Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Grammatiken und die Chomsky-Hierarchie

INSTITUT FÜR THEORETISCHE INFORMATIK

Thema dieses Kapitels

- Grammatiken sind Regelsysteme, mit denen sich die Wörter einer vorgegebenen Sprache erzeugen lassen

Beispiel 1

Die Sprache aller Graphen $G = (V, E)$, die eine Clique der Größe $\frac{|V|}{2}$ enthalten, lässt sich aufbauen durch:

1. Wahl der Zahl n für $|V|$
2. Wahl einer Teilmenge der Größe $\frac{|V|}{2}$
3. Zugabe aller Kanten zwischen Knoten aus dieser Teilmenge
4. Zugabe weiterer Kanten

Dieses Regelsystem ist an drei Stellen nichtdeterministisch: 1, 2 und 4.

Beispiel 2 - Arithmetische Ausdrücke

- a, $a + a$ und $a \cdot a$ sind arithmetische Ausdrücke (a Symbol aus Alphabet).
- falls A_1 und A_2 arithmetische Ausdrücke sind, so sind auch $(A_1) + (A_2)$ und $(A_1) \cdot (A_2)$ arithmetische Ausdrücke.

Die Klammern sind teilweise überflüssig.

Grammatiken

Eine **Grammatik** $G = (\Sigma, V, S, R)$ besteht aus vier Komponenten:

- Endliches **Alphabet** Σ (auch Terminalalphabet genannt);
- Endliche Menge V mit $V \cap \Sigma = \emptyset$ von **Variablen** (Nichtterminale)
- **Startsymbol** $S \in V$;
- Endliche Menge von **Ableitungsregeln** R (Produktionen).

  Dabei ist eine Ableitungsregel ein Paar (ℓ, r), wobei
  - $\ell \in (V \cup \Sigma)^+$
  - $r \in (V \cup \Sigma)^*$.

  Wir schreiben oft auch $\ell \to r$.

Bemerkungen

Karlsruher Institut für Technologie

Bedeutung

- Wenn in einem Wort z das Wort ℓ Teilwort von z ist, so darf ℓ durch r in z ersetzt werden.

Notation

- Wir schreiben $w \to z$, wenn w durch Anwendung einer Ableitungsregel in z verwandelt wird
- Wir schreiben $w \overset{*}{\to} z$, wenn w durch eine Anwendung von mehreren Ableitungsregeln in z verwandelt wird.

Erzeugte Sprache

- Die von einer Grammatik G **erzeugte Sprache** $L(G)$ ist die Menge aller Wörter $z \in \Sigma^*$, für die $S \overset{*}{\to} z$ gilt.

Beispiel

Grammatik für die Menge aller arithmetischen Ausdrücke über a.

$$
\begin{aligned}
\Sigma &= \{(,), a, +, \cdot\} \\
V &= \{S\} \\
R &: \quad S \to (S) + (S) \\
&\qquad S \to (S) \cdot (S) \\
&\qquad S \to a \\
&\qquad S \to a + a \\
&\qquad S \to a \cdot a
\end{aligned}
$$

Die Chomsky Hierarchie

- **Typ–0:** Grammatiken ohne weitere Einschränkungen.
- **Typ–1/kontextsensitiv:** Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \setminus \{S\})^+$ und $|u| \leq |v|$, oder
  - $S \to \varepsilon$
- **Typ–2/kontextfrei:** Grammatiken, ausschließlich mit Ableitungsregeln der Form

$$A \to v \quad \text{mit } A \in V \text{ und } v \in (V \cup \Sigma)^*$$

- **Typ–3/rechtslinear:** Grammatiken, ausschließlich mit Ableitungsregeln der Form

$$A \to v \qquad \text{mit } A \in V \text{ und } v = \varepsilon \text{ oder } v = aB \text{ mit } a \in \Sigma, B \in V$$

Die Chomsky Hierarchie

- **Typ–1/kontextsensitiv:** Grammatiken, ausschließlich mit Ableitungsregeln der Form
  - $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \setminus \{S\})^+$ und $|u| \leq |v|$, oder
  - $S \to \varepsilon$

Bemerkung: Bei kontextsensitiven Grammatiken kann die Ableitung

$$ABC \to AXYC$$

erlaubt, aber

$$DBC \to DXYC$$

verboten sein.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ–0:** Grammatiken ohne weitere Einschränkungen.

Satz:
Falls L rekursiv aufzählbar (semi-entscheidbar) ist, so gibt es eine Chomsky–0–Grammatik mit $L(G) = L$.

Chomsky-0 Grammatiken und Semientscheidbarkeit

Karlsruher Institut für Technologie

Satz:
Falls L rekursiv aufzählbar (semi-entscheidbar) ist, so gibt es eine Chomsky–0–Grammatik mit $L(G) = L$.

Beweis:

- L rekursiv aufzählbar $\Rightarrow \exists$ DTM $\mathcal{M}$, die L akzeptiert.
- O.B.d.A. habe $\mathcal{M}$ genau einen akzeptierenden Endzustand q_J
- O.B.d.A stehen auf dem Band nur Blanks wenn q_J erreicht wird
- q_0 komme nur in der Anfangskonfiguration vor

Wir konstruieren eine Grammatik $G = (\Sigma, V = \Gamma \cup Q \cup \{S\}, S, R)$, die die Berechnung von $\mathcal{M} = (Q, \Sigma, \Gamma, s, \delta, F)$ rückwärts durchläuft.

Beweis - Beschreibung der Grammatik G

- **Rückwärtsrechnung**

  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

  $$a' q' \to qa$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

  $$q' b a' \to bqa \quad \text{für alle } b \in \Gamma$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

  $$q' a' \to qa$$

Beweis - Beschreibung der Grammatik G

- $G = (\Sigma, V, S, R)$ mit $\Sigma = \Gamma$ und $V = \{q_J\}$
- **Erzeugung der akzeptierenden Schlusskonfiguration**

$$S \to q_J, \quad q_J \to \sqcup q_J, \quad q_J \to q_J \sqcup$$

- **Rückwärtsrechnung**

  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

$$a' q' \to qa$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

$$q' b a' \to bqa \quad \text{für alle } b \in \Gamma$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

$$q' a' \to qa$$

Beweis - Beschreibung der Grammatik G

- **Rückwärtsrechnung**
  Falls $\delta(q, a) = (q', a', R)$, dann enthält G die Ableitungsregel

$$a' q' \to qa$$

  Falls $\delta(q, a) = (q', a', L)$, dann enthält G die Ableitungsregel

$$q' b a' \to bqa \quad \text{für alle } b \in \Gamma$$

  Falls $\delta(q, a) = (q', a', N)$, dann enthält G die Ableitungsregel

$$q' a' \to qa$$

- **Schlussregeln**
  Transformiere $\sqcup \sqcup \ldots \sqcup q_0 w_1 \ldots w_n \sqcup \sqcup \ldots \sqcup$ zu $w_1 \ldots w_n$

$$\sqcup q_0 \to q_0, \quad q_0 a \to a q_0, \quad q_0 \sqcup \to q_0, \quad q_0 \to \varepsilon$$

Beweis - Zusammenfassung

- Alle Wörter $w \in L$ können durch G erzeugt werden, indem die Berechnung von $\mathcal{M}$ rückwärts durchlaufen wird.
- Umgekehrt kann G nur Berechnungen von $\mathcal{M}$ rückwärts erzeugen.
- Daher ist $L(G) = L$.

Chomsky-0 Grammatiken und Semientscheidbarkeit

- **Typ–0:** Grammatiken ohne weitere Einschränkungen.

Satz:
Die von Typ–0–Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Chomsky-0 Grammatiken und Semientscheidbarkeit

Karlsruher Institut für Technologie

Satz:
Die von Typ–0–Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Beweis: Wir konstruieren zunächst eine NTM $\mathcal{M}$, die $L(G)$ akzeptiert:

1. Schreibe S auf das Band
2. Wähle eine beliebige anwendbare Ableitungsregel aus
3. Vergleiche das erzeugte Wort mit der Eingabe w
4. Bei Gleichheit wird w akzeptiert
5. Ansonsten springe zu Punkt 2

Falls $w \in L(G)$, so gibt es eine akzeptierende Berechnung.

Chomsky-0 Grammatiken und Semientscheidbarkeit

Karlsruher Institut für Technologie

Satz:
Die von Typ–0–Grammatiken G erzeugten Sprachen sind rekursiv aufzählbar.

Beweis:

Gemäß Übungsaufgabe 2, Blatt 4, kann aus der NTM eine DTM konstruiert werden, die dasselbe leistet.

Zwischenfazit

Die Klasse der rekursiv aufzählbaren Sprachen ist genau

- die Klasse der von DTMs akzeptierten Sprachen
- die Klasse der von NTMs akzeptierten Sprachen
- die Klasse der von Typ–0–Grammatiken erzeugten Sprachen

Bemerkung

- Das Wortproblem für Typ–0–Grammatiken ist unentscheidbar.
- Als Grundlage für Programmiersprachen sind die Typ–0–Grammatiken also sicherlich zu allgemein.

Chomsky–3–Grammatiken und reguläre Sprachen

- **Typ–3/rechtslinear:** Grammatiken, ausschließlich mit Ableitungsregeln der Form

$$A \to v \qquad \text{mit } A \in V \text{ und } v = \varepsilon \text{ oder } v = aB \text{ mit } a \in \Sigma, B \in V$$

Satz:
Die Klasse der von endlichen Automaten akzeptierten Sprachen ist genau die Klasse der von Chomsky–3–Grammatiken erzeugten Sprachen.

Beweis

$\Rightarrow$ Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

Beweis

$\Rightarrow$ Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Beweis

$\Rightarrow$ Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Für $w = w_1 \dots w_n$ durchlaufe $\mathcal{A}_L$ die Zustände $q_0, \dots, q_n \in Q$ mit $q_n \in F$. Dann gilt:

$$q_0 \to w_1 q_1 \to w_1 w_2 q_2 \to \dots \to w q_n \to w$$

Beweis

$\Rightarrow$ Zu einem DEA $\mathcal{A}_L = (Q, \Sigma, \delta, q_0, F)$, gibt es eine Typ-3 Grammatik G_L, die $L(\mathcal{A}_L)$ erzeugt.

G_L sei definiert durch:

- $V := Q$;
- $S := q_0$;
- R enthält die Regel
  - $q \to \varepsilon$ für alle $q \in F$
  - $q \to aq'$, falls $\delta(q, a) = q'$.

Außerdem gibt es für alle Ableitungen von G_L eine entsprechende akzeptierende Berechnung des endlichen Automatens $\mathcal{A}_L$.

Beweis

$\Leftarrow$ Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

Beweis

$\Leftarrow$ Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V \mid (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B \mid (A \to aB) \in R\}$.

Beweis

$\Leftarrow$ Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V \mid (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B \mid (A \to aB) \in R\}$.

