt

$Z \underset{G}{\vdash} xq_i \overset{1}{\vdash} xaq_j$ mit Regel $q_i \to aq_j \in \Pi_G$.

Dann aber $q_i a \to q_j \in \Pi$.

Nach Ind. Vor: $q_0 x \underset{A}{\vdash} q_i$ und somit $q_0 xa \underset{A}{\vdash} q_i a \overset{1}{\underset{\Pi}{\vdash}} q_j$.

Endliche Automaten und Typ-3-Grammatiken (2)

Behauptung: $L(A) = L(G)$

„$\subseteq$“ $x \in L(A)$

-: $x = \varepsilon$, so ist $q_0 \in F$, $Z \to \varepsilon \in \Pi_G$, d. h. $x \in L(G)$

-: $x = ya$, $q_0 y \underset{A}{\vdash} q_i$, $q_i a \to q$ mit $q \in F$.

Dann folgt aus $(*)$ $Z \underset{G}{\vdash} yq_i \overset{1}{\vdash} ya$, da $q_i \to a \in \Pi_G$,

d. h. $ya \in L(G)$. Also $x \in L(G)$

„$\supseteq$“ $x \in L(G)$

-: $x = \varepsilon$, so $Z \to \varepsilon \in \Pi_G \rightsquigarrow q_0 \in F \rightsquigarrow x \in L(A)$

-: $x = ya$, $Z \underset{G}{\vdash} yq_i \overset{1}{\underset{G}{\vdash}} ya$. Wegen $(*)$ ist $q_0 y \underset{A}{\vdash} q_i$ und $q_i a \to q$ mit $q \in F$, d. h. $q_0 ya \underset{A}{\vdash} q_i a \underset{A}{\vdash} q \in F$.

Also $x \in L(A)$.

Beachte:

G ist rechts-linear und „eindeutig“, d. h. ist $w \in L(G)$, so gibt es genau eine Ableitung für w.

Falls A NEA, so Problem mit Spontanübergängen, diese würden Regeln der Form $q_i \to q_j$ bedeuten. Sonst ok.

Beispielkonstruktion

7.16 Beispiel Sei $A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \Pi, q_0, \{q_0\})$.

Π	a	b
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$G_A = (N, \Sigma, \Pi_G, Z)$, $N = \{q_0, \ldots, q_3\}$, $Z = q_0$

$$\begin{aligned} \Pi_G: \quad & q_0 \to aq_2|bq_1|\varepsilon \quad (q_0 \in F) \\ & q_1 \to aq_3|bq_0|b \quad (q_0 \in F) \\ & q_2 \to aq_0|a|bq_3 \quad (q_0 \in F) \\ & q_3 \to aq_1|bq_2 \end{aligned}$$

Beachte: $|\Pi_G| \leq 2 \cdot |\Sigma| \cdot |Q| + 1$.

Frage: Wird jede Typ-3 Sprache von einem DEA akzeptiert?

Problem: Bei Typ-3 Grammatiken ist $A \to aB$ und $A \to aC$ erlaubt, d. h. Indeterminismus.

Endliche Automaten und Typ-3-Grammatiken (3)

7.17 Lemma Charakterisierungssatz

Zu jeder Typ-3 Sprache L gibt es NEA A mit $L = L(A)$.

Beweis: Sei G Typ-3 Grammatik $G = (N, T, \Pi_G, Z)$ mit $L = L(G)$.

Definiere:

$A = (Q, T, \Pi_A, q_0, F)$ mit $Q = N \dot\cup \{S\}$, $q_0 = Z$.

$\Pi_A : \{Xa \to Y : \text{ für } X \to aY \in \Pi_G\}$
$\quad \cup \{Xa \to S : \text{ für } X \to a \in \Pi_G\}$

$F = \{S\} \cup \{X \mid X \to \varepsilon \in \Pi_G\}$

Behauptung:

a) $q_0 w \underset{A}{\vdash} X$ gdw $Z \underset{G}{\vdash} wX$ für $X \in N, w \in T^*$.

b) $w \in L(A)$ gdw $w \in L(G)$ gdw $Z \underset{G}{\vdash} w$ für $w \in T^*$.

Beweis:

a) Induktion nach $|w|$ :: -: $w = \varepsilon$
„$\Rightarrow$“ $X = q_0 = Z$,
„$\Leftarrow$“ dito.
-: $w = va$
„$\Rightarrow$“ $q_0 va \underset{A}{\vdash} X$, $x \in N$: Dann $q_0 v \underset{A}{\vdash} Y$, $Y \in N$ und $ya \vdash X$.
D. h. nach Ind. Vor. $Z \underset{G}{\vdash} vY \overset{1}{\underset{G}{\vdash}} vaX$.

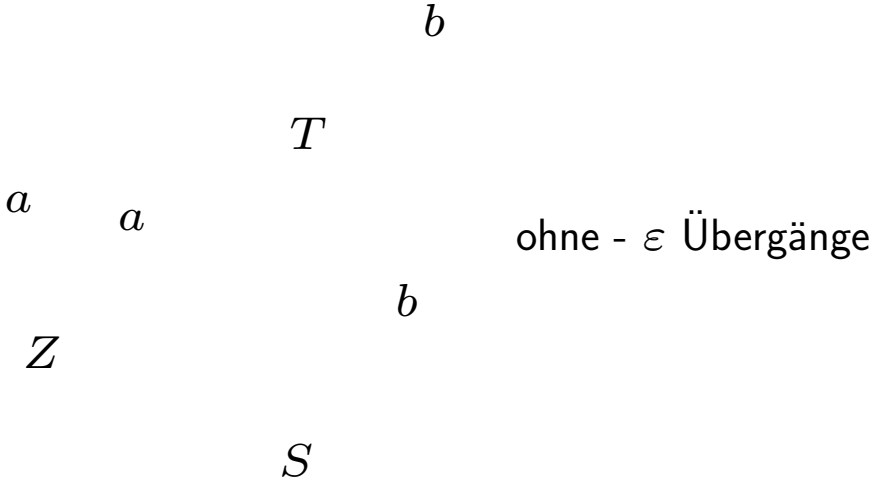
Konstruktion-Beispiele

„$\Leftarrow$“ $Z \underset{G}{\vdash} vaX$, $X \in N$. Dann $Z \underset{G}{\vdash} vY$, für ein $Y \in N$ und $Y \to aX \in \Pi_G$. Dann $q_0va \underset{A}{\vdash} Ya \underset{A}{\vdash} X$.

b) $w \in L(A)$.
Dann $q_0w \underset{A}{\vdash} S$ oder $q_0w \vdash X$ mit $X \to \varepsilon \in \Pi_G$. Dann aber $w = va$, $q_0w \underset{A}{\vdash} Xa \vdash S$.
$X \in N \rightsquigarrow Z \underset{G}{\vdash} vX \underset{G}{\vdash} va \in L(G)$ oder $Z \underset{G}{\vdash} wX \underset{G}{\vdash} w \in L(G)$. $\rightsquigarrow$ Behauptung.

7.18 Beispiel

1. $G = (N, \Sigma, \Pi_G, Z)$, $N = \{Z, T\}$, $\Sigma = \{a, b\}$
$\Pi_G :: Z \to aZ|aT$, $T \to bT|b$
Behauptung: $L(G) = \{a^n b^m : n, m \geq 1\}$ (klar).
Konstruktion:



Beispiele

2. Betrachte

a

a b a

b

b a b

$L(A) = \{ab, aba\}^*$

DEA

a, b

3. Sei

b

$L(A) = \{ab, aba\}^*$

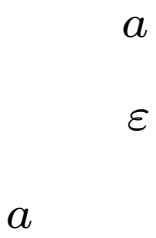
a

b NEA

a

Beispiele

4. Sei



$L(A) = \{ab, aba\}^*$
NEA
fast deterministisch

Kann man Spontanübergänge vermeiden?
JA: Idee $q \sim q'$ gdw es gibt $q_0, \ldots, q_n \in Q$
$q_0 = q, q_n = q', q_i \to q_{i+1} \in \Pi$. Lässt sich effektiv berechnen!

$$\Pi^* = \{qa \to q' : \exists q''(q \sim q'' \wedge q''a \to q' \in \Pi)\}$$

$$F^* = \{q : \exists f \in F : q \sim f\}$$

Dann $L(A) = L(A^*)$.

Wir haben somit:

7.19 Lemma

$L \subseteq T^*$ ist Typ-3 Sprache gdw $L = L(A)$ für ein NEA A.

Charakterisierungssatz für r.l. Sprachen

7.20 Satz

Zu jedem NEA A gibt es einen DEA A' mit $L(A) = L(A')$.

Beweis: Sei $A = (Q, \Sigma, \Pi, q_0, F)$ ein NEA. A enthalte keine ε-Übergänge. Definition DEA $A' = (Q', \Sigma, \Pi', q_0', F')$ mit

- $Q' =$ Potenzmenge von $Q = \{T : T \subseteq Q\}$
- $\Pi' = \{Ta \to \{q' \in Q : \exists q \in T \;\; qa \to q' \in \Pi\} : T \in Q', a \in \Sigma\}$
- $q_0' = \{q_0\}$
- $F' = \{T \subseteq Q : T \cap F \neq \varnothing\}$

Behauptung: $L(A') = L(A)$.

Beweis: Es gilt $Ty \underset{A'}{\vdash} \{q' \in Q : \exists q \in T \;\; qy \underset{A}{\vdash} q'\} =: T'$ für $T \subseteq Q, y \in \Sigma^*$.

Ind. nach $|y| : y = \varepsilon$, so $T' = T$, da keine Spontanübergänge.

Sei $y = az$, $a \in \Sigma$, dann

$$\begin{aligned} Taz &\underset{A'}{\vdash} \{q' : \exists q \in T \;\; qa \to q' \in \Pi\}z \\ &\underset{\text{Ind.Vor.}}{\underset{A'}{\vdash}} \{q'' : \exists q' \exists q \in T \;\; qa \to q' \in \Pi, q'z \underset{A}{\vdash} q''\} \\ &= \{q'' : \exists q \in T \;\; qaz \underset{A}{\vdash} q''\} \end{aligned}$$

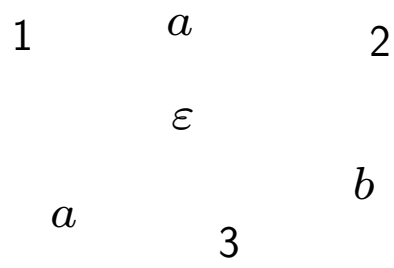
Beispiele

Sei

$$y \in L(A') \quad \text{gdw } \exists T \in Q\ (T \cap F \neq \varnothing \wedge \{q_0\} y \underset{A'}{\vdash} T)$$
$$\text{gdw } \{q \in Q : q_0 y \underset{A}{\vdash} q\} \cap F \neq \varnothing$$
$$\text{gdw } y \in L(A)$$