Für $w = w_1 \dots w_n \in L$ hat die Ableitung von w mittels G_L das Aussehen:

$$S \to w_1 A_1 \to w_1 w_2 A_2 \to \dots \to w A_n \to w$$

$\mathcal{A}_L$ kann bei Eingabe $w = w_1 \dots w_n$ folgende Abarbeitung durchlaufen:

$$S \xrightarrow{w_1} A_1 \xrightarrow{w_2} A_2 \xrightarrow{w_3} \dots \xrightarrow{w_{n-1}} A_{n-1} \xrightarrow{w_n} A_n,$$

wobei $A_n \in F$, also w akzeptiert wird.

Beweis

$\Leftarrow$ Zu einer Typ-3 Grammatik G_L gibt es einen NEA $\mathcal{A}_L := (Q, \Sigma, \delta, q_0, F)$, der $L(G_L)$ akzeptiert.

- $Q := V$;
- $q_0 := S$;
- $F := \{A \in V \mid (A \to \varepsilon) \in R\}$
- $\delta(A, a) := \{B \mid (A \to aB) \in R\}$.

Außerdem gibt es für alle akzeptierenden Berechnungen von $\mathcal{A}_L$ eine entsprechende Ableitung in G_L.

Bemerkung

- Wir wissen, dass die Sprache der korrekten Klammerausdrücke nicht regulär ist.
- Typ–3–Grammatiken sind also zu einschränkend, um syntaktisch korrekte Programme zu beschreiben.

Fazit: Programmiersprachen sollten echt zwischen Typ–0 und Typ–3 angesiedelt sein.

Chomsky–1–Grammatiken bzw. kontextsensitive Sprachen

Karlsruher Institut für Technologie

Annahme:

- Die Eingabe einer Turingmaschine steht auf einem separaten Read-Only-Band.
- Der Speicherbedarf einer Turingmaschine wird nur anhand des Arbeitsbandes gemessen.

- $\mathcal{DTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **DTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.
- $\mathcal{NTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **NTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.

Chomsky–1–Grammatiken bzw. kontextsensitive Sprachen

- $\mathcal{DTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **DTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.
- $\mathcal{NTAPE}(s(n))$ ist die Klasse der Sprachen L, für die es eine **NTM** mit Platzbedarf $s(n)$ (bei Eingabelänge n) gibt, die L akzeptiert.

Offensichtlich gilt

$$\mathcal{DTAPE}(s(n)) \subseteq \mathcal{NTAPE}(s(n))$$

Außerdem gilt

$$\mathcal{NTAPE}(n) = \mathcal{NTAPE}(f(n))$$

für alle $f(n) \in \theta(n)$.

Satz

Satz:
Die Klasse der von Chomsky–1–Grammatiken erzeugten Sprachen stimmt mit der Klasse $\mathcal{NTAPE}(n)$ überein.

Ohne Beweis.

Bemerkung

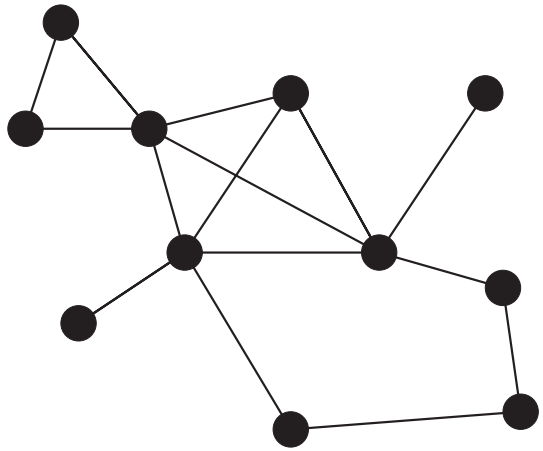
Es ist offen, ob $\mathcal{NTAPE}(n) = \mathcal{DTAPE}(n)$ ist.

Wiederholung: Das Problem CLIQUE

Eine **Clique** in einem Graphen $G = (V, E)$ ist eine Menge $V' \subseteq V$ so, dass für alle $i, j \in V'$, $i \neq j$, gilt: $\{i, j\} \in E$.

Problem CLIQUE

Gegeben: Graph $G = (V, E)$ und ein Parameter $K \leq |V|$
Frage: Gibt es in G eine Clique der Größe mindestens K?



Satz

Satz:
Das Cliquen-Problem gehört zu $\mathcal{DTAPE}(n)$.

- Gegeben sei $G = (V, E)$ mit $V = \{1, \ldots, n\}$ und $1 \leq K \leq n$.
- Benutze n-Vektor $c \in \{0, 1\}^n$ mit der Bedeutung für eine Menge C

$$c_i = 1 \quad \Longleftrightarrow i \in C$$

- Teste für jeden Vektor $c \in \{0, 1\}^n$, ob die zugehörige Menge $C \subseteq V$ eine Clique der Größe K in G ist:
  - Zähle die Einsen in c.
  - Überprüfe für jedes Paar c_i, c_j mit $c_i = c_j = 1$ ob $\{i, j\} \in E$.

Satz

Satz:
Das Cliquen-Problem gehört zu $\mathcal{DTAPE}(n)$.

Alle Vektoren $c \in \{0,1\}^n$ können nacheinander, beginnend mit $(0,0,\ldots,0)$, durchgetestet werden, wobei:

- nach einem positiven Test (G,K) akzeptiert wird;
- nach einem negativen Test der Vektor durch seinen lexikalischen Nachfolger überschrieben wird.

Dazu wird insgesamt nur linearer Speicherplatz benötigt.

Bemerkung 1

Falls $\mathcal{P} \neq \mathcal{NP}$, kann das Wortproblem für kontextsensitive Grammatiken im allgemeinen nicht in polynomialer Zeit entschieden werden.

Bemerkung 2

Für jede Chomsky–1–Grammatik gibt es eine äquivalente Chomsky–1–Grammatik, bei der alle Regeln folgende Form haben:

- $A \to C$
- $A \to CD$
- $AB \to CD$
- $A \to a$
- $S \to \varepsilon$

wobei jeweils $A, B \in V$, $C, D \in V \setminus \{S\}$ und $a \in \Sigma$.

Notation

Statt Regeln

$$S \to \alpha \text{ und } S \to \beta$$

schreiben wir abkürzend

$$S \to \alpha \mid \beta$$

Typ–2 / Kontextfreie Grammatiken

Karlsruher Institut für Technologie

Grammatiken, ausschließlich mit Ableitungsregeln der Form

$$A \to v \quad \text{mit } A \in V \text{ und } v \in (V \cup \Sigma)^*$$

Typ–2 Grammatiken: Beispiel 1

Die Sprache

$$L = \{0^n 1^n \mid n \geq 1\}$$

Typ–2 Grammatiken: Beispiel 1

Die Sprache

$$L = \{0^n 1^n \mid n \geq 1\}$$

wird erzeugt durch die Grammatik

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 01 \mid 0S1\} \,. \end{aligned}$$

Typ–2 Grammatiken: Beispiel 2

Sei $L = \{w \in \{0,1\}^* \mid w = w^R\}$ die Sprache der Palindrome über $\{0,1\}$.

Es gilt:

1. $0, 1, \varepsilon$ sind Palindrome
2. falls w Palindrom, so auch $0w0$ und $1w1$
3. alle Palindrome lassen sich durch endliche viele Anwendungen von (1.) und (2.) erzeugen

Typ–2 Grammatiken: Beispiel 2

Sei $L = \{w \in \{0,1\}^* \mid w = w^R\}$ die Sprache der Palindrome über $\{0,1\}$.

Es gilt:

1. $0, 1, \varepsilon$ sind Palindrome
2. falls w Palindrom, so auch $0w0$ und $1w1$
3. alle Palindrome lassen sich durch endliche viele Anwendungen von (1.) und (2.) erzeugen

Zugehörige kontextfreie Grammatik:

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0,1\} \\ R &= \{S \to \varepsilon \mid 0 \mid 1, \\ &\qquad S \to 0S0 \mid 1S1\} \end{aligned}$$

Typ–2 Grammatiken: Beispiel 3

Sei L die Sprache aller $w \in \{0,1\}^+$ bei denen die Anzahl der Nullen gleich der Anzahl der Einsen ist.

Typ–2 Grammatiken: Beispiel 3

Sei L die Sprache aller $w \in \{0,1\}^+$ bei denen die Anzahl der Nullen gleich der Anzahl der Einsen ist.

Setze

$$\begin{aligned} V &= \{S, A, B\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 0B \mid 1A, \\ &\quad A \to 0 \mid 0S \mid 1AA, \\ &\quad B \to 1 \mid 1S \mid 0BB\} \end{aligned}$$

Durch Induktion über die Länge der durch G erzeugten Wörter lässt sich beweisen, dass $L = L(G)$.

Syntaxbäume

Syntaxbäume visualisieren die Ableitung eines einzelnen Wortes.

- An der Wurzel eines Syntaxbaumes steht das Startsymbol.
- Jeder innere Knoten enthält eine Variable.
- Die Blätter sind Symbole aus Σ oder ε.
- Wenn ein innerer Knoten A als Nachfolger von links nach rechts $\alpha_1, \ldots, \alpha_r \in V \cup \Sigma$ hat, so muss $A \to \alpha_1 \ldots \alpha_r$ eine Ableitungsregel der Grammatik sein.

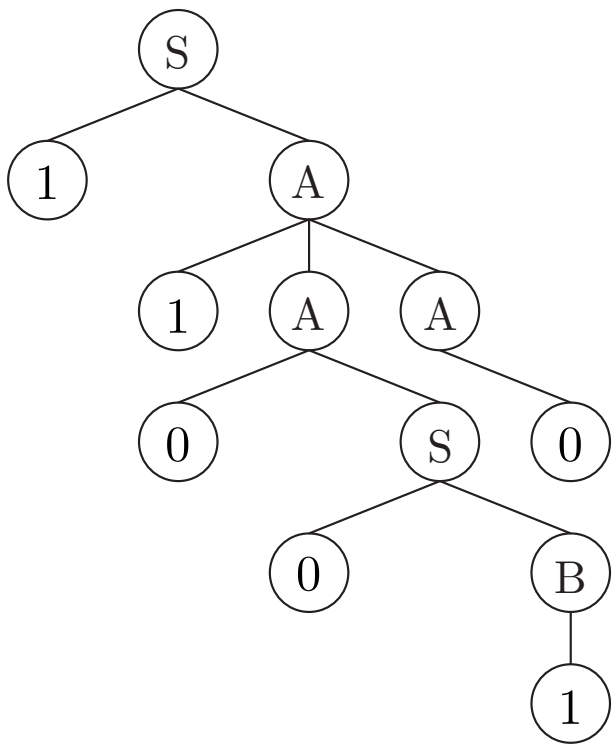
Syntaxbäume - Beispiel

Zu den Regeln

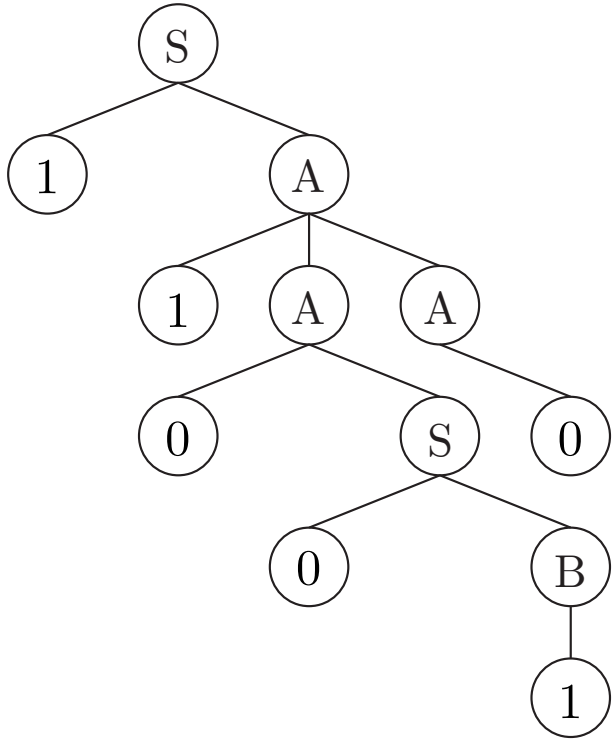
$$R = \{S \to 0B \mid 1A, A \to 0 \mid 0S \mid 1AA, B \to 1 \mid 1S \mid 0BB\}$$

betrachte die Ableitung

$$S \to 1A \to 11AA \to 11A0 \to 110S0 \to 1100B0 \to 110010$$



Syntaxbäume - Beispiel



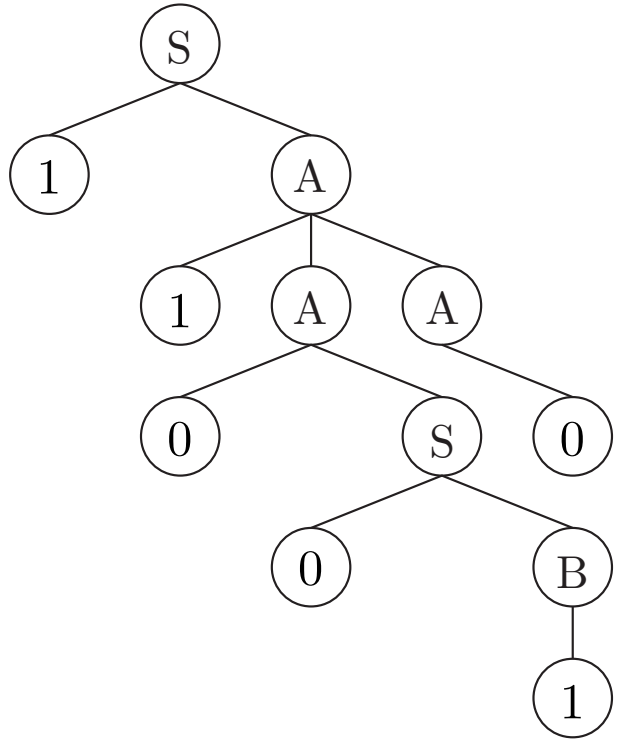
- Zu jeder Ableitung gehört genau ein Syntaxbaum
- Zu jedem Syntaxbaum gehören jedoch verschiedene Ableitungen des gleichen Wortes.

Syntaxbäume - Beispiel

$$R = \{S \to 0B \mid 1A, A \to 0 \mid 0S \mid 1AA, B \to 1 \mid 1S \mid 0BB\}$$

$$S \to 1A \to 11AA \to 110SA \to 1100BA \to 11001A \to 110010$$

$$S \to 1A \to 11AA \to 11A0 \to 110S0 \to 1100B0 \to 110010$$



Links/Rechtsableitung, Eindeutigkeit

Für Chomsky-2 Grammatiken gilt:

- Wegen der Kontextfreiheit ist die Reihenfolge, in der abgeleitet wird, für das Ergebnis unerheblich.

Eine **Linksableitung** (**Rechtsableitung**) ist eine Ableitung, bei der in jedem Schritt die linkeste (rechteste) Variable abgeleitet wird.

Eine kontextfreie Grammatik G heißt **eindeutig**, wenn es für jedes Wort $w \in L(G)$ genau einen Syntaxbaum gibt.

Eine kontextfreie Sprache L heißt **eindeutig**, wenn es eine eindeutige Grammatik G mit $L(G) = L$ gibt. Ansonsten heißt L **inhärent mehrdeutig**.

Beispiel

Die Sprache

$$L = \{0^n 1^n \mid n \geq 1\}$$

erzeugt durch die Grammatik

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 01 \mid 0S1\} \,. \end{aligned}$$

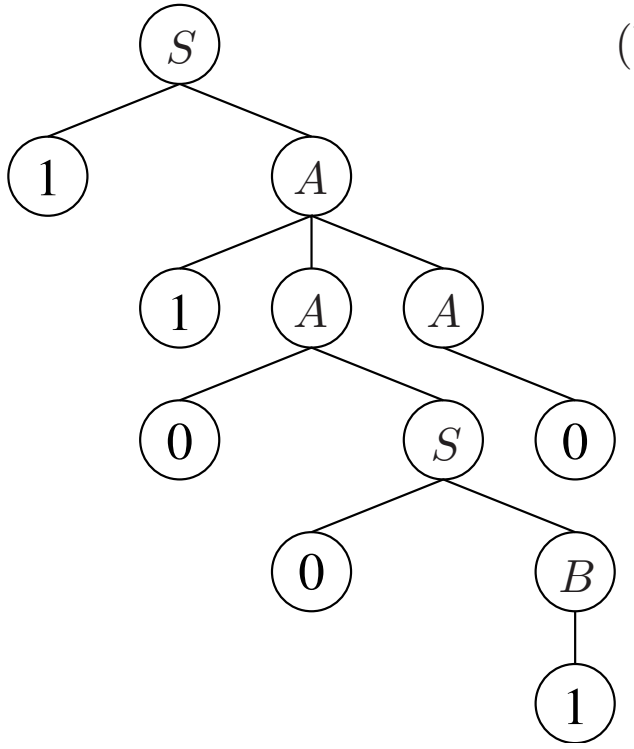
ist eindeutig.

Die Sprache gegeben durch die Regeln

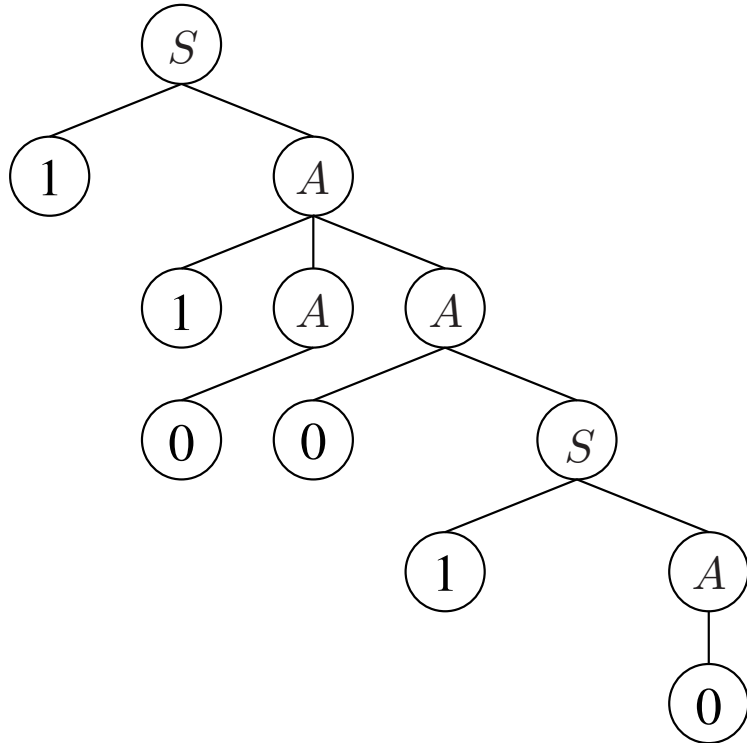
$$R = \{S \to 0B \mid 1A, A \to 0 \mid 0S \mid 1AA, B \to 1 \mid 1S \mid 0BB\}$$

ist nicht eindeutig.

(a)



(b)



Chomsky-Normalform

Karlsruher Institut für Technologie

Eine kontextfreie Grammatik ist in **Chomsky–Normalform**, wenn alle Regeln von der Form:

$$A \to BC \qquad \text{oder} \qquad A \to a$$

sind, mit $A, B, C \in V$ und $a \in \Sigma$.

$$A \to BC \qquad \text{oder} \qquad A \to a$$

- Grammatiken in Chomsky–Normalform können also nicht das Wort ε erzeugen.
- Für kontextfreie Sprachen, die ε enthalten, läßt sich eine Grammatik leicht ergänzen durch die Regeln

$$S' \to \varepsilon \qquad \text{und} \qquad S' \to S$$

wobei S' ein neues Startsymbol zur Erzeugung von ε ist.

Satz:
Jede kontextfreie Grammatik, die nicht das leere Wort erzeugt, kann in eine Grammatik in Chomsky–Normalform überführt werden.

Beweis (konstruktiv):

- Wir geben eine Schritt–für–Schritt–Überführung der Regeln in Regeln in Normalform an.
- Großbuchstaben repräsentieren immer Nichtterminale
- Kleinbuchstaben repräsentieren immer Terminale

Wir veranschaulichen den Beweis an folgendem Beispiel:
Grammatik $G = (\Sigma, V, S, R)$ mit

$$\begin{aligned} \Sigma &= \{a, b\} \\ V &= \{A, B, C, D, E, S\} \end{aligned}$$

und der folgenden Regelmenge R:

$$\begin{aligned} S &\to A \mid aAa \mid bBb \mid \varepsilon \\ A &\to C \mid a \\ B &\to b \\ C &\to CDE \mid \varepsilon \\ D &\to A \mid B \mid ab \\ E &\to B \end{aligned}$$

Schritt 1:

Karlsruher Institut für Technologie

Ziel:

- Alle Regeln enthalten auf der rechten Seite nur Symbole aus V oder nur ein Symbol aus Σ.

Vorgehen:

- Ersetze dazu in allen rechten Seiten von Regeln Symbole aus $a \in \Sigma$ durch neue Variablen Y_a und füge die Regeln $Y_a \to a$ hinzu.

Schritt 1

$$
\begin{aligned}
S &\to A \mid aAa \mid bBb \mid \varepsilon \\
A &\to C \mid a \\
B &\to b \\
C &\to CDE \mid \varepsilon \\
D &\to A \mid B \mid ab \\
E &\to B
\end{aligned}
$$

$$
\begin{aligned}
S &\to A \mid Y_aAY_a \mid Y_bBY_b \mid \varepsilon \\
A &\to C \mid a \\
B &\to b \\
C &\to CDE \mid \varepsilon \\
D &\to A \mid B \mid Y_aY_b \\
E &\to B \\
Y_a &\to a \\
Y_b &\to b
\end{aligned}
$$

Schritt 2

Ziel:

- Alle rechten Seiten haben Länge ≤ 2.