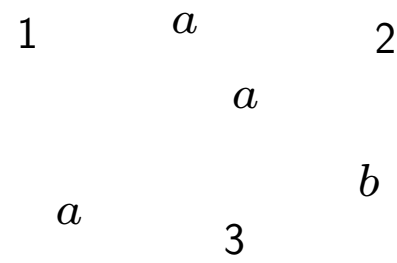
7.21 Beispiel

- Sei



hat Spontanübergänge
$3 \sim 1$

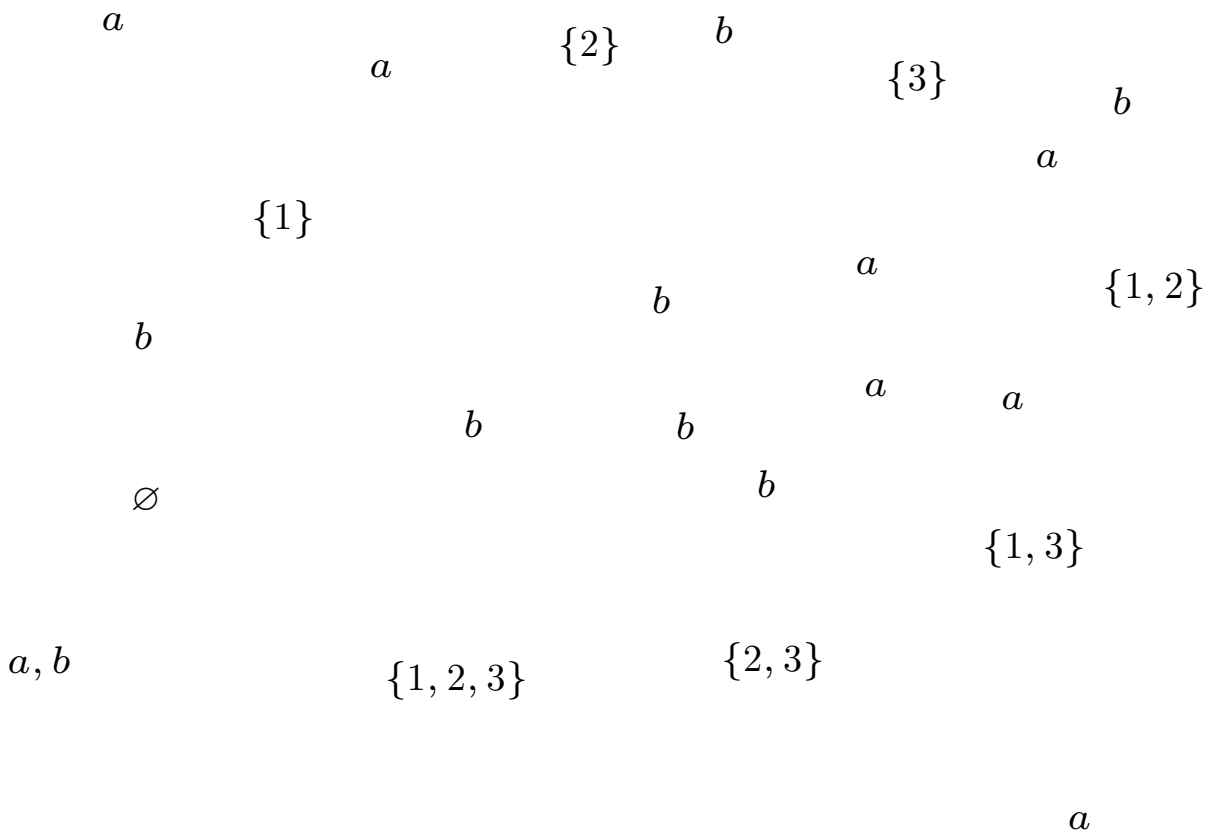
ohne ε-Übergänge



Neue Zustandsmenge:

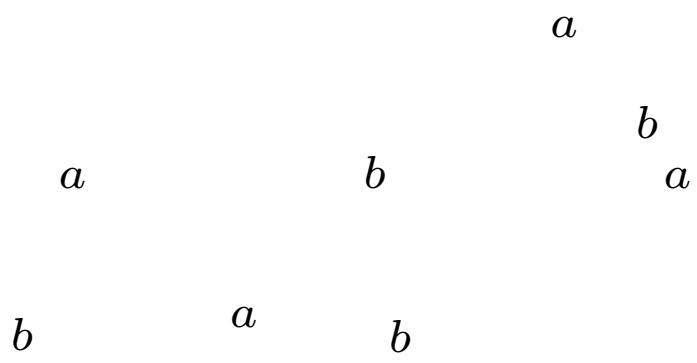
$\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}$

Beispiele (Fort.)



Konstruktion liefert oft zu viele Zustände. Nicht erreichbare Zustände (vom Startzustand aus) streichen.

Beispiele (Fort.)



a, b

Ist dies minimaler DEA der $L(A)$ akzeptiert, d. h. minimale Anzahl von Zuständen? JA.

$x \underset{A}{\sim} y$ gdw $(q_0 x \underset{A}{\vdash} q$ gdw $q_0 y \underset{A}{\vdash} q)$.

$\underset{A}{\sim}$ ist rechtsinvariant, d.h.

$x \underset{A}{\sim} y \rightarrow xz \sim yz$ für alle $z \in \Sigma^*$.

Index = Anzahl der Äquivalenzklassen.

$L(A)$ ist Vereinigung von Äquivalenzklassen (Myhill-Nerode).

Es gibt Verfahren um einen äquivalenten minimalen DEA zu bestimmen.

Folgerungen

7.22 Folgerung

a) Rechts-lineare Sprachen sind abgeschlossen gegenüber Komplement und Durchschnitt.
$A = (Q, \Sigma, \Pi, q_0, F)$ DEA $L = L(A)$.
$A' = (Q, \Sigma, \Pi, q_0, Q - F)$ DEA mit $L(A') = \neg L$.
$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$ oder direkt mit Produktautomaten.
$A_1 \times A_2 = (Q_1 \times Q_2, \Sigma, \Pi_1 \times \Pi_2, (q_{0_1}, q_{0_2}), F_1 \times F_2)$.

b) Jede Typ-3 Sprache kann von Typ-3 Grammatik G erzeugt werden mit: Π enthält für $X \in N, a \in \Sigma$ $X \to aY$ oder $X \to a$ (genau eine Produktion $X \to aY$). D. h. G ist eindeutig und somit ist jede Typ-3 Sprache eindeutig.

c) Das WP für Typ-3 Grammatiken ist in linearer Zeit entscheidbar.

d) **Pumping-Lemma** für Typ-3 Sprachen.
Zu jeder Typ-3 Sprache L gibt es ein $n \in \mathbb{N}$, so dass für alle $y \in L$ gilt: Ist $|y| \geq n$. Dann lässt sich y zerlegen in $y = uvw$ mit $0 < |v| \leq |uv| \leq n$, so dass für alle $i \in \mathbb{N}$ $uv^iw \in L$.
Beweis:
Sei A DEA mit $L(A) = L$ und $n := |Q|$. Ist $y \in L(A)$, $|y| \geq n$. Betrachte
$q_0 y \overset{1}{\vdash} q_1 y_1 \vdash \cdots \overset{1}{\vdash} q_{n-1} y_{n-1} \overset{1}{\vdash} q_n y_n \vdash \cdots \overset{1}{\vdash} q \in F$, $\{q_0, \ldots, q_n\} \subseteq Q$. Es gibt Zustand q', der zweimal vorkommt $q_0 uvw \underset{A}{\vdash} q'vw \underset{A}{\vdash} q'w \vdash q_0, v \neq \varepsilon, |uv| \leq n$. Dann aber $q_0 uv^i w \vdash q$ für alle $i \geq 0$.

Beispiel

7.23 Beispiel

$L = \{w \in \{a,b\}^* : |w|_a = |w|_b\}$ nicht Typ 3 Sprache.

Angenommen, L ist rechts-linear, sei n Konstante für L.
Betrachte $y = a^n b^n \in L$
Pumping-Lemma $\rightsquigarrow a^{k_0}(a^k)^i a^{k_1} b^n \in L$ für alle i $(k_0 + k + k_1 = n, k > 0)$ $\lightning$

Oder: $L \cap \{a\}^*\{b\}^* = \{a^n b^n \mid n \geq 0\}$ wäre rechts-linear, falls L es ist. $\lightning$

e) Für eine Typ-3 Sprache sind folgende Probleme entscheidbar. Dabei soll L durch eine Typ-3 Grammatik, oder durch einen DEA, oder durch einen NEA gegeben sein.
   - Ist L leer?
   - Ist $L = \Sigma^*$?
   - Ist L endlich?
   - Ist $L = L_1$ für eine Typ-3 Sprache L_1?

Es gibt weitere Charakterisierungen von rl-Sprachen, z.B. durch rechtsinvariante Äquivalenzrelationen auf Σ^* von endlichen Index (d.h. nur endlich viele Äquivalenzklassen) oder etwa durch reguläre Ausdrücke.

Andere Charakterisierung von Typ-3 Sprachen

Reguläre Ausdrücke über Σ : $REG(\Sigma)$

Wörter über $\Sigma \cup \{\Lambda, \varepsilon, \cup, *, (,)\}$ (oft $+$ für $\cup$).

Kalkül:

$\overline{\Lambda}$, $\overline{\varepsilon}$, $\overline{a}$ für $a \in \Sigma$, $\dfrac{\alpha, \beta}{(\alpha\beta)}$, $\dfrac{\alpha, \beta}{(\alpha \cup \beta)}$, $\dfrac{\alpha}{\alpha^*}$

Semantik: Reguläre Sprachen, die durch reg. Ausdrücke über Σ dargestellt werden: $\langle\ \rangle$: reg. Ausdruck $\longrightarrow$ Sprachen über Σ

- $\langle \Lambda \rangle = \varnothing$
- $\langle a \rangle = \{a\}$ $a \in \Sigma$
- $\langle (\alpha \cup \beta) \rangle = \langle \alpha \rangle \cup \langle \beta \rangle$
- $\langle \varepsilon \rangle = \{\varepsilon\}$
- $\langle (\alpha\beta) \rangle = \langle \alpha \rangle \circ \langle \beta \rangle$
- $\langle \alpha^* \rangle = \langle \alpha \rangle^*$

7.24 Satz

L ist Typ-3 Sprache gdw L ist reguläre Sprache, d. h.
es gibt $\alpha \in REG(\Sigma) : \langle \alpha \rangle = L$.

Beweis:

„$\Leftarrow$“ Typ-3 Sprachen enthalten $\varnothing$,$\{\varepsilon\}$, $\{a\}$ für $a \in \Sigma$ und sind abgeschlossen gegen $\cdot, \cup, *$.

„$\Rightarrow$“ Sei $A = (Q, \Sigma, \Pi, q_1, F)$, $Q = \{q_1, \ldots, q_n\}$ DEA mit $L(A) = L$. Für $i, j \in \{1, \ldots, n\}$ und $t \in \{0, \ldots, n\}$ definiere

$$L_{ij}^t = \{y \in \Sigma^* : q_i y \overset{1}{\vdash} q_{i_1} y_1 \overset{1}{\vdash} \cdots \overset{1}{\vdash} q_{i_k} y_k \overset{1}{\vdash} q_j$$
mit Zwischenzuständen
$q_{i_1}, \ldots, q_{i_k} \in \{q_1, \ldots, q_t\}\}$

Behauptung: Jedes L_{ij}^t ist durch regulären Ausdruck darstellbar. Insbesondere auch $L(A)$.

Beweis: Induktion nach t:

$L_{ij}^0 = \{y \in \Sigma^* : q_i y \overset{1}{\vdash} q_j\}$ ist endlich.

$L_{ij}^{t+1} = L_{ij}^t \cup L_{it+1}^t (L_{t+1t+1}^t)^* L_{t+1j}^t$

$L(A) = \bigcup_{q_j \in F} L_{1j}^n$

7.25 Beispiel

$$1$$

$$q_1 \qquad 0 \qquad q_2 \qquad 1 \qquad q_3$$

$$0 \qquad 0, 1$$

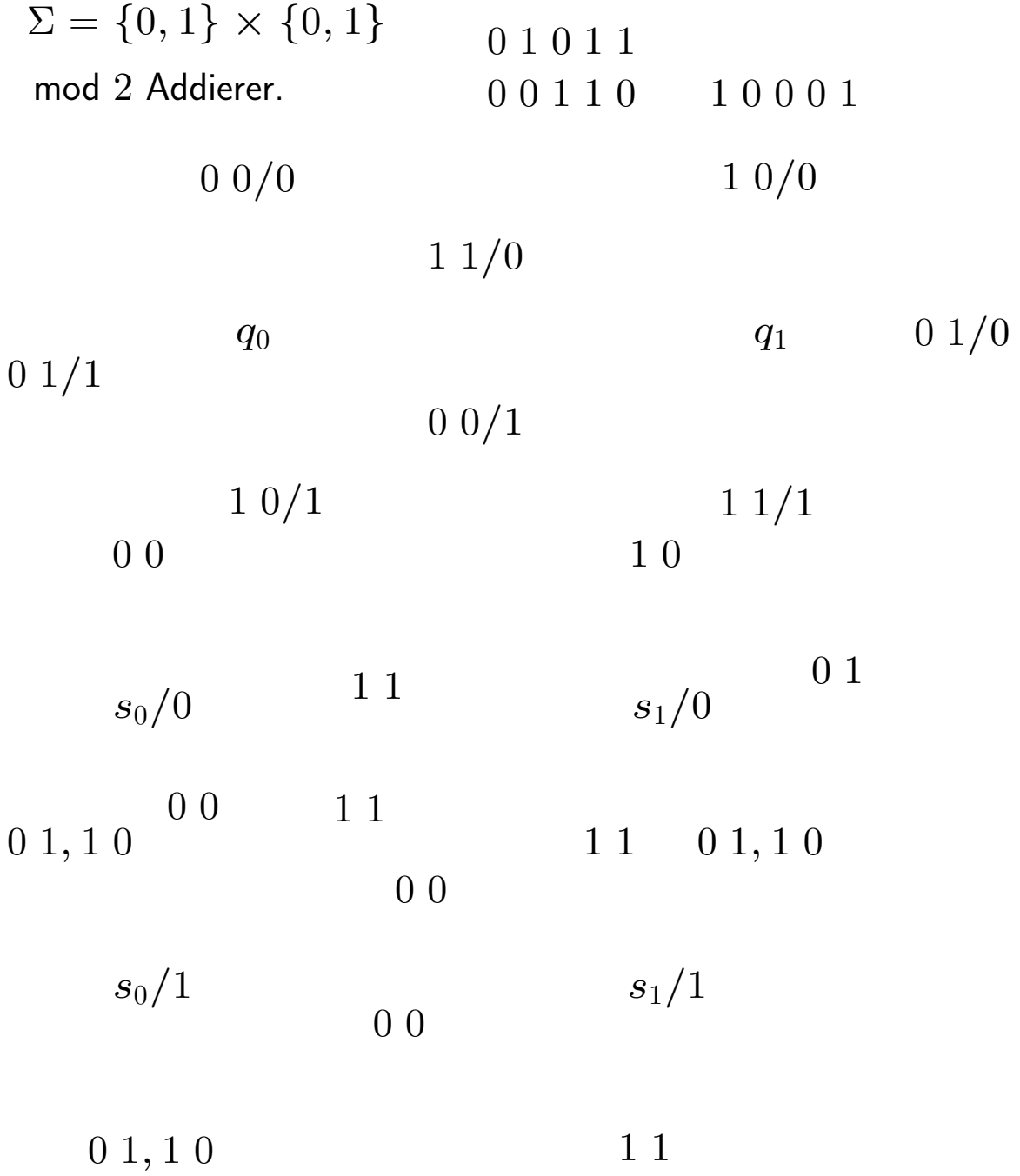
i	j	$t =$	0	1	2	3
1	1		ε	ε	$(00)^*$	
1	2		0	0	$0(00)^*$	
1	3		1	1	0^*1	
2	1		0	0	$0(00)^*$	
2	2		ε	$\varepsilon + 00$	$(00)^*$	
2	3		1	$1 + 01$	0^*1	
3	1		$\varnothing$	$\varnothing$	$(0+1)(00)^*0$	
3	2		$0 + 1$	$0 + 1$	$(0+1)(00)^*$	
3	3		ε	ε	$\varepsilon + (0+1)0^*1$	

Varianten + Verallgemeinerungen EA

Endliche Automaten mit Ausgaben Mealy und Moore Automaten

$\Sigma = \{0,1\} \times \{0,1\}$

mod 2 Addierer.



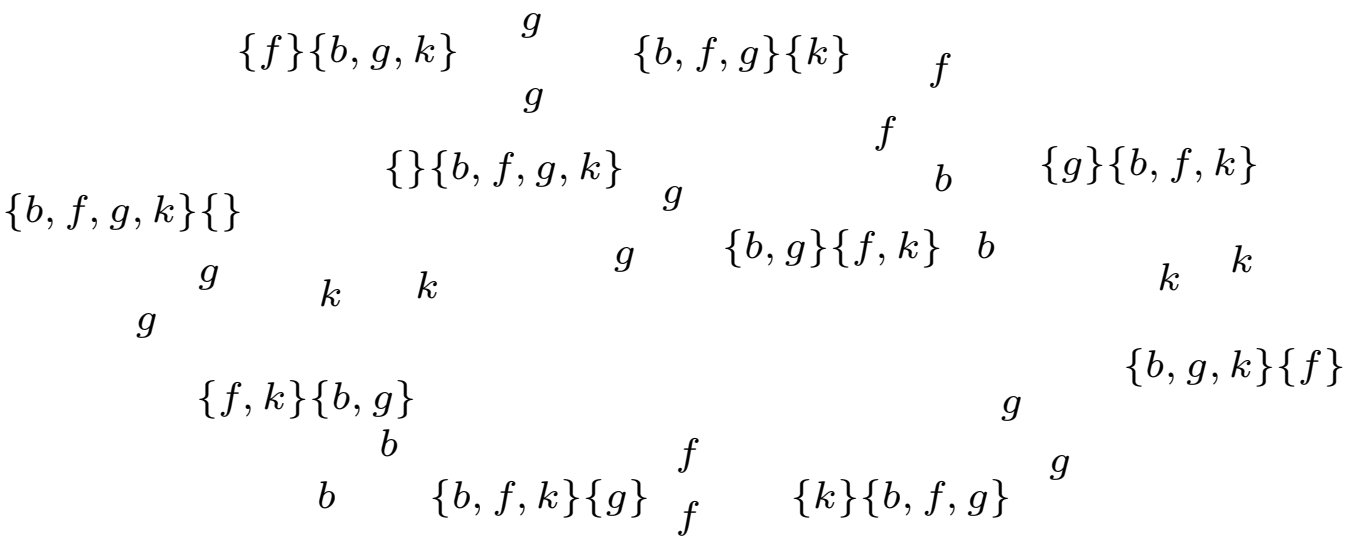
Spezifikation von Prozessen
Dynamisches Verhalten

Statecharts, Petri-Netze, SDL

UML Verhaltensdiagramme (Statecharts, Activity diagrams, MSC)

Event-Condition-Action: $e[c]a$: Übergänge.

Prozess: Bauer/Boot /Fluss, Gans/Fuchs/Korn.



7.4 Kontextfreie Sprachen - Typ2-Sprachen

Erinnerung Sei $G = (N, T, \Pi, Z)$ Grammatik.

G ist vom Typ 2 (kontextfrei), falls $l \to r \in \Pi$, so $l = A, r = z, A \in N, z \in (N \cup T)^*$.

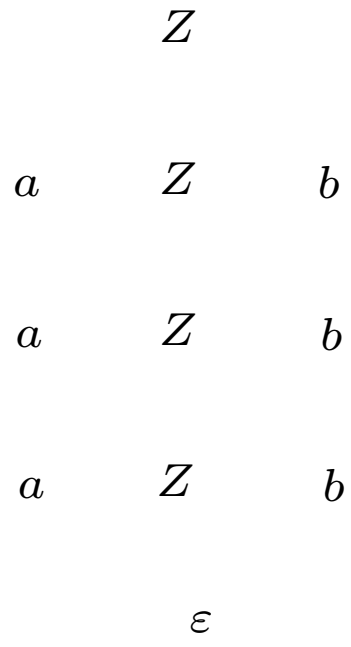
Eine Sprache heißt kontextfrei, falls sie durch eine kontextfreie Grammatik erzeugt werden kann.

Beispiel: $G = (N, T, \Pi, Z)$, $T = \{a, b\}$, $N = \{Z\}$.
$\Pi : Z \to aZb \mid \varepsilon \qquad L(G) = \{a^n b^n \mid n \in \mathbb{N}\}$

Behauptung: $L(G)$ ist nicht rechtslinear. Sei n Konstante für L $y = a^n b^n$. Pumping-Lemma $\rightsquigarrow (a^{k_0})(a^k)^i(a^{k_1})b^n \in L$
für alle $i \in \mathbb{N}$ $(k_0 + k + k_1 = n, k > 0)$ $\lightning$

Gibt es auch ein Pumping-Lemma für kontextfreie Sprachen?

Es ist $aaabbb \in L(G)$. Ableitung als Baum:



Ableitungsbäume - Strukturbäume

7.26 Definition

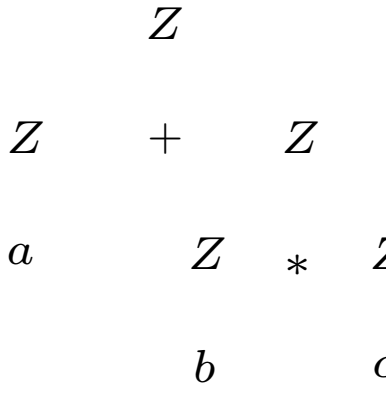
Sei G eine kontextfreie Grammatik und $(Z, u_1, \ldots, u_n)$ eine Ableitung in G. Der Strukturbaum zu dieser Ableitung wird induktiv über n definiert:

1. Der Strukturbaum zur Ableitung (Z) besteht aus einem einzigen mit Z beschrifteten Knoten. Blattwort ist Z.
2. Es sei die Ableitung $(Z, u_1, \ldots, u_n, u_{n+1})$ mit $u_n = uAv$, $u_{n+1} = ub_1 \ldots b_m v$ und eine Produktion $A \longrightarrow b_1 \ldots b_m$ von G mit einzelnen Zeichen b_i gegeben. Sei weiter der Strukturbaum von $(Z, u_1, \ldots, u_n)$ schon konstruiert. Erweitere in diesem Baum den $(|u| + 1)$-ten Knoten (mit dem zu ersetzenden A beschriftet) mit m Folgeknoten, die mit $b_1, \ldots, b_m$ beschriftet sind. (ε als Zeichen erlaubt). Blattwort ist u_{n+1}.

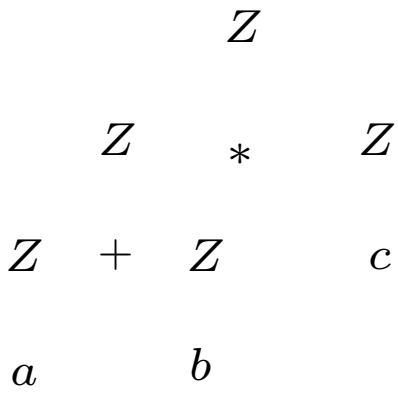
7.27 Beispiel

$G = (N, T, \Pi, Z)$ mit $N = \{Z\}, T = \{a, b, c, +, *\}$,
$\Pi : Z \longrightarrow Z + Z,\ Z \longrightarrow Z * Z,\ Z \longrightarrow a|b|c$

a)

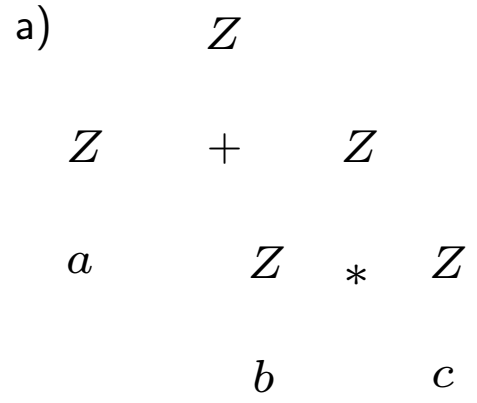


b)



Strukturbäume

a)



Es gibt zu $a + b * c$ verschiedene Ableitungen:

(i) $(\underset{\uparrow}{Z}, \underset{\uparrow}{Z} + Z, a + \underset{\uparrow}{Z}, a + \underset{\uparrow}{Z} * Z, a + b * \underset{\uparrow}{Z}, a + b * c)$

(ii) $(\underset{\uparrow}{Z}, Z + \underset{\uparrow}{Z}, Z + Z * \underset{\uparrow}{Z}, Z + \underset{\uparrow}{Z} * c, \underset{\uparrow}{Z} + b * c, a + b * c)$

Die Ableitungen (i) und (ii) sind unterschiedlich, erzeugen aber denselben Strukturbaum: a).