Vorgehen:

- Sei $A \to B_1 \dots B_m$ Regel mit $m > 2$.
- Führe $m-2$ neue Variablen $C_1, \dots, C_{m-2}$ ein, und ersetze die Regel

$$A \to B_1 \dots B_m$$

durch neue Regeln

$$\begin{aligned} A &\to B_1 C_1 \\ C_i &\to B_{i+1} C_{i+1} \qquad \text{für } 1 \leq i \leq m-3 \\ C_{m-2} &\to B_{m-1} B_m \end{aligned}$$

Schritt 2

$$
\begin{aligned}
S &\to A \mid Y_aAY_a \mid Y_bBY_b \mid \varepsilon \\
A &\to C \mid a \\
B &\to b \\
C &\to CDE \mid \varepsilon \\
D &\to A \mid B \mid Y_aY_b \\
E &\to B \\
Y_a &\to Y_a \\
Y_b &\to Y_b
\end{aligned}
$$

$$
\begin{aligned}
S &\to A \mid Y_aC_1 \mid Y_bC_2 \mid \varepsilon \\
A &\to C \mid a \\
B &\to b \\
C &\to CC_3 \mid \varepsilon \\
D &\to A \mid B \mid Y_aY_b \\
E &\to B \\
C_1 &\to AY_a \\
C_2 &\to BY_b \\
C_3 &\to DE \\
Y_a &\to a \\
Y_b &\to b
\end{aligned}
$$

Schritt 3:

Karlsruher Institut für Technologie

Ziel:

- Es kommen keine Regeln $A \to \varepsilon$ vor.

Vorgehen, Phase 1:
Finde die Menge V' aller Variablen A für die $A \overset{*}{\to} \varepsilon$ existiert:

- Es werden erst alle A mit $A \to \varepsilon$ in V' aufgenommen.
- Dann wird geprüft, ob neue Regeln $B \to \varepsilon$ entstehen, wenn man A in allen Regeln auf der rechten Seite A durch ε ersetzt.
- Ist dies der Fall, so werden die entsprechenden Variablen B in V' aufgenommen und genauso behandelt.
- Das Verfahren hört auf, wenn V' sich nicht mehr ändert

Schritt 3:

Karlsruher Institut für Technologie

Ziel:

- Es kommen keine Regeln $A \to \varepsilon$ vor.

Vorgehen, Phase 1:
Finde die Menge V' aller Variablen A für die $A \overset{*}{\to} \varepsilon$ existiert:

- Bemerkung: In Phase 1 werden noch keine Regeln geändert
- Die Ersetzung ist also nur „testweise“
- Am Ende enthält V' alle Variablen A mit $A \overset{*}{\to} \varepsilon$.

Schritt 3:

Karlsruher Institut für Technologie

Ziel:

- Es kommen keine Regeln $A \to \varepsilon$ vor.

Vorgehen, Phase 2: Ersetzung.

- Gegeben V' aus Phase 1
- Streiche alle Regeln $A \to \varepsilon$
- Für $A \to BC$ füge die zusätzliche Regel
  - $A \to B$ falls $C \in V'$
  - $A \to C$ falls $B \in V'$

  ein.
- (Die Regel $A \to BC$ wird nicht gestrichen).

- Initialisierung
  $V' = \{A \mid A \to \varepsilon\} = \{S, C\}$
- Erster Durchlauf liefert: A
  $\Rightarrow V' = \{A, C, S\}$
- Zweiter Durchlauf liefert: D
  $\Rightarrow V' = \{A, C, D, S\}$
- Dritter Durchlauf liefert nichts neues

$$
\begin{aligned}
S &\to A \mid Y_a C_1 \mid Y_b C_2 \mid \varepsilon \\
A &\to C \mid a \\
B &\to b \\
C &\to CC_3 \mid \varepsilon \\
D &\to A \mid B \mid Y_a Y_b \\
E &\to B \\
C_1 &\to AY_a \\
C_2 &\to BY_b \\
C_3 &\to DE \\
Y_a &\to a \\
Y_b &\to b
\end{aligned}
$$

- Streiche ε-Produktionen
- Simuliere Ableitungen $A \overset{*}{\rightarrow} \varepsilon$ auf den verbleibenden Regeln $V' = \{A, C, D, S\}$

$$
\begin{aligned}
S &\rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
A &\rightarrow C \mid a \\
B &\rightarrow b \\
C &\rightarrow CC_3 \mid C_3 \\
D &\rightarrow A \mid B \mid Y_a Y_b \\
E &\rightarrow B \\
C_1 &\rightarrow AY_a \mid Y_a \\
C_2 &\rightarrow BY_b \\
C_3 &\rightarrow DE \mid E \\
Y_a &\rightarrow a \\
Y_b &\rightarrow b
\end{aligned}
$$

Schritt 4.

Ziel

- Die Grammatik enthält keine (Ketten-)Regeln der Form $A \to B$.
- Beispiel

$$\begin{aligned} A &\to B \mid C \\ B &\to C \\ C &\to c \end{aligned}$$

Schritt 4.

Ziel

- Die Grammatik enthält keine (Ketten-)Regeln der Form $A \to B$.

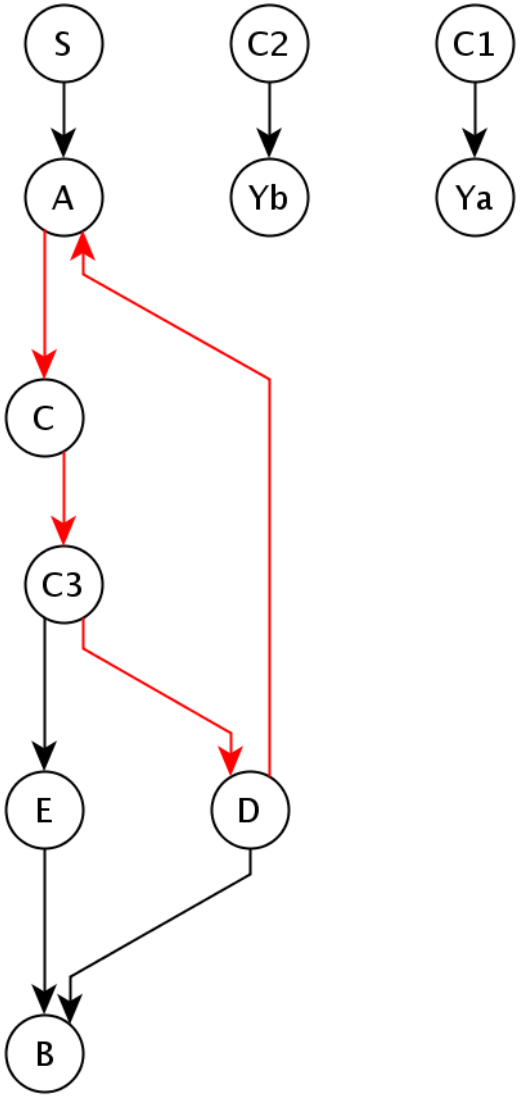
Vorgehen

- **Phase 1:** Finde alle Kreise $A_1 \to A_2 \to \ldots \to A_n \to A_1$
  Ersetze alle A_i durch A_1
- **Phase 2:** Betrachte die Regeln der Form $A \to B$ in umgekehrt topologischer Reihenfolge
  - Für Regel $A \to B$ und jede Regel $B \to b$ füge Regel $A \to b$ hinzu
  - Lösche Regel $A \to B$

Topologische Sortierung der Regelmenge

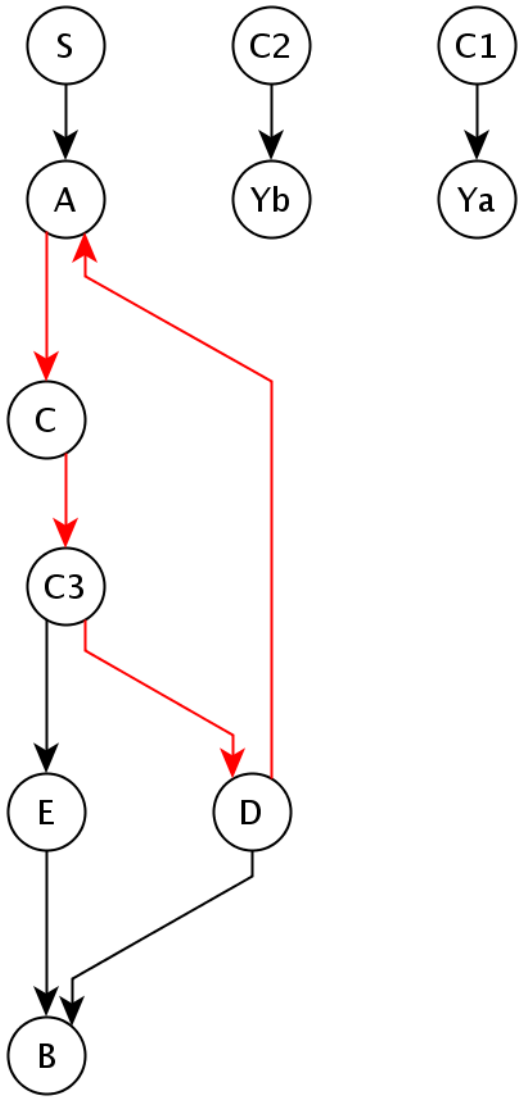
- $V_1, \ldots, V_k$ Menge von Variablen aus Kettenregeln
- $V_1, \ldots, V_k$ Topologisch sortiert, wenn gilt:
- $V_i \overset{*}{\rightarrow} V_j \Rightarrow i < j$
- Voraussetzung: Es gibt keine zyklischen Abhängigkeiten

Abhängigkeitsgraph



$$
\begin{aligned}
S &\rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
A &\rightarrow C \mid a \\
B &\rightarrow b \\
C &\rightarrow CC_3 \mid C_3 \\
D &\rightarrow A \mid B \mid Y_a Y_b \\
E &\rightarrow B \\
C_1 &\rightarrow AY_a \mid Y_a \\
C_2 &\rightarrow BY_b \\
C_3 &\rightarrow DE \mid E \\
Y_a &\rightarrow a \\
Y_b &\rightarrow b
\end{aligned}
$$

Schritt 4 - Phase 1

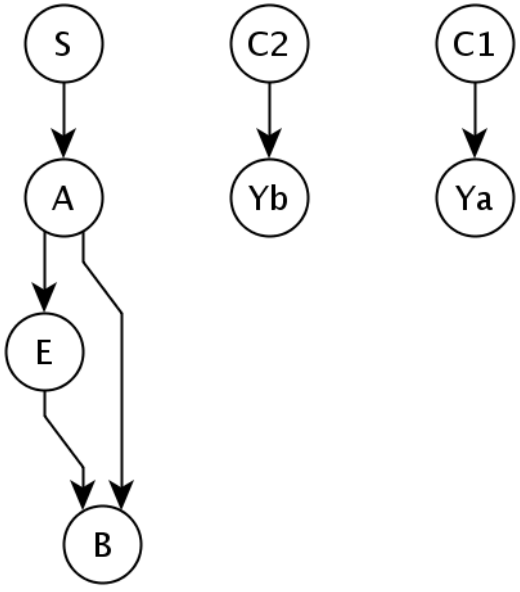


- Zykus $A \to C \to C_3 \to D \to A$
- $\Rightarrow$ A, C, $C3$, D äquivalent
- Entferne an Zyklus beteiligte Regeln
- Ersetze Vorkommen von C, C_3, D in allen Regeln durch A
- Lösche Regeln der Form $A \to A$