Desweiteren wird in Ableitung (i) immer das am weitesten links stehende Nichtterminalzeichen ersetzt. (siehe ↑).

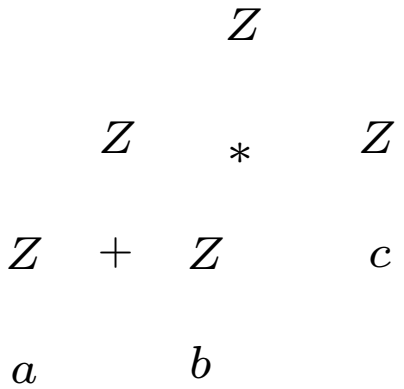
Betrachte die Ableitungen:

(iii) $(\underset{\uparrow}{Z}, \underset{\uparrow}{Z} * Z, \underset{\uparrow}{Z} + Z * Z, a + \underset{\uparrow}{Z} * Z, a + b * \underset{\uparrow}{Z}, a + b * c)$

(iv) $(\underset{\uparrow}{Z}, Z * \underset{\uparrow}{Z}, \underset{\uparrow}{Z} * c, Z + \underset{\uparrow}{Z} * c, \underset{\uparrow}{Z} + b * c, a + b * c)$

Strukturbäume

b)



Ableitungen (iii) und (iv) erzeugen Strukturbaum b).

Insgesamt:

1. Ein Strukturbaum repräsentiert eine Menge von Ableitungen.
2. Ein ableitbares Wort kann verschiedene Ableitungen haben, die nicht durch **einen** Strukturbaum dargestellt werden können.

Punkt 2 kann Schwierigkeiten bereiten, wenn einem ableitbaren Ausdruck eine Semantik (etwa ein Wert) zugeordnet werden soll.

Eindeutigkeit der Termsyntax geht verloren, wenn auf Klammern verzichtet wird. Was ist der Wert von $1 + 2 * 3$?
$(1 + 2) * 3 = 6$
$1 + (2 * 3) = 7$

Eindeutigkeit

7.28 Definition

Eine kontextfreie Grammatik G heißt **eindeutig**, falls für jedes $w \in L(G)$ gilt: Alle Ableitungen von w besitzen denselben Strukturbaum. Eine k.f. Sprache L ist **eindeutig**, falls $L = L(G)$, mit G eindeutig.

7.29 Beispiel Betrachte Grammatik $G = (N, T, \Pi, Z)$ mit $N = \{Z\}$, $T = \{a, b, c, +, *, (,)\}$,

$$\begin{aligned} \Pi: \quad & Z \to (Z + Z) \\ & Z \to (Z * Z) \\ & Z \to a|b|c \end{aligned}$$

G ist eindeutig und somit die Sprache $L(G)$ auch. $\rightsquigarrow$ Übung.

7.30 Definition

Sei G eine kontextfreie Grammatik und $(u_0, u_1, \ldots, u_n)$ eine Ableitung in G. Die Ableitung heißt **Linksableitung** in G, falls für alle $i < n$ u_{i+1} aus u_i durch Ersetzen des am weitesten links stehende Nichterminalzeichen mit Hilfe einer Regel in G entsteht.
(**Rechtsableitung** analog).

7.31 Beispiel G aus vorherigem Beispiel

$$(\underset{\uparrow}{Z}, (\underset{\uparrow}{Z} * Z), ((\underset{\uparrow}{Z} + Z) * Z), ((a + \underset{\uparrow}{Z}) * Z),$$
$$((a + b) * \underset{\uparrow}{Z}), ((a + b) * c))$$

Ableitung für $((a + b) * c)$ $\rightsquigarrow$ Linksableitung.

Eindeutigkeit k.f. Grammatiken

7.32 Lemma

Eine kontextfreie Grammatik ist genau dann eindeutig, wenn jedes durch die Grammatik erzeugte Wort genau eine Linksableitung (bzw. Rechtsableitung) besitzt.

Beweis: Übung.

Beachte:

1. Ist $w \in L(G)$, so gibt es eine Linksableitung zu w.
2. Jede rechtslineare Sprache ist eindeutig.
3. Es gibt sogenannte ererbt mehrdeutige kontextfreie Sprachen, etwa
   $$L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n, m \geq 1\}$$
   Man kann zeigen:
   Jede kontextfreie Grammatik G, die L erzeugt, ist mehrdeutig.

Problem: Wie kann man möglichst effizient testen, ob ein Wort aus einer kontextfreien Grammatik ableitbar ist?

$\rightsquigarrow$ Konstruiere Automaten, der den Strukturbaum einer Ableitung in einer bestimmten Weise aufbaut: Top-Down, Preorder.

LL-Automaten zu einer k.f. Grammatik

7.33 Definition

Sei $G = (N, T, \Pi, Z)$ eine kontextfreie Grammatik. Der **LL-Automat** zu G ist das folgende Tupel

$$A_{LL}(G) = (\{\#\}, N, T, \Pi_{LL}(G), Z\#, \{\#\})$$

Mit folgenden Produktionen in $\Pi_{LL}(G)$:

Für alle $t \in T$ und alle Produktionen
$A \to B_1 \dots B_n \in \Pi$ mit einzelnen Zeichen B_i

$A\# \to B_n \dots B_1\#$	(Produce) (Beachte die Reihenfolge der B's)
$t\#t \to \#$	(Compare)

Ableitbarkeit in A_{LL} bedeutet Ableitbarkeit in diesem Wortersetzungssystem. Die von A_{LL} akzeptierte Sprache ist die Menge

$$\{x \in T^* : Z\#x \underset{\Pi_{LL}(G)}{\vdash} \#\}$$

Initialkonfiguration bei Eingabe $x \in T^* : Z\#x$, d. h.
$i(X) = Z\#x$.

Finalkonfigurationen: $\{\#\}$

7.34 Lemma Sei G eine kontextfreie Grammatik.

Es ist $x \in L(G)$ gdw $x \in L(A_{LL}(G))$.

Beispielkonstruktion

7.35 Beispiel G aus vorherigem Beispiel,

$$\begin{aligned}\Pi_{LL}(G): \quad & Z\# \to\ )Z + Z(\# \\ & Z\# \to\ )Z * Z(\# \\ & Z\# \to a\#|\ b\#|\ c\# \\ & a\#a \to \# \\ & b\#b \to \# \\ & \vdots \\ &)\#) \to \#\end{aligned}$$

Wir wissen $((a + b) * c) \in L(G)$.

Betrachte Ableitung (Berechnung)

$$\begin{aligned}(\quad & Z\#((a + b) * c, \\ & \overset{\cdots\cdots}{)Z * Z(\#((a + b) * c)}, \\ & \overset{\cdots\cdots}{)Z * Z\#(a + b) * c)}, \\ & \overset{\cdots\cdots}{)Z*)Z + Z(\#(a + b) * c)}, \\ & \overset{\cdots\cdots}{)Z*)Z + Z\#a + b) * c)}, \\ & \overset{\cdots\cdots}{)Z*)Z + a\#a + b) * c)}, \\ & \overset{\cdots\cdots}{)Z*)Z + \#+ b) * c)}, \\ & \overset{\cdots\cdots}{)Z*)Z\#b) * c)}, \\ & \vdots \\ & \#)\end{aligned}$$

Spezielle Eigenschaften kontextfreier Sprachen
Pumping-Lemma

Erinnerung: Syntaxanalyse: G Typ-2 Grammatik.

- $w \in L(G)$, so gibt es eine Linksherleitung (Ableitung) für w aus z, d. h.

$$Z \overset{1}{\underset{G}{\vdash}} \alpha_1 \overset{1}{\underset{G}{\vdash}} \alpha_2 \overset{1}{\underset{G}{\vdash}} \cdots \vdash \alpha_n = w$$

- LL-Automat akzeptiert w (simuliert die Linksableitung).
- Zugehöriger Strukturbaum (geordneter markierter Baum, mit Blattwort w).

Z

w

- G ist eindeutig gdw für kein $w \in L(G)$ gibt es zwei verschiedene Strukturbäume.

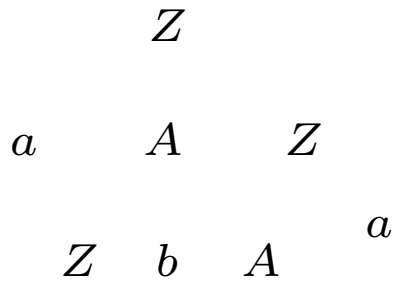
  gdw keine zwei verschiedene Linksableitungen.

  Es gibt kontextfreie Sprachen, die nicht von eindeutiger kontextfreier Grammatik erzeugt werden können.
  z. B. $\{b^m c^m d^l : m, l \geq 1\} \cup \{b^l c^n d^n : l, n \geq 1\}$
  Alle Wörter der Form $b^i c^i d^i \;\; i \geq 1$ sind mehrdeutig.

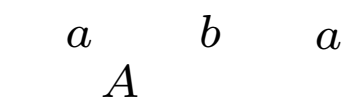
Beispiel: Pumping Eigenschaft

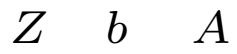
7.36 Beispiel $G = (\{Z, A\}, \{a, b\}, \Pi, Z)$ mit
$\Pi : Z \to aAZ \mid a \qquad A \to ZbA \mid ZZ \mid ba$

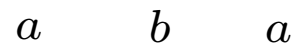
- $Z \vdash aAZ \vdash aZbAZ \vdash aabAZ \vdash aabbaZ \vdash aabbaa$
- Strukturbaum für $aabbaa$



Teilbaum mit Wurzel
A ist Strukturbaum
für Begrenzung vom Teilbaum
$A \underset{G}{\vdash} abba$







Beachte
$A \underset{G}{\vdash} abA \underset{G}{\vdash} (ab)^n A \underset{G}{\vdash} (ab)^n ba$ oder
$Z \underset{G}{\vdash} aabbaZ \underset{G}{\vdash} (aabba)^n Z \vdash (aabba)^n a$

„Aufpumpen“ von Teilwörter bei Wiederholung nichtterminaler Buchstaben.

Pumping Lemma für k.f. Sprachen

7.37 Lemma $G = (N, T, \Pi, Z)$ kontext-freie Grammatik.

Sei $p = \max\{|\beta_i| : \alpha_i \to \beta_i \in \Pi\}$. Ist $\mathcal{B}$ Strukturbaum für $\alpha \in (N \cup T)^*$ der Tiefe h, so gilt $|\alpha| \leq p^h$.
(Da Anzahl der Blätter $\leq p^h$).

7.38 Satz $uvwxy$**-Theorem** (Bar-Hillel, Perles, Shamir).

Sei L eine kontext-freie Sprache. Dann gibt es ein $n \in \mathbb{N}$, so dass für jedes Wort $z \in L(G)$ mit $|z| \geq n$ gilt:
Es gibt eine Zerlegung von z in $uvwxy$ mit $0 < |vx|$ und $|vwx| \leq n$ und für jedes $i \in \mathbb{N}$ ist auch $uv^iwx^iy \in L(G)$.

- (Beachte: Insbesondere ist auch $uwy \in L(G)$).

Beweis-Idee: o.B.d.A. sei L erzeugt von kontext-freier Grammatik G ohne ε-Regeln (bis auf $Z \to \varepsilon$).

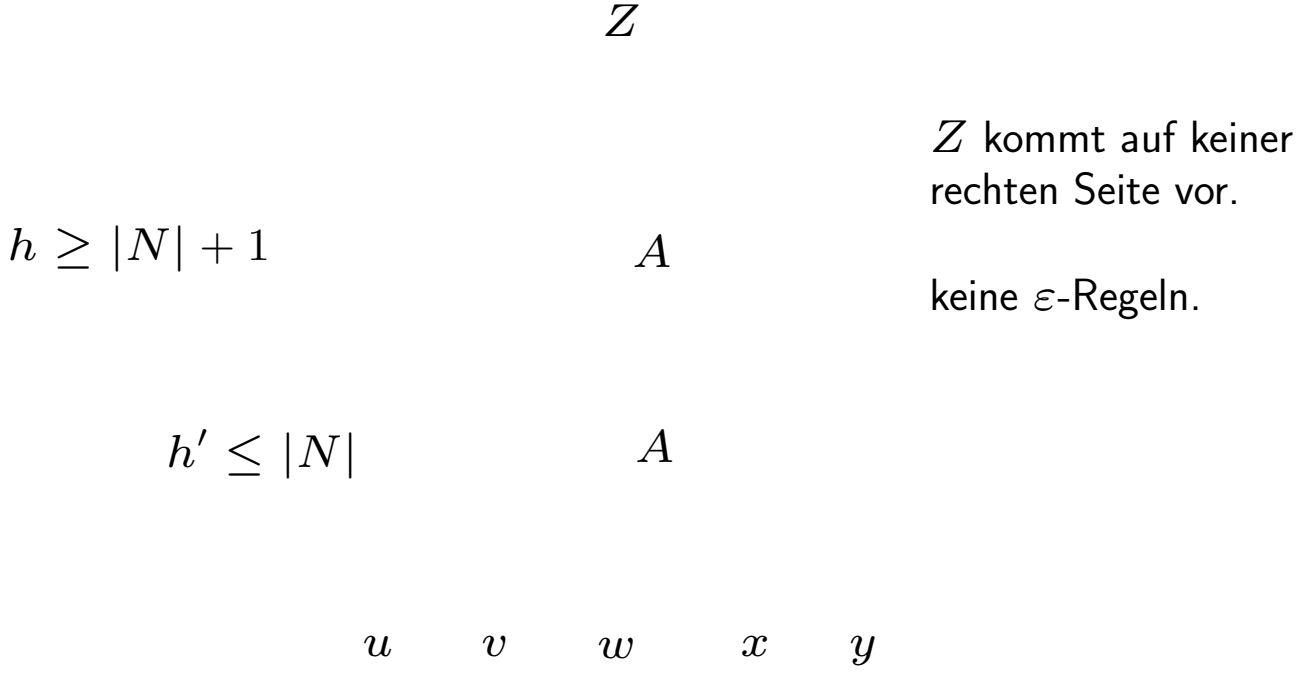
Sei $p = \max\{|\beta| : A \to \beta \in \Pi_G\}$. Betrachte $p^{|N|}$ und $z \in L(G)$ mit $|z| > p^{|N|}$. Ist $\mathcal{B}$ Strukturbaum für z, so ist die Tiefe von $\mathcal{B}$ mindestens $|N| + 1$. Sei $\mathcal{B}$ gewählt von minimaler Tiefe h.

Behauptung: Es gibt $A \in N$ mit

$Z \underset{G}{\vdash} uAy \underset{G}{\vdash} uvAxy \underset{G}{\vdash} uvwxy = z$, wobei
$u, v, w, x, y \in \Sigma^*$, $vx \neq \varepsilon$, $|vwx| \leq p^{|N|}$.
Dann $A \underset{G}{\vdash} vAx$, $A \underset{G}{\vdash} w$, wähle $n = p^{|N|} + 1$.

Beweisargument

Beachte: Analoges Argument führt zu Beweis des Pumping-Lemmas für RL-Grammatiken.



- Innere Knoten sind mit Nichtterminalsymbolen (NT) markiert.
- Da $h \geq |N|+1$, gibt es eine Weg zu Blatt der Länge $\geq |N|+1$
- NT-Symbol (verschieden von Z) wiederholt sich.
- Wähle NT A maximaler Tiefe, d.h. Teilbaum unter A hat Tiefe $\leq |N|$ und $|vwx| \leq p^{|N|}$.
- Dann $vx \neq \varepsilon$, da $\mathcal{B}$ minimaler Tiefe.

$\rightsquigarrow$ Behauptung.

Anwendungen

7.39 Folgerung und Anwendungen

a) Die Sprache $L = \{a^m b^m c^m \mid m > 0\}$ ist **nicht kontextfrei**. Angenommen L ist kontextfrei, n die Konstante vom $uvwxy$-Theorem. Wähle $m > n/3$.
$z = a^m b^m c^m = uvwxy$, $vx \neq \varepsilon$, $|vwx| \leq n$
Enthält v oder x mindestens zwei Buchstaben aus $\{a, b, c\}$, so $uv^2wx^2y \notin L$, da falsche Reihenfolge der Buchstaben.
Falls v und x nur aus a's, b's oder c's, so falsche Anzahl, da nur zwei gekoppelt.

b) $L = \{a^n : n \text{ Primzahl }\} \subseteq a^*$ ist **nicht kontextfrei**. Angenommen ja. Dann ist L RL-Sprache (warum?). Sei n Konstante des Pumping-Lemmas für RL-Sprachen $a^p \in L$ mit $p > n$. Dann ist $a^p = a^i a^j a^k$, $j > 0$, $a^{i+l\cdot j+k} \in L$, $l \geq 0$. D. h. $i + l \cdot j + k$ ist Primzahl für alle l, insbesondere für $l = i + k$
↯

c) Kontextfreie-Sprachen (Typ-2 Sprachen) sind nicht abgeschlossen gegen $\cap$ und $\neg$.
Beweis:
$L_1 = \{a^n b^n c^m : n, m \geq 1\}$, $L_2 = \{a^m b^n c^n : n, m \geq 1\}$ sind kontextfrei, aber $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 1\}$ ist nicht kontextfrei, wegen $L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$ folgt Behauptung.

Anwendungen (Forts.)

d) Sei $G = (N, T, \Pi, Z)$ kontextfreie Grammatik
$p = \max\{|\beta| : A \to \beta \in \Pi\}$, $n = p^{|N|}$. $L(G)$ ist unendlich gdw es gibt $z \in L(G) : n < |z| \leq n \cdot (p+1)$.

Beweis:

„$\Leftarrow$“ Pumping-Lemma.

„$\Rightarrow$“ $z \in L(G)$ minimale Länge mit $|z| > n$. Angenommen $|z| > n \cdot (p+1)$, dann $z = uvwxy \in L(G)$, $0 < |vx| \leq |vwx| \leq n$ und $uwy \in L(G)$ nach Pumping-Lemma. Dann ist $n < |uwy| < |z|$ ↯

Insbesondere ist es entscheidbar, ob $L(G)$ unendliche Sprache für G Typ-2 Grammatik.

e) Beachte: Pumping-Lemma liefern notwendige, jedoch nicht hinreichende Bedingungen für L Typ-2 (3) Sprache:
$\{a^p b^n : p\text{-Primzahl}, n \geq p\}$ ist nicht kontextfrei, dies kann aber nicht mit Pumping-Lemma bewiesen werden.

LL-Automat für G $(\{\#\}, N, T, \Pi_{LL}(G), Z\#, \{\#\})$ kann als Kellerautomat aufgefasst werden. Nur ein Zustand $\#$.

Kontextfreie Sprachen und Kellerautomaten

7.40 Definition

Ein Kellerautomat $K = (Q, N, T, \Pi, iq_0, F)$ mit Q Zustandsmenge, T Eingabealphabet, N Kelleralphabet, $i \in N, q_0 \in Q$, $F \subset Q$. Anfangskonfiguration: Für $x \in T^*$ $i(x) = iq_0x$,
Π Produktionen der Form

$$aqb \to xq' \quad \text{(Lesen eines Zeichens)}$$
$$aq \to xq' \quad \text{(Spontanübergang)}$$

mit $x \in N^*, a \in N, q, q' \in Q$ und $b \in T$.
Die von K akzeptierte Sprache ist die Menge

$$L(K) = \{x \in T^* : iq_0x \underset{\Pi}{\vdash} f \text{ für ein } f \in F\}$$

Lesen eines Zeichens und Spontanübergänge erzeugen in Abhängigkeit eines gewissen Buchstabens im Keller ein neues Wort.

b $\qquad\qquad$ $b\ b'$

a $\qquad$ q $\qquad\qquad$ x $\qquad$ q'

Kelleralphabet
und
Bandalphabet
nicht unbedingt
disjunkt

Beispiele

Deterministische Kellerautomaten:
Für $(a, q) \in N \times Q$ gibt es entweder genau eine Produktion der Form $aq \to xq'$ oder für jedes $b \in T$ genau eine Produktion der Form $aqb \to xq'$. $\rightsquigarrow$ **Deterministische kontextfreie Sprachen**.

7.41 Beispiel

1. $L = \{w \not\in w^{mi} : w \in \{a, b\}^*\}$
   k.f. Grammatik für L: $Z \to aZa \mid bZb \mid \not\in$
   $K = (\{q_0, q_1\}, \{Z, a, b\}, \{a, b, \not\in\}, \Pi, Zq_0, F = \{q_1\})$
   $\Pi ::$ $zq_0a \mapsto zaq_0$ $zq_0b \to zbq_0$ $z \in \{Z, a, b\}$
   $zq_0 \not\in \to zq_1$ $z \in \{Z, a, b\}$
   $aq_1a \to q_1$ $bq_1b \to q_1$
   $Zq_1 \to q_1$
   K ist deterministischer Kellerautomat $L(K) = L$. Also ist L eine deterministische k.f. Sprache.

2. $G = (N, T, \Pi, Z)$, $I = \{a, b\}$, $\Pi : Z \to aZa \mid bZb \mid \varepsilon$
   Dann gilt $L(G) = \{ww^{mi} : w \in T^*\}$.
   Sei K mit $Q = \{q\}$, $N = \{Z, a, b\}$, $q_0 = q$, $i = Z$, und Π_K:
   $aqa \to q$, $bqb \to q$
   $Zq \to aZaq \mid bZbq \mid q$
   (nicht deterministische Produktionen).

Beispiele (Fort.)

Behauptung: $L(K) = L(G)$

„$\supseteq$“ klar.

„$\subseteq$“ $Zqw \vdash q \rightsquigarrow Z$ muss vom Keller gelöscht werden., d. h.

$Zqw \vdash uZqv \overset{1}{\vdash} uqv \vdash q$

$uqv \vdash q$, wobei Z in u nicht enthalten ist.

$\rightsquigarrow$ nur Vergleiche, also $|u| = |v| \wedge u^{mi} = v$

(Ind. $|u|$).

v ist Endwort von w, d. h. $w = xv = xu^{mi}$ und $Zq_0w \vdash uZu^{mi}qxu^{mi} \vdash uZqu^{mi}$, d. h. $2|u|$ Schritte und $w = uu^{mi}$

Induktion nach $|u|$.

Charakterisierungssatz

7.42 Satz

Die kontextfreien Sprachen sind genau diejenigen, die durch einen Kellerautomaten akzeptiert werden.

Beweis: „$\curvearrowright$“ LL-Automat.

„$\curvearrowleft$“ Sei K ein Kellerautomat. o.B.d.A. Finalzustand nur ein Zustand $f \in Q$.