Zyklus: $A \to C \to C_3 \to D \to A$

$$
\begin{array}{rcl}
S & \to & A \mid Y_a C_1 \mid Y_b C_2 \\
A & \to & C \mid a \\
B & \to & b \\
C & \to & CC_3 \mid C_3 \\
D & \to & A \mid B \mid Y_a Y_b \\
E & \to & B \\
C_1 & \to & AY_a \mid Y_a \\
C_2 & \to & BY_b \\
C_3 & \to & DE \mid E \\
Y_a & \to & a \\
Y_b & \to & b
\end{array}
\qquad
\begin{array}{rcl}
S & \to & A \mid Y_a C_1 \mid Y_b C_2 \\
A & \to & a \\
B & \to & b \\
A & \to & AA \\
A & \to & B \mid Y_a Y_b \\
E & \to & B \\
C_1 & \to & AY_a \mid Y_a \\
C_2 & \to & BY_b \mid Y_b \\
A & \to & AE \mid E \\
Y_a & \to & a \\
Y_b & \to & b
\end{array}
$$

Schritt 4 - Phase 2



$$
\begin{aligned}
S &\rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
A &\rightarrow a \mid AA \mid B \mid Y_a Y_b \mid AE \mid E \\
B &\rightarrow b \\
E &\rightarrow B \\
C_1 &\rightarrow AY_a \mid Y_a \\
C_2 &\rightarrow BY_b \mid Y_b \\
Y_a &\rightarrow a \\
Y_b &\rightarrow b
\end{aligned}
$$

Topologische Sortierung: $S, A, E, B, C_2, Y_b, C_1, Y_a$

- Gehe in umgekehrter topologischer Sortierung vor
- Ersetze $A \to B$ durch $A \to \beta$, falls Regel $B \to \beta$ existiert
- Topologische Sortierung: $S, A, E, B, C_2, Y_b, C_1, Y_a$

$$
\begin{array}{rcl}
S & \to & A \mid Y_a C_1 \mid Y_b C_2 \\
A & \to & a \mid AA \mid B \mid Y_a Y_b \mid AE \mid E \\
B & \to & b \\
E & \to & B \\
C_1 & \to & AY_a \mid Y_a \\
C_2 & \to & BY_b \mid Y_b \\
Y_a & \to & a \\
Y_b & \to & b
\end{array}
\qquad
\begin{array}{rcl}
S & \to & Y_a C_1 \mid Y_b C_2 \\
S & \to & a \mid AA \mid b \mid Y_a Y_b \mid AE \mid b \\
B & \to & b \\
E & \to & b \\
C_1 & \to & AY_a \mid a \\
C_2 & \to & BY_b \mid b \\
Y_a & \to & a \\
Y_b & \to & b
\end{array}
$$

Sonderbehandlung von ε-Produktionen

- Grammatik ist nun in Chomsky-Normalform, aber
- G enthält Regel $S \to \varepsilon$ (haben wir entfernt)
- Erweitere Grammatik um Regeln $S' \to S$ und $S' \to \varepsilon$ für neues Startsymbol S'

Der CYK-Algorithmus

Satz:
Es gibt einen Algorithmus (den Cocke–Younger–Kasami Algorithmus), der für eine kontextfreie Grammatik G in Chomsky–Normalform und ein Wort $w \in \Sigma^*$ in Zeit $\mathcal{O}(|R| \cdot n^3)$ entscheidet, ob $w \in L(G)$, wobei $n = |w|$ und $|R|$ die Anzahl der Regeln von G ist.

Beweis - Beschreibung des CYK-Algorithmus

- Sei $w = w_1 \dots w_n$.
- Sei $V_{ij} \subseteq V$ so dass $A \overset{*}{\to} w_i \dots w_j$ impliziert $A \in V_{ij}$.

- Für alle $1 \leq i \leq j \leq n$ berechne die Menge $V_{ij} \subseteq V$.
- Dann ist $w \in L(G)$ genau dann, wenn $S \in V_{1n}$ ist.

- Die Tabelle der V_{ij} wird nach wachsendem $\ell := j - i$ aufgebaut, beginnend mit $\ell = 0$.
- Für $j - i = \ell > 0$ wird die Berechnung von V_{ij} systematisch auf zuvor berechnete V_{ik}, $V_{k+1\,j}$ mit $i \leq k < j$ zurückgeführt

Bemerkung

- Wir benutzen dynamische Programmierung.

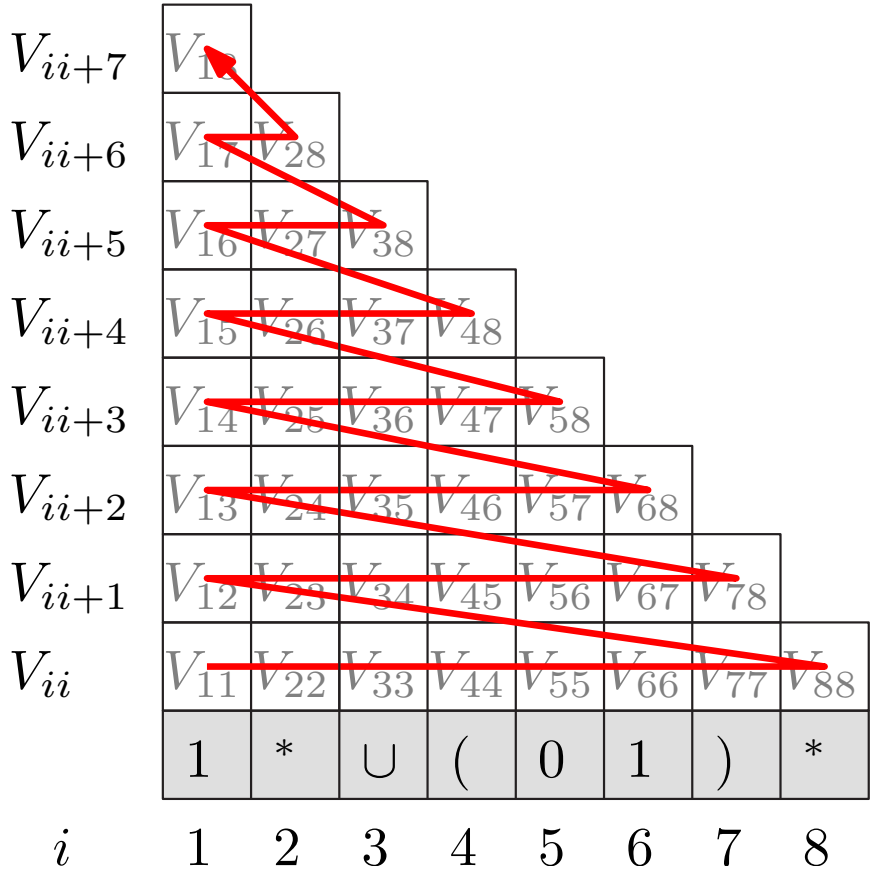
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	V_{11}	V_{22}	V_{33}	V_{44}	V_{55}	V_{66}	V_{77}	V_{88}
	1	*	$\cup$	(	0	1	)	*
i	1	2	3	4	5	6	7	8

$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \rightarrow & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \rightarrow & Y_\cup S \\
C_2 & \rightarrow & SY_) \\
Y_* & \rightarrow & * \\
Y_\cup & \rightarrow & \cup \\
Y_(& \rightarrow & (\\
Y_) & \rightarrow &)
\end{array}
$$



Fall $\ell = 0$:

- Konstruiere die Mengen V_{ii}, d.h. alle $A \in V$ mit $A \overset{*}{\to} w_i$.
- Da G in Chomsky–Normalform ist, gilt $A \overset{*}{\to} w_i$ nur, wenn $(A \to w_i) \in R$.
- Die Berechnung von V_{ii} ist für alle $i \in \{1, \ldots n\}$ in $\mathcal{O}(|R|)$ möglich.

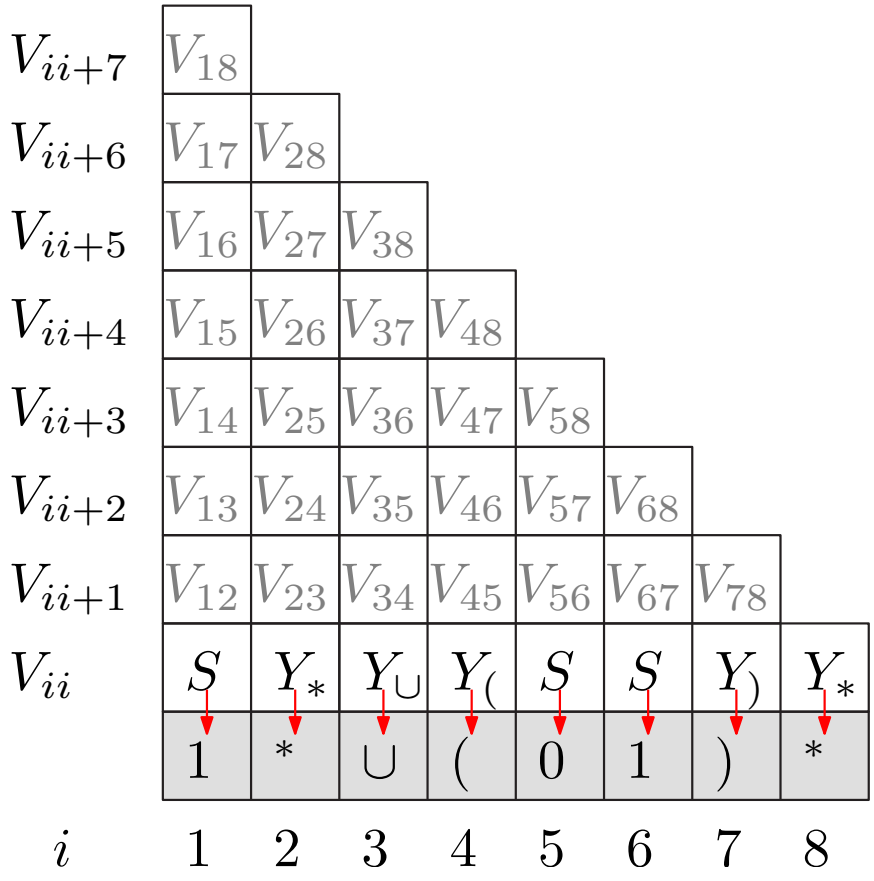
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	V_{11}	V_{22}	V_{33}	V_{44}	V_{55}	V_{66}	V_{77}	V_{88}
	1	*	$\cup$	(	0	1	)	*
i	1	2	3	4	5	6	7	8

$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



Fall $\ell > 0$:

- Jede Ableitung von $w_i \dots w_j$ muss mit einer Regel der Form

$$A \to BC$$

beginnen, wobei ein $k \in \{i, \dots, j-1\}$ existiert mit

- $B \overset{*}{\to} w_i \dots w_k$ und
- $C \overset{*}{\to} w_{k+1} \dots w_j$.