Definiere eine kontextfreie Grammatik G mit nichtterminalen
$N_G = \{[xq, q'] : x \in \Gamma, q, q' \in Q\}$,
Startzustand $Z = [iq_0, f]$,
Terminalsymbolen Σ und Produktionen

$$[xq, q'] \longrightarrow a[x_m q_m, q_{m-1}][x_{m-1} q_{m-1}, q_{m-2}] \cdots [x_2 q_2, q_1][x_1 q, q']$$

Für jeden Befehl $xqa \longrightarrow x_1 \cdots x_m q_m$, $a \in \Sigma \cup \{\varepsilon\}$ und alle $q_1, \ldots, q_{m-1}, q' \in Q$.

Es gilt für $x \in \Gamma, q, q' \in Q$ und $w \in \Sigma^*$

$$(*) \qquad xqw \underset{K}{\vdash} q' \text{ gdw } [xq, q'] \underset{G}{\vdash} w$$

Insbesondere erzeugt also G, die von K akzeptierte Sprache.

Beweis von $(*)$:

Es gelte $[xq, q'] \underset{G}{\vdash} w$. Durch Induktion über die Länge einer Ableitung in G zeige im Kellerautomaten gilt $xqw \underset{K}{\vdash} q'$.

Charakterisierungssatz (Forts.)

Erster Ableitungsschritt

$$[xq, q'] \overset{1}{\underset{G}{\vdash}} a[x_m q_m, q_{m-1}][x_{m-1} q_{m-1}, q_{m-2}] \cdots [x_2 q_2, q_1][x_1 q_1, q'] \underset{G}{\vdash} w$$

mit $a \in \Sigma \cup \{\varepsilon\}$. Somit ist w zerlegbar in Teilwörter $aw_m \ldots w_1$ mit der Eigenschaft $[x_i q_i, q_{i-1}] \underset{G}{\vdash} w_i$.

Für $1 < i \leq m$ und $[x_1 q_1, q'] \underset{G}{\vdash} w_1$.

Nach Induktion vor folgt $x_i q_i w_i \underset{K}{\vdash} q_{i-1}$ für $1 < i \leq m$ und $x_1 q_1 w_1 \underset{K}{\vdash} q'$.

Da es die Regel $xqa \longrightarrow x_1 \cdots x_m q_m$ im Kellerautomaten gibt, erhält man die Ableitung:

$$\begin{aligned} xqw = xqaw_m \ldots w_1 &\underset{K}{\vdash} x_1 \cdots x_{m-1} x_m q_m w_m w_{m-1} \cdots w_1 \\ &\underset{K}{\vdash} x_1 \cdots x_{m-1} q_{m-1} w_{m-1} \cdots w_1 \\ &\underset{K}{\vdash} x_1 \cdots q_{m-2} \cdots w_1 \\ &\cdots \\ &\underset{K}{\vdash} x_1 q_1 w_1 \\ &\underset{K}{\vdash} q' \end{aligned}$$

Charakterisierungssatz (Forts.)

Es gelte umgekehrt $xqw \underset{K}{\vdash} q'$. Induktiv über die Länge einer Ableitung im Kellerautomaten zeigt man nun $[xq, q'] \underset{G}{\vdash} w$.

Betrachte ersten Schritt:

$xqw \overset{1}{\underset{K}{\vdash}} x_1 \cdots x_m q_m v \underset{K}{\vdash} q'$

mit $w = av$, $a \in \Sigma \cup \{\varepsilon\}$. Zerlege die Ableitung von $x_1 \cdots x_m q_m v$ nach q' in m-Phasen, die dadurch definiert sind, dass nach der i-ten Phase im Keller nur noch die Zeichen $x_1 \cdots x_{m-i}$ verbleiben. (Da Keller leer gemacht werden muss). Die Ableitung hat somit die Form

$$\begin{aligned} xqw \quad & \overset{1}{\underset{K}{\vdash}} x_1 \cdots x_m q_m v = x_1 \cdots x_m q_m w_m \cdots w_1 \\ & \underset{K}{\vdash} x_1 \cdots x_{m-1} q_{m-1} w_{m-1} \cdots w_1 \\ & \underset{K}{\vdash} x_1 \cdots q_{m-2} \cdots w_1 \\ & \cdots \\ & \underset{K}{\vdash} x_1 q_1 w_1 \\ & \underset{K}{\vdash} q' \end{aligned}$$

und es gilt $x_i q_i w_i \underset{K}{\vdash} q_{i-1}$ für $1 < i \leq m$, $x_1 q_1 w_1 \underset{K}{\vdash} q'$. Ind.vor $\rightsquigarrow [x_i q_i, q_{i-1}] \underset{G}{\vdash} w_i$, $K_i \leq m$ und $[x_1 q_1, q'] \underset{G}{\vdash} w_1$, also $[xq, q'] \overset{1}{\underset{G}{\vdash}} a[x_m q_m, q_{m-1}] \cdots [x_1 q_1, q'] \underset{G}{\vdash} a w_m \cdots w_1 = av = w$.

Abschlusseigenschaften

7.43 Lemma

Der Durchschnitt einer kontextfreien Sprache mit einer rechts-linearen Sprache ist eine kontextfreie Sprache.

Beweis: Idee: Lasse gleichzeitig Kellerautomat und endlicher Automat ablaufen.

Sei K mit Produktionen der Form

$$xq_Ka \to x'q'_K \text{ bzw. } yq_K \to y'q'_K$$

und A DEA mit Produktionen

$$q_Aa \to q'_A(:= \delta(q_A, a))$$

Bilde Produktautomat $[K, A]$, d. h. $Q = [Q_K, Q_A] \ni [q_K, q_A]$ als Kellerautomat mit Produktionen
$x[q_K, q_A]a \to x'[q'_K, \delta(q_A, a)]$
bzw.
$y[q_K, q_A] \to y'[q'_K, q_A]$
Startzustand $[i_K, i_A]$, d. h. $i(x) = i[{q_0}_K, {q_0}_A]x$ für x Eingabewort.

Finalzustände $[f_K, f_A]$ $f_K \in$ Finalzustand von K, $f_A \in$ Finalzustand von A.

Abschlusseigenschaften (Forts.)

Es gilt

$$\begin{aligned} w \in L(K) \cap L(A) \text{ gdw } & \exists f_K \in F_K \; {i_K}{q_0}_K w \underset{K}{\vdash} f_K \wedge \\ & \exists f_A \in F_A \; {q_0}_A w \underset{A}{\vdash} f_A \\ \text{gdw } & \exists [f_K, f_A] \in F_K \times F_A : \\ & i_K[{q_0}_K, {q_0}_A] w \underset{[K,A]}{\vdash} [f_K, f_A] \\ \text{gdw } & w \in L([K, A]) \end{aligned}$$

7.44 Beispiel

$L = \{ww : w \in \{a, b\}^*\}$ ist keine kontextfreie Sprache.

Sei $R = a^+b^+a^+b^+$ rechts-lineare Sprache (warum?)

$L \cap R = \{a^i b^j a^i b^j : i \geq 1, j \geq 1\}$ ist keine kontextfreie Sprache.

Angenommen JA: Pumping-Lemma für kontextfreie Sprachen: Sei $n \in \mathbb{N}$ die Konstante für die kontextfreie Sprache $L \cap R$.

Wähle $i = j = n \quad a^n b^n a^n b^n \in L \cap R$.

$uvwxy$ Zerlegung: $\underset{|vwx| \leq n}{\vdash\!\!\!\dashv} a^n b^n a^n b^n = uvwxy$

$\rightsquigarrow$ Kopplung zwischen $a-$EXP und $b-$EXP kann nicht aufrechterhalten werden $\lightning$

(Frage: Ist L kontext-sensitive Sprache?)

Bemerkung zu Pumping Lemmata

Beachte Quantoren bei Pumping Lemmata.

$$
\begin{array}{lllll}
\forall & \exists & \forall & \exists & \forall \quad uv^iw \in L \\
L \in \mathcal{L}_3 & n \in \mathbb{N} & z \in L & u,v,w \in \Sigma^* & i \\
 & & |z| \geq n & z = uvw & \\
 & & & 0 < |v| \leq n &
\end{array}
$$

$$
\begin{array}{lll}
L \in \mathcal{L}_2 & u,v,w, & \\
 & x,y \in \Sigma^* & uv^iwx^iy \in L \\
 & z = uvwxy & \\
 & vx \neq \varepsilon & \\
 & |vwx| \leq n &
\end{array}
$$

Es gibt schärfere Versionen dieser Lemmata. z. B. $|uv| \leq n$ oder $|vw| \leq n$ für rechts-lineare Sprachen. Odgen's Lemma für kontext-freie Sprachen (Man darf sogar gewisse Buchstaben markieren).

Wortproblem für kontextfreie Grammatiken

G kontextfreie Grammatik. $w \in \Sigma^*$ $w \in L(G)$? Wortproblem ist primitiv rekursiv entscheidbar. (schlechte obere Schranke!)

Kellerautomat der $L(G)$ akzeptiert Ist dieser effizient?

Problem:

- keine Endeutigkeit (mehrere Strukturbäume)
- Kellerautomat ist nicht-deterministisch.
- Falls deterministischer Kellerautomat möglich, so effizienter.

Beachte Beispiele:

- Boolesche Formeln über Signatur (PL-Formeln)
- Terme über Signatur
- Formeln über Signatur
- While Programme über Signatur

Mehrere Regeln mit gleicher linken Seite!

Verallgemeinerung der deterministischen Kellerautomaten

Mit Vorausschau $n \in \mathbb{N}$, falls in Abhängigkeit vom Kellerinhalt und den n-nächsten Eingabezeichen eindeutig die Möglichkeit besteht, die einzig richtige, als nächstes anzuwendende Produktion zu finden.

1-Vorausschau $\{a^n b^n : n \geq 1\}$

Schlagwort LR(k)-LL(k) Analyse.

7.45 Definition Normalformen für kontext-freie Grammatiken

Sei G eine kontext-freie Grammatik, G ist in

• **Chomsky-Normalform**: Produktionen der Form

$$A \to BC \text{ oder } A \to a \qquad A, B, C \in N, a \in T$$

• **Greibach-Normalform**: Produktionen der Form

$$A \to a\alpha \qquad A \in N, a \in T, \alpha \in N^*$$

7.46 Satz

Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ gibt es eine kontextfreie Grammatik G' in Chomsky-Normalform, mit $L(G) = L(G')$.

Die Transformation $G \rightsquigarrow G'$ ist effektiv.

Beweisidee

Beweisidee: $G = (N, T, \Pi, Z)$ kontextfrei.

1. Schritt: ε**-frei**: Da $\varepsilon \notin L(G)$, gibt es eine äquivalente Grammatik G', die keine Produktionen der Form $A \to \varepsilon$ enthält.

2. Schritt: **Normierte Terminierung**: Zu $G = (N, T, \Pi, Z)$ gibt es eine äquivalente Grammatik $G' = (\tilde{N}, T, \Pi', Z)$, die nur Produktionen $A \to a$ mit $a \in T$ und $A \to \alpha$ mit $\alpha \in \tilde{N}^*$ enthält.
Sei $\tilde{N} = N \cup \{A_a : a \in T\}$. Π' entsteht aus Π indem jedes $a \in T$ in Π durch A_a ersetzt wird, vereinigt mit $\{A_a \to a : a \in T\}$ Platzhalter.

3. Schritt: **Keine Kettenproduktionen**: Zu einer Grammatik $G = (N, T, \Pi, Z)$ gibt es eine äquivalente Grammatik $G' = (N, T, \Pi', Z)$, die keine Produktionen der Form $A \to B$ mit $A, B \in N$ enthält (siehe NEA).
Sei $M = \{(A, B) \in N^2 : A \underset{G}{\vdash} B\}$
(lässt sich berechnen: Entferne Zyklen $(A, B), (B, A) \in M$. Beginne mit (A, A).)
$\Pi' = \Pi \backslash \{A \to B : A, B \in N\}$ vereinigt mit Produktionen $A \to r'$, $|r'| > 1$, die aus Produktionen $A \to r \in \Pi$ durch Ersetzen mancher B in r durch C mit $(B, C) \in M$ entstehen, vereinigt mit $A \to a$ für alle $(A, A_a) \in M$.

Beweisidee (Forts.)

4. Schritt: Chomsky-Normalform: Produktionen der Form $A \to B_1 \dots B_n$, $n > 2$ ersetzen: Dazu $A \to B_1 H_1, H_1 \to B_2 H_2, \cdots, H_{n-3} \to B_{n-2} H_{n-2}, H_{n-2} \to B_{n-1} B_n$ mit neuen Nichterminalsymbolen $H_1, \dots, H_{n-2}$.

Falls $\varepsilon \in L(G)$, so ist $(N \cup \{Z'\}, T, \Pi \cup \{Z' \to Z, Z' \to \varepsilon\}, Z')$, wobei (N, T, Π, Z) in Chomsky-Normalform.

7.47 Beispiel Sei $G = (\{Z, A, B\}, \{a, b\}, \Pi, Z)$

$$
\begin{array}{lll}
\Pi: & Z \to bA & Z \to aB \\
& A \to a & B \to b \\
& A \to aZ & B \to bZ \\
& A \to bAA & B \to aBB
\end{array}
$$

1. Schritt: ε-frei: ok.

2. Schritt: $Z \to A_b A \mid A_a B$

$A \to a, B \to b, A_a \to a, A_b \to b$

$$
\begin{array}{ll}
A \to A_a Z & B \to A_b Z \\
A \to A_b AA & B \to A_a BB
\end{array}
$$

3. Schritt: Keine Kettenproduktionen: ok.

4. Schritt: Letzte Zeile oben:

$$A \to A_b C_1, C_1 \to AA$$
$$B \to A_a D_1, D_1 \to BB$$

Auswirkungen auf Strukturbaum? (binär)

$\rightsquigarrow$ Pumping Lemma Konstante: $2^{|N|} + 1$.

$x \in L(G) \rightsquigarrow x$ ist in höchstens $2|x| + 1$ Schritten in G ableitbar (exponentieller Aufwand für Entscheidung $x \in L(G)$).

7.5 Algorithmus von Cocke-Kasami-Younger

7.48 Satz

Sei G in Chomsky-Normalform. Dann gibt es einen Algorithmus der das Wortproblem für G mit Laufzeit $O(n^3)$ entscheidet.

$w \in L(G) \qquad |w| = n$ Laufzeit $O(n^3)$

Beweis: Sei $w = a_1 \dots a_n$,

$L_{ij}(w) = \{A \in N : A \underset{G}{\vdash} a_i \dots a_j\} \qquad (i \leq j)$

Es gilt $w \in L(G)$ gdw $Z \in L_{1n}(w)$.

Wie berechnet man aus w die L_{ij}. **Dynamisches Programmieren**.

Induktiv über $j - i$ Berechnung von L_{ij}:

- $j - i = 0 : L_{jj} = \{A : A \to a_j \in \Pi\}$
  (da Chomsky-Normalform)

Algorithmus von Cocke-Kasami-Younger

- $j - i > 0$: Berechne L_{ij} aus L_{ik-1} und L_{kj} für ein k mit $i < k \leq j$, wobei $A \in L_{ij}$, falls $A \to BC \in \Pi$, $B \in L_{ik-1}$, $C \in L_{kj}$.

In jedem Schritt müssen maximal $2n$ Mengen betrachtet werden und es gibt weniger als n^2 Mengen L_{ij}, daher kann die Laufzeit durch cn^3 beschränkt werden, wobei c eine Konstante ist, die von der Grammatik G abhängt.

Verwaltung mithilfe einer Erkennungs-Matrix

$j-i$ \ i	1	2	...	n
0	{}	{}		{}
1	{}	...	{}	
2	⋮			
⋮	⋮			
$n-1$	{}			

Beispiel:

$Z \to CB \mid FA \mid FB$
$A \to CZ \mid FD \mid a$
$B \to FZ \mid CE \mid b$
$D \to AA, E \to BB, C \to a, F \to b$

$w = aababb$, $|w| = 6$

Teilw. Länge	$j-i$	$i=1$	2	3	4	5	6	
1	0	A, C	A, C	B, F	A, C	B, F	B, F	(n Einträge)
2	1	D	Z	Z	Z	E, Z		($n-1$ Eintr.) Kosten 1
3	2	A	A	B	A, B			($n-1$ Eintr.) Kosten 2
4	3	D	Z	Z, E				⋮
5	4	A	A, B					⋮
6	5	D, Z						1-Eintrag Kosten $n-1$

$$n + \sum_{i=2}^{n} (n-i+1)(i-1) = \frac{n^3+5}{6}$$

Auf Mehrband TM mit Zeit n^3 realisierbar. Siehe z. B. Hopcroft/Ullman Automaten + formale Sprachen.

Viele Verbesserungen: Mit Einschränkungen oft $O(n)$ möglich! (Vorausschau 1 Det.).

7.6 Unentscheidbare Probleme für kontextfreie Grammatiken

Unentscheidbare Probleme für allgemeine Grammatiken

- Wortproblem
- $L(G) = \varnothing$
- $L(G) = \Sigma^*$
- $L(G)$ endlich
- $L(G_1) = L(G_2)$
- $\varepsilon \in L(G)$?

Für rechts-lineare-Grammatiken alle entscheidbar.

Für kontexfreie Grammatiken? Wortproblem, $L(G)$ endlich ?, $L(G) = \varnothing?, \varepsilon \in L(G)?$ entscheidbar.

7.49 Satz

Sind G_1, G_2 kontextfreie Grammatiken.
Es ist unentscheidbar, ob die zugehörigen Sprachen disjunkt sind.

Folgendes Problem ist nicht rekursiv entscheidbar:

Eingabe: kontextfreie Grammatiken G_1, G_2.

Frage: $L(G_1) \cap L(G_2) \neq \varnothing$?

Unentscheidbare Probleme für kontextfreie Grammatiken (Forts.)

Beweis: Reduktion des PCP auf dieses Problem.

Sei $\mathcal{L} = (u_1 \sim v_1, \ldots, u_k \sim v_k)$, $u_i, v_i \in \Gamma^+$, $k \geq 1$.

Sei $J = \{1, \ldots, k\}$, $J \cap \Gamma = \varnothing$.

Definiere Grammatiken $G_j = (N_j, T, \Pi_j, Z_j)$, $j = 1, 2$.

$T = \Gamma \cup J$, $N_j = \{Z_j\}$

$\Pi_1 = \{Z_1 \to u_i Z_1 i \mid u_i i : i = 1, \ldots, k\}$ $2k$-Regeln

$\Pi_2 = \{Z_2 \to v_i Z_2 i \mid v_i i : i = 1, \ldots, k\}$ $2k$-Regeln

$$
\begin{aligned}
L(G_1) \cap L(G_2) \neq \varnothing \quad & \text{gdw } \exists x \in \Sigma^* \;\; x \in L(G_1) \cap L(G_2) \\
& \text{gdw } \exists t_1, t_2 \in J^* \\
& \qquad x = U(t_1)t_1^{mi} = V(t_2)t_2^{mi} \\
& \text{gdw } \exists t \in J^+ \;\; x = U(t)t^{mi} = V(t)t^{mi} \\
& \text{gdw } \exists t \in J^+ \;\; U(t) = V(t) \\
& \text{gdw } PCP(\mathcal{L})
\end{aligned}
$$

Beachte:
Die Konstruktion liefert „einfache“ k.f. Grammatiken G_1 und G_2:
Sie sind **linear** ($A \to uBv$ Regeln) und **eindeutig**: nur eine Linksableitung möglich!

Folgerungen

• Es gibt kein effektives Verfahren, um für zwei kontextfreie Grammatiken G_1, G_2 eine kontextfreie Grammatik G zu bestimmen mit $L(G) = L(G_1) \cap L(G_2)$. (Begründung: $L(G) \neq \varnothing$ ist für kontextfreie Grammatiken entscheidbar).

• Man kann jedoch eine kontextsensitive Grammatik berechnen mit $L(G) = L(G_1) \cap L(G_2)$, d. h. $L(G) \neq \varnothing$ ist nicht entscheidbar für kontextsensitive Grammatiken.

7.50 Satz Das Mehrdeutigkeitsproblem für kontextfreie Grammatiken ist unentscheidbar.

Eingabe: G kontextfreie Grammatik.

Frage: Ist G mehrdeutig?

Beweis:

PCP auf Mehrdeutigkeitsproblem reduzieren: Seien G_1 und G_2 die kontextfreien Grammatiken wie oben zu PCP $\mathcal{L}$ konstruiert.

$$G_{\mathcal{L}} := (\{Z, Z_1, Z_2\}, \Gamma \cup J, \Pi_1 \cup \Pi_2 \cup \{Z \to Z_1, Z \to Z_2\}, Z)$$

$\mathcal{L} \rightsquigarrow G_{\mathcal{L}}$ effektiv. $L(G_{\mathcal{L}}) = L(G_1) \cup L(G_2)$

G_1, G_2 sind eindeutig.

$G_{\mathcal{L}}$ ist mehrdeutig gdw $L(G_1) \cap L(G_2) \neq \varnothing$
gdw $PCP(\mathcal{L})$

Weitere unentscheidbare Probleme für kontextfreie Grammatiken

7.51 Satz

Folgende Probleme für kontextfreie Grammatiken sind nicht entscheidbar.

1)	$P_1(G)$	gdw $L(G) = \Sigma^*$ (G über $\Sigma = T$)
2)	$P_2(G)$	gdw $L(G)$ ist rechts-linear
3)	$P_3(G)$	gdw $\neg L(G)$ ist kontextfrei (rechts-linear, unendlich)
4)	$P_4(G)$	gdw $L(G_1) = L(G_2)$ (beide über T)
5)	$P_5(G)$	gdw $L(G_1) \subseteq L(G_2)$
6)	$P_6(G_1, G_2)$	gdw $L(G_1) \cap L(G_2)$ ist kontextfrei
7)	$P_7(G_1, G_2)$	gdw $L(G_1) \cap L(G_2)$ unendlich
8)	$P_8(G_1, G_2)$	gdw $L(G_1) \cap L(G_2)$ RL-Sprache

$\mathcal{L}_{\text{endl}} \subsetneq \mathcal{L}_3 \subsetneq \mathcal{L}_{\text{det-kf}} \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_{\text{prim-rek}} \subsetneq \mathcal{L}_{\text{entsch.}} \subsetneq \mathcal{L}_0 = \mathcal{L}_{\text{rek-aufzb.}}$

Kontextsensitive Grammatiken und Sprachen

Erinnerung: Die Sprache $\{a^n b^n c^n : n \in \mathbb{N}\}$ ist eine Typ-1 Sprache: Grammatik $(\{Z, A, B, H, C\}, \{a, b, c\}, \Pi, Z)$ mit Produktionen $\Pi = \{Z \to \varepsilon \mid Ac, A \to ab \mid aACB, CB \to CH, CH \to BH, BH \to BC, B \to b, Cc \to cc\}$.
Sie ist nicht kontexfrei.

Das Wortproblem für k.s. Grammatiken ist entscheidbar. Man muss nur Ableitungen bis zur Länge $(|N| + |T| + 1)^{|x|} + 1$ durchsuchen (ansonsten enthält die Ableitung zwei identische Wörter mit Länge $\leq |x|$).