Verfahren

- Speichere alle Mengen V_{rs} als Arrays der Länge $|V|$, in denen für jedes $A \in V$ markiert ist, ob $A \in V_{rs}$.
- **Berechnung von V_{ij}:**
  Überprüfe für jede Regel $(A \to BC) \in R$ und jedes k, ob

$$B \overset{*}{\to} w_i \dots w_k$$
$$C \overset{*}{\to} w_{k+1} \dots w_j$$

  durch Ansehen der Stelle
  - B im Array zu V_{ik} und
  - C im Array zu $V_{k+1\ j}$.

Verfahren

- Speichere alle Mengen V_{rs} als Arrays der Länge $|V|$, in denen für jedes $A \in V$ markiert ist, ob $A \in V_{rs}$.

- **Berechnung von V_{ij}:**
  Überprüfe für jede Regel $(A \to BC) \in R$ und jedes k, ob

$$B \overset{*}{\to} w_i \dots w_k$$
$$C \overset{*}{\to} w_{k+1} \dots w_j$$

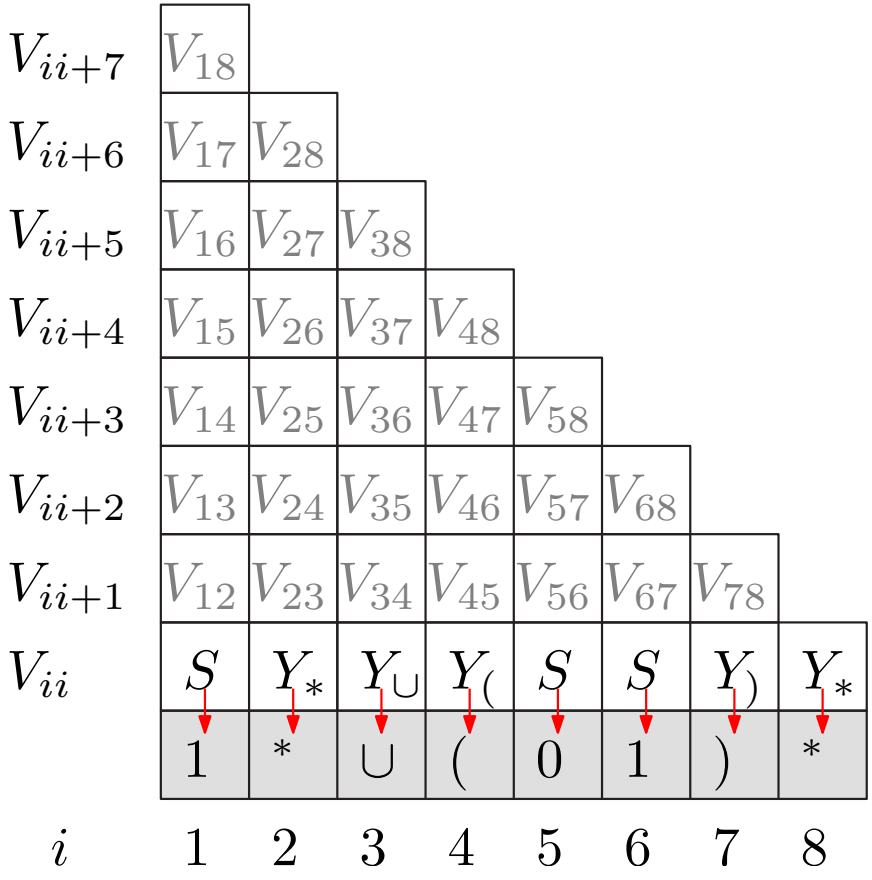
  durch Ansehen der Stelle
  - B im Array zu V_{ik} und
  - C im Array zu $V_{k+1\ j}$.

Bemerkung

- Dies benötigt Aufwand in $\mathcal{O}(n \cdot |R|)$

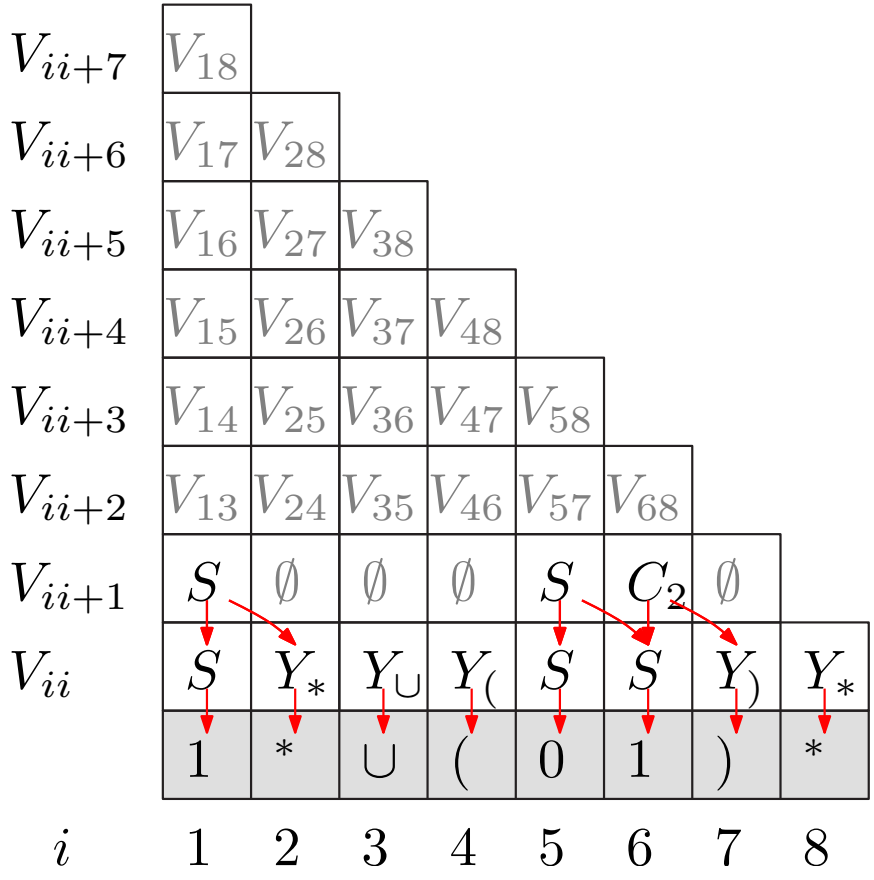
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



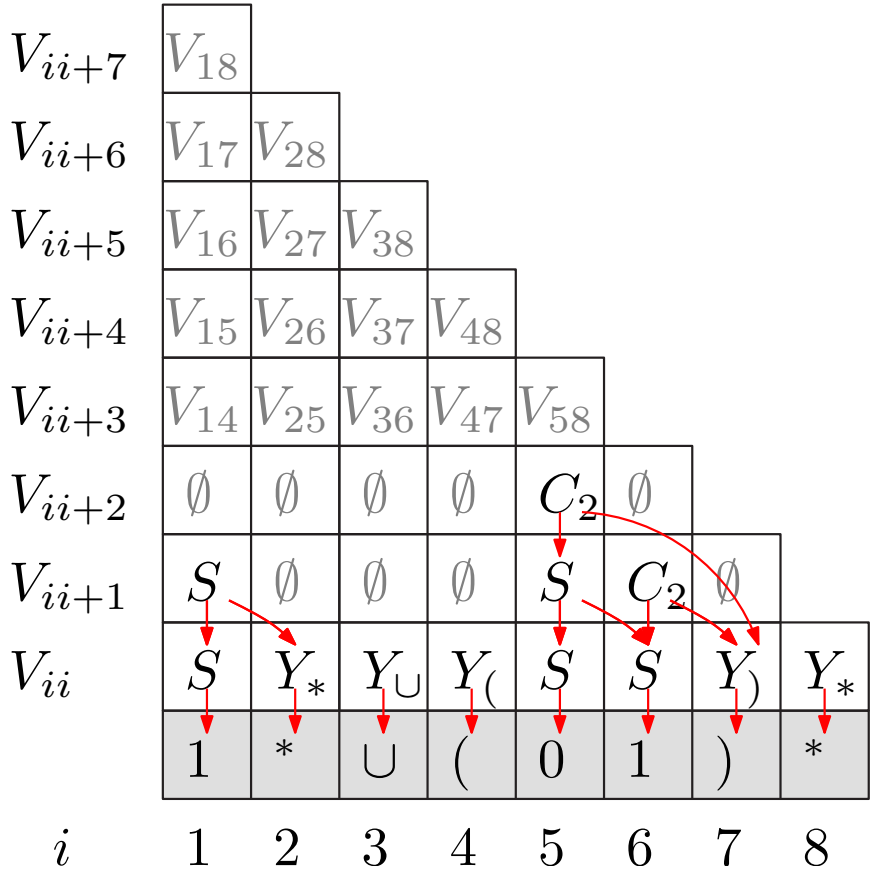
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



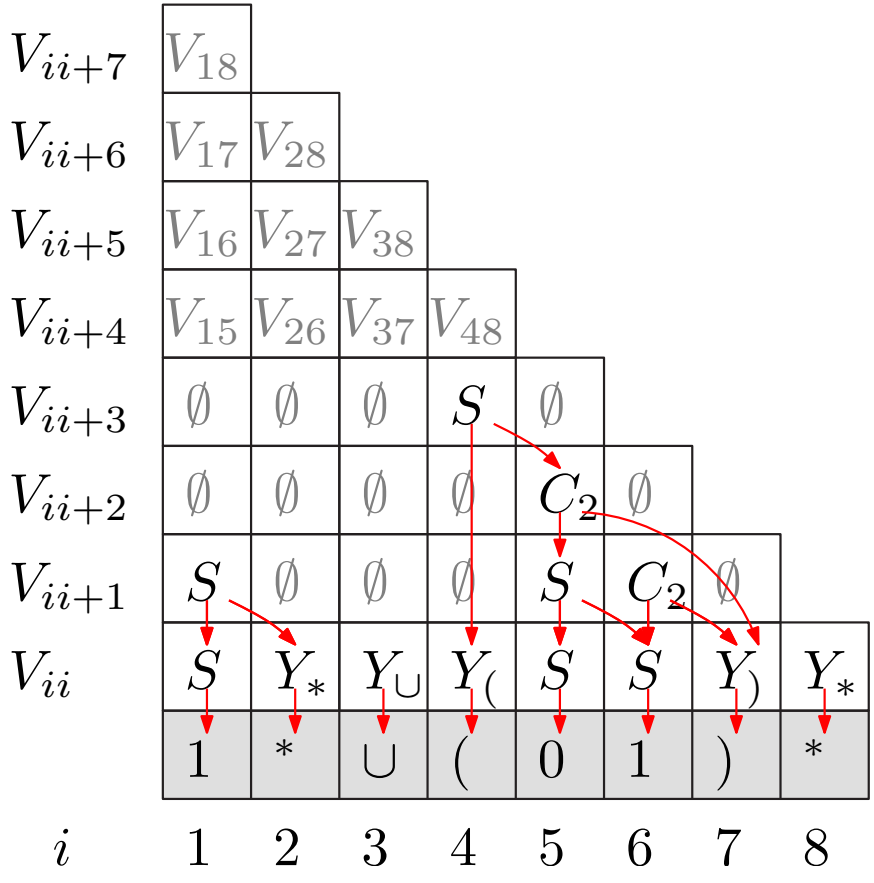
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



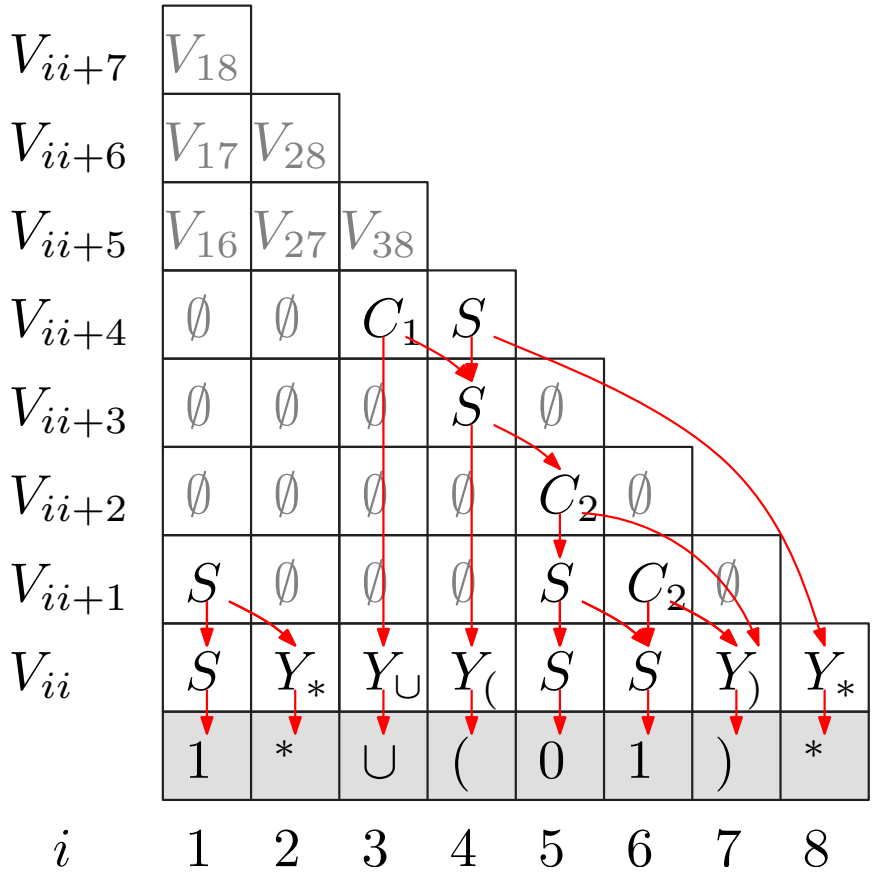
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



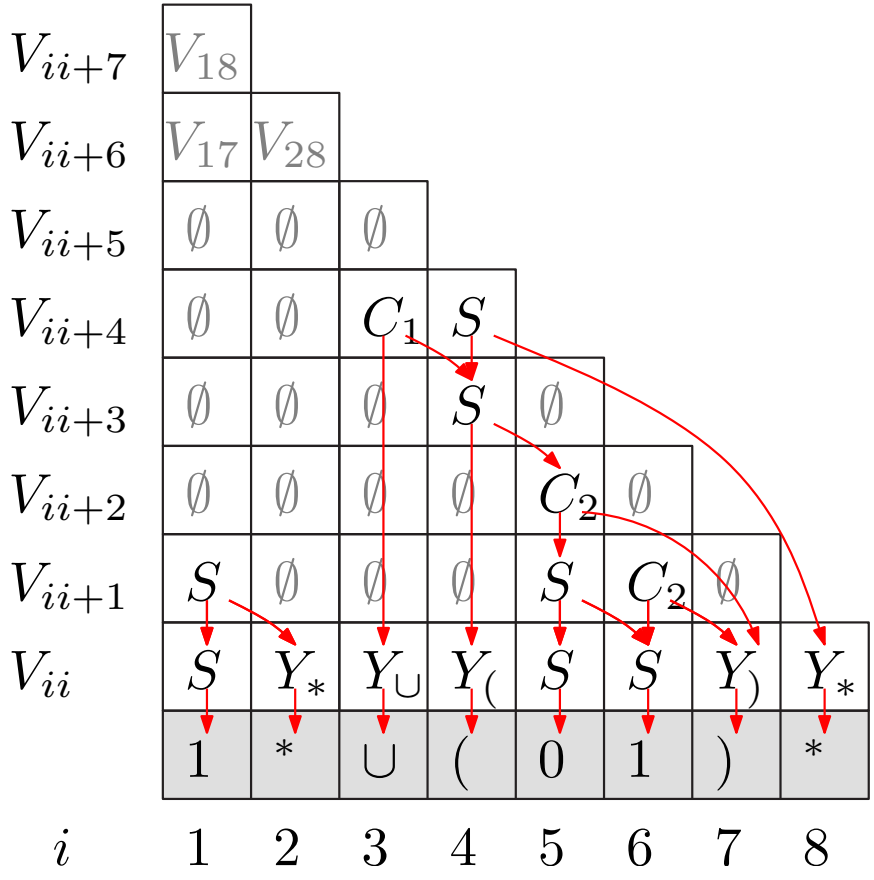
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



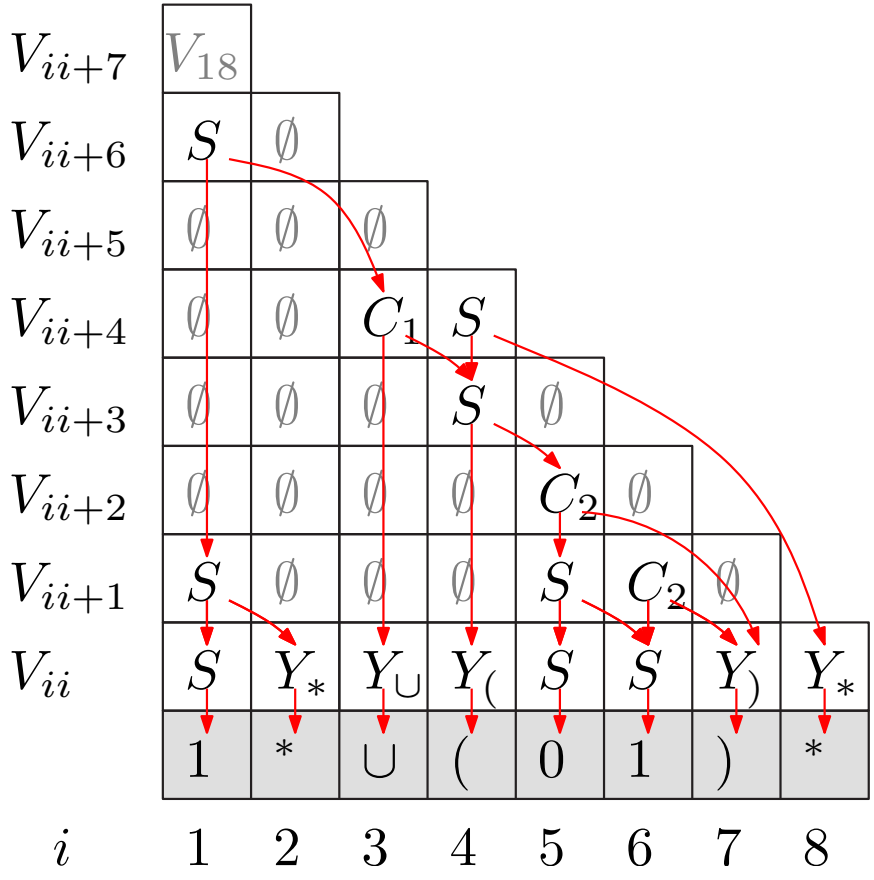
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



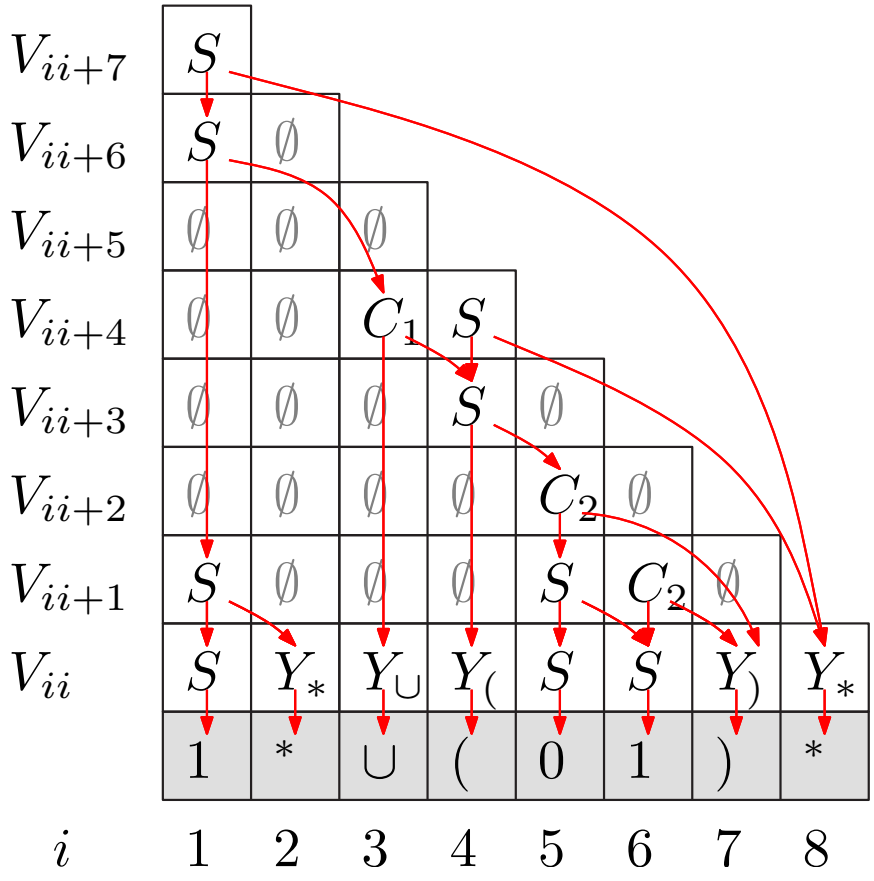
$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



$w = 1^* \cup (01)^*$

$$
\begin{array}{rcl}
S & \to & SY_* \mid SC_1 \mid \\
 & & SS \mid Y_(C_2 \mid \\
 & & e \mid \emptyset \mid 0 \mid 1 \\
C_1 & \to & Y_\cup S \\
C_2 & \to & SY_) \\
Y_* & \to & * \\
Y_\cup & \to & \cup \\
Y_(& \to & (\\
Y_) & \to &)
\end{array}
$$



Ergebnisse zum Wortproblem

- **Typ-0 Grammatik.** Das Wortproblem ist nicht entscheidbar.
- **Typ-1 Grammatik.** Das Wortproblem ist NP-vollständig.
- **Typ-2 Grammatik.** Das Wortproblem ist in polynomieller Zeit lösbar.
- **Typ-3 Grammatik.** Das Wortproblem ist in linearer Zeit lösbar.