Ein **linear beschränkter Automat (LBA)** ist eine (nichtdeterministische) Turing-Maschine, deren Lese-/Schreibkopf den Bereich, auf dem beim Start die Eingabe steht, nicht verlassen darf.
• Die Typ-1-Sprachen sind genau die Sprachen, die sich mit einem LBA akzeptieren lassen.

Solche Automaten lassen sich auch durch Produktionen Charakterisieren. Sie haben die Gestalt:
$qa \to q'a' \qquad q, q' \in Q, a, a' \in N \cup T$
$qa \to aq' \qquad q, q' \in Q, a, a' \in N \cup T$
$bqa \to q'ba \qquad q, q' \in Q, a, a' \in N \cup T$
letztere für alle $b \in N \cup T$ falls keine andere diese linke Seite hat.

Kontextsensitive Grammatiken und Sprachen (Fort.)

Es gibt weitere Charakterisierungen der k.s. Sprachen durch spezielle Grammatiken. Eine Grammatik $G = (N, T, \Pi, Z)$ heißt **erweiternd**, falls Π nur Produktionen der Form $l \longrightarrow r$ mit $l \neq \varepsilon$ und $|r| \geq |l|$ enthält. Es gilt: Zu jeder erweiternden Grammatik gibt eine äquivalente k.s. Grammatik.

8 Grundlagen der Programmierung
Zusammenfassung

Zusammenfassung
Ausblick

Grundlagen für die Entwicklung von Software-Systemen
Aktivitäten: **Spezifikation - Entwurf - Implementierung**

Benötigt: Formale Beschreibungstechniken

- Syntax
- Semantik

Spezifikation: Was soll ein SW-System leisten?

Funktionalität: Beschreibung funktionaler Eigenschaften.
Natürliche Sprache, Logik (Vor- Nachbedingungen).
Hier nur Aussagen- und Prädikatenlogik.

Abstrakte Datentypen $\equiv$ Algebren über Signatur.

Was: Axiome. Oft genügen „=“-Axiome, d. h. Gleichheitslogik. Bedingte Gleichungen sind standard.

Beispiel: Keller(X) X Parameter $=_X$ definiert

empty: $\longrightarrow$ stack; is_empty: stack $\longrightarrow$ bool;
push: elem, stack $\longrightarrow$ stack;
pop: stack $\longrightarrow$ elem;

Formale Spezifikationstechniken (Forts.)

Axioms:

All-Quantif.: $\mathsf{pop}(\mathsf{push}(x, y)) = x$
$\mathsf{is\_empty}(\mathsf{empty}) = \mathsf{true}$
$\mathsf{is\_empty}(\mathsf{push}(x, y)) = \mathsf{false}$

$\rightsquigarrow$ **Formale Spezifikationstechniken**

Benötigt: Logik, Modelltheorie

Nicht funktionale Eigenschaften:

z. B. Zeitverhalten, Platzverhalten,. . . „Komplexität“

$\rightsquigarrow$ Klassifikation der berechenbaren Funktionen/Prädikate (Relationen).

Hier: Nicht jede Funktion ist berechenbar. Komplexitätsmaße.

Präzisierung der Berechenbarkeit

- While-berechenbare Funktionen
- μ-rekursive Funktionen $\mathcal{R}_p(\Sigma)$
- RM-berechenbare Funktionen Techniken
- Turing-berechenbare Funktionen **Simulation**

Formale Spezifikationstechniken (Forts.)

Komplexitätsmaß für ein Berechnungskonzept φ ist eine Abbildung $\Phi : \mathbb{N}^2 \longrightarrow \mathbb{N}$ mit den Eigenschaften

$\Phi(i,n) \downarrow \text{ gdw } \varphi_i(n) \downarrow$

$\{(i,n,m) \in \mathbb{N}^3 : \Phi(i,n) \leq m\}$ ist entscheidbar.

z. B. While Programme: Laufzeit eines While-Programms

$$\Phi(p,x) = \mu t.\text{first}(i^t(inp(p,x))) = 0$$

wobei i Interpreterfunktion.

- **Zeitkomplexität** der i-ten TM bei Eingabe n
  $\Phi(i,n) \downarrow \text{ gdw } \varphi_i(n) \downarrow$
  $\Phi(i,n) = \max_R\{t$: Es gibt eine Berechnung R der Länge t der i-ten TM auf Eingabe $n\}$
- **Platzkomplexität** bei TM
  $\Phi(i,n)$ Anzahl der verschiedenen Bandstellen, die die i-te TM auf Eingabe n höchstens besucht.
  $\{\Phi(i,n) \leq m\}$ ist entscheidbar.
  Laufzeit muss $\leq m \cdot |Q||\Gamma|^m + 1$ (Anzahl der Konfigurationen der Länge $\leq m$).
- Bedarf an **Speicherplatz** in einem Goto-Programm.
  z. B. Anzahl der Register (**Einheitskostenmaß**)
  oder Anzahl der Register und Größe der Zahlen (**log-Kostenmaß**)

Wichtige Begriffe

Beachte:

Ist Φ ein Komplexitätsmaß und $B : \mathbb{N} \longrightarrow \mathbb{N}$ eine totale berechenbare Funktion. Dann gibt es eine totale, berechenbare Funktion $f : \mathbb{N} \longrightarrow \{0, 1\}$, so dass jedes Programm i, das f berechnet, für fast alle Werte n eine Komplexität $\Phi(n) \geq B(n)$ hat.

Es gibt beliebig komplexe Funktionen (**Diagonalisierung!**)

$\rightsquigarrow$ - **Komplexitätstheorie, Komplexitätsklassen**

DTime$(s(n))$, NTime$(s(n))$, DSpace$(s(n))$, NSpace$(s(n))$

$$P = \bigcup_{pol} \text{DTime}(pol), \qquad NP = \bigcup_{pol} \text{NTime}(pol)$$

- **Reduzierbarkeit**: $P \leq_m Q$

Verfeinerungen: One-one Reduzierbarkeit: Injektion
pm: Pol-Zeit berechenbare Reduktionen.
Wichtig für die Klassen P, NP.

- **Vollständigkeitsbegriff**: z. B.

$$K = \{a \mid \varphi_a(a) \downarrow\}, K_0 = \{(a, x) : \varphi_a(x) \downarrow\}$$

  Sind vollständig in der Klasse der rekursiv aufzählbaren Relationen.
- **Rekursionstheorie**: Universelle Funktionen $\varphi_p^{(n)}(x_1, \ldots, x_n)$
  SMN-Theorem: Stelligkeiten der universellen Funktionen.

Wichtige Begriffe (Forts.)

Ergebnisse sind richtig für jede **zulässige Aufzählung** der berechenbaren Funktionen. D.h. beschränkte Berechenbarkeit muss entscheidbar sein und SMN-Satz muss gelten.

$\rightsquigarrow$ Charakterisierung der r.a. Relationen, Rekursionssatz, Fixpunktsatz sowie die Sätze von RICE über **Indexmengen**

$$S \subset \mathcal{R}_p(\mathbb{N}) \qquad Ind(S) = \{p : \varphi_p \in S\}$$

- **Existenz berechenbarer Funktionen mit bestimmten Eigenschaften**.
  Verwende Churchsche These, SMN-, Rekursionssatz oder FPS.

- **Nicht Entscheidbarkeit** oder **nicht rekursiv aufzählbar**.
  Verwende Reduktion von K oder K_0 auf Relationen oder Rice Sätze, falls Indexmengen.
- Weitere Klassen: superrekursiv
  subrekursiv

Formale Beschreibungstechniken

Welcher Aspekt eines Systems soll beschrieben werden?

Wie soll beschrieben werden?

Nur das **was** oder **was** und **wie**.

- **Funktionale Aspekte**
- Verhaltensaspekt (Temporale Logik, Statecharts, SDL,...)
- Entwurfsaspekt (logischer Entwurf, Systemstruktur, „Architektur“)
- Implementierungsaspekt (Programmiersprache, abstrakte Maschine, Compiler,...)

Zielsetzung: Übergänge zwischen Beschreibungen sollten „natürlich“-sein, Werkzeug-unterstützt.

$\rightsquigarrow$ Verifikation, Validierung, Testen sollten ermöglicht werden.

- Ableitung partieller Korrektheitsaussagen
- Verifikationsbedingungen
- Testgeneratoren
- Dokumenterstellung

Programmiersprachen

Deklarativ, Funktional (μ-rekursiver Ausdruck), Prozedural (While-Programm), Maschinennahe TM-, RM-Programme.
$\rightsquigarrow$ **Syntax, Semantik**

Syntax Programmiersprachen (allg. Beschreibungstechniken)

Grammatiken

i. Allg. kontextfrei (Teile kontext sensitiv: Prozedurdeklarationen, $\rightsquigarrow$ attributierten Grammatiken).

Grundlage: **Kalküle**

- Syntaktische Definition von Objektmengen
  Regeln, Ableitungen
- Müssen nicht immer Zeichenreihen sein
- Können auch graphische Objekte sein
- Oft beides zusammen

Beispiel: Terme, Formeln, Programme, gültige Formeln, gültige partielle Korrektheitsaussagen, Diagramme, UML,...

Speziell für Zeichenreihen: **Wortersetzungssysteme**.

Grundlage für: Grammatiken (RL, KF, KS ...)
Automaten (EA, NDEA, KA, TM ...)
Algebraische Strukturen: Monoide, Gruppen, Algebren

Z.B.: Prozessalgebren: $(a, b;\ ba \longrightarrow ab)$ kommutatives Monoid.

Termersetzungssysteme: Signatur (S, Σ)

Listen: nil $:\longrightarrow L$, cons : $X, L \longrightarrow L$ append : $L, L \longrightarrow L$

append(nil, l) $\Rightarrow l$

append(cons(x, l), l') $\Rightarrow$ cons(x, append(l, l'))

Kalküle (Fort.)

Funktionale Programmiersprachen: Applikative Programme.

(Bedingte Gleichungen ...) $\rightsquigarrow$ **Programmieren mit Gleichungen**

Beweiskalküle:

Aussagenlogik: Axiome (Schemata)

- $A \rightarrow (B \rightarrow A)$
- $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
- $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

Regeln: MP Modus Ponens oder Abtrennungsregel (cut Regel)

$$\frac{\alpha, \alpha \rightarrow \beta}{\beta}$$

$Ax \vdash \alpha$, so α Tautologie.

Beweis: $\alpha_1, \ldots, \alpha_n \qquad \alpha_n = \alpha$
$\alpha_i \in Ax$ oder $\alpha_j, \alpha_k = \alpha_j \rightarrow \alpha_i$, $j, k < i$

Kompaktheitssatz: $\Sigma \vdash \alpha$, so gibt es eine endliche Teilmenge $\Sigma_0 \subset \Sigma : \Sigma_0 \vdash \alpha$.

Verallgemeinerungen: - Prädikatenlogik.
- Andere Logiken. $\rightsquigarrow$ Logik Vorlesung.

Logik, Prozessbeschreibungssprachen

Grundlage automatischer Beweiser

Beachte: $Nat = (\mathbb{N}, 0, 1, +, \cdot, <)$ Theorie von Nat (d. h. gültige Formeln in Nat) nicht rekursiv aufzählbar!

Grundlage für Prozess-BT: **Endliche Automaten**

- Endliche Automaten
- Petri-Netze (Verallgemeinerungen)
- Statecharts, SDL $\cdots$
- $\vdots$

Semantik von Beschreibungstechniken

- Operational (Zustandsübergänge)
- Denotional (Programme „bezeichnen“ Funktionen)
- Transformation (Übersetzen in Konstrukte mit wohldefinierter Semantik)

Interpreterfunktionen, Compiler, abstrakte Maschinen, Termersetzung,...

Wichtige Eigenschaften von BT:

Modularisierung, **Parametrisierung**, **Kompositionsmöglichkeiten**, **Verfeinerung** (Abstraktionsmöglichkeiten).