Das Pumping-Lemma für kontextfreie Sprachen

Pumping-Lemma für kontextfreie Sprachen

Für jede kontextfreie Sprache L
gibt es eine Konstante $n \in \mathbb{N}$,
so dass sich jedes Wort $z \in L$ mit $|z| \geq n$

so als

- $z = uvwxy$

schreiben lässt, dass

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- für alle $i \geq 0$ das Wort $uv^i wx^i y \in L$ ist.

Ogden's Lemma für kontextfreie Sprachen

Ogden's Lemma für kontextfreie Sprachen

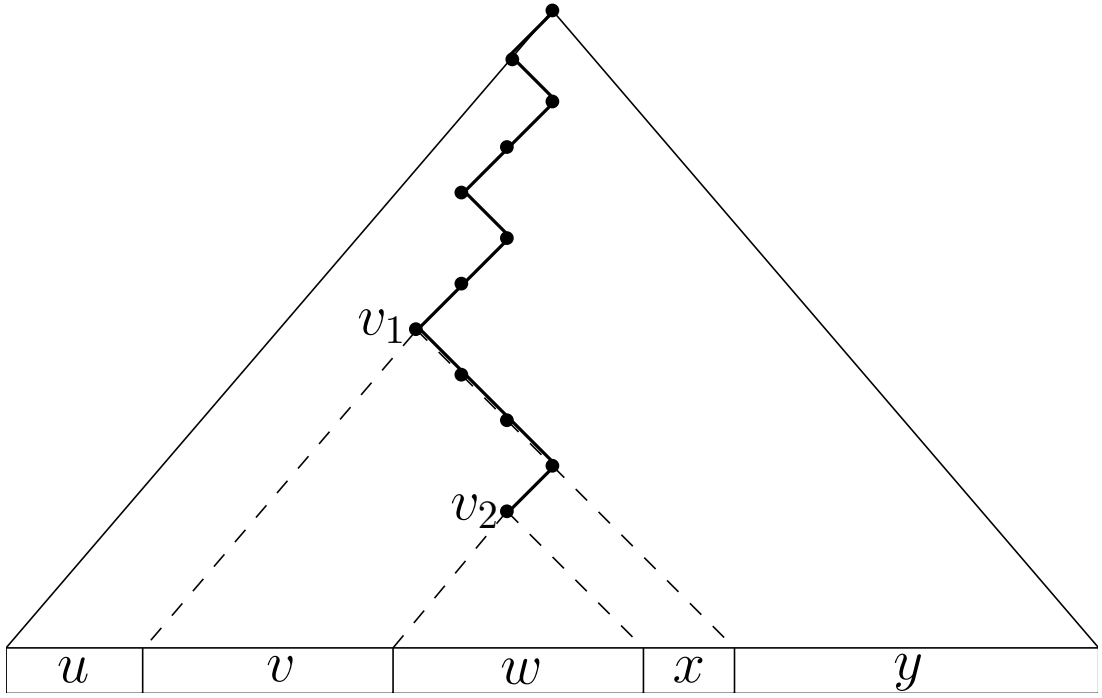
Für jede kontextfreie Sprache L
gibt es eine Konstante $n \in \mathbb{N}$,
so dass für jedes Wort $z \in L$ mit $|z| \geq n$ gilt:

Wenn wir in z mindestens n Buchstaben markieren, so lässt sich z so als $z = uvwxy$ schreiben,

- dass von den mindestens n markierten Buchstaben
  - mindestens einer zu vx gehört und
  - höchstens n zu vwx gehören und
- für alle $i \geq 0$ das Wort $uv^i wx^i y \in L$ ist.

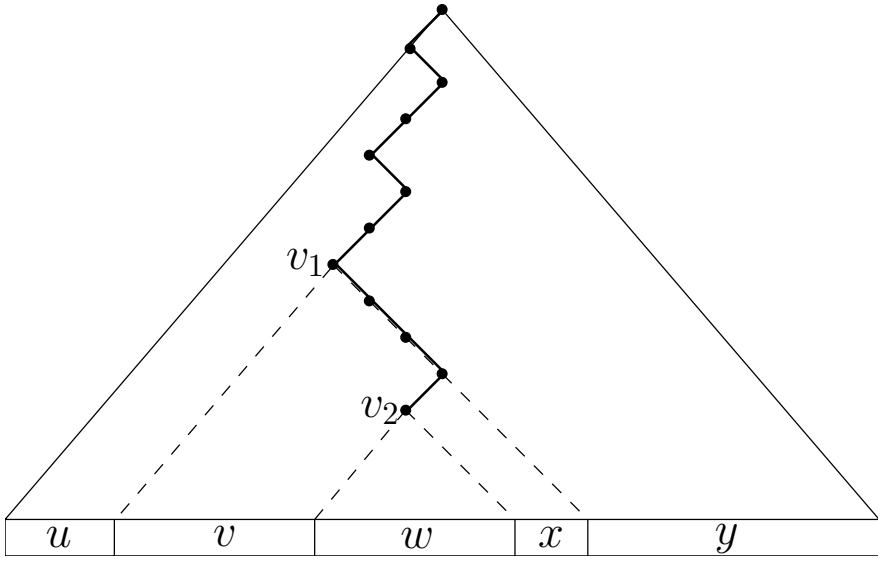
Beweis

- Sei L kontextfreie Sprache
- Sei G Grammatik zu L mit Variablen V in Chomsky-Normalform, d.h. alle Regeln sind von der Form $A \to BC$ oder $A \to a$.
- Setze $n := 2^{|V|+1}$.
- Wähle beliebiges Wort $z \in L$ mit $|z| \geq n$
- Betrachte einen Syntaxbaum T zu z.



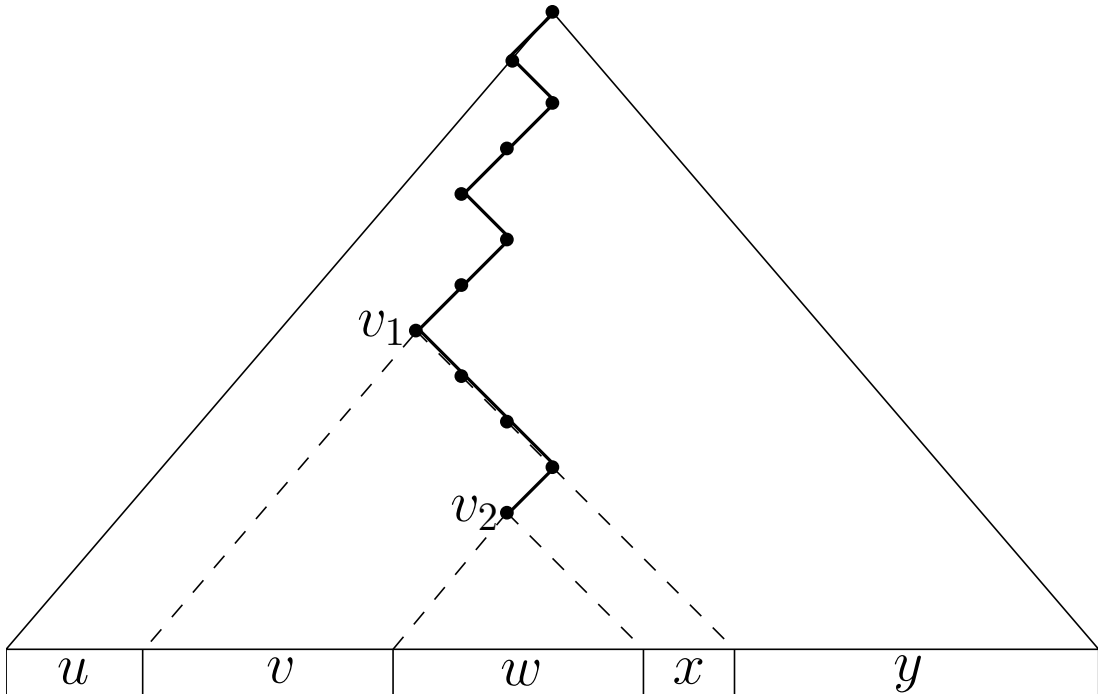
Beweis

- T hat $|z|$ Blätter und alle inneren Knoten außer den Vorgängern der Blätter haben Grad 2, ansonsten Grad 1.
- Seien mindestens n Blätter markiert.
- Durchlaufe einen Weg von der Wurzel zu einem Blatt.
  Wähle stets den Nachfolger, auf dessen Seite die größere Anzahl markierter Blätter liegt.
- Nenne Knoten auf dem Weg, für die rechter und linker Unterbaum markierte Blätter hat, **Verzweigungsknoten**.



Beweis

- Wegen $n > 2^{|V|}$ liegen auf dem Weg mindestens $|V| + 1$ Verzweigungsknoten
- Von den letzten $|V| + 1$ Verzweigungsknoten entsprechen mindestens zwei Knoten v_1, v_2 derselben Variablen A.
- Sei vwx Wort unter Teilbaum mit Wurzel v_1
- Sei w Wort unter Teilbaum mit Wurzel v_2.
- Damit sind u und y eindeutig bestimmt.



Beweis

- Da v_1 Verzweigungsknoten ist, enthält vx mindestens einen markierten Buchstaben.
- Da der Unterbaum von v_1 inkl. v_1 nur $|V| + 1$ Verzweigungsknoten enthält, gibt es in vwx höchstens $2^{|V|+1} = n$ markierte Buchstaben.
- Zu G existieren die Ableitungen

$$S \overset{*}{\to} uAy, \qquad A \overset{*}{\to} vAx, \qquad A \overset{*}{\to} w.$$

Daraus kann z abgeleitet werden durch

$$S \overset{*}{\to} uAy \overset{*}{\to} uvAxy \overset{*}{\to} uvwxy = z,$$

aber auch $uv^i wx^i y$ für jedes $i \geq 1$ durch

$$S \overset{*}{\to} uAy \overset{*}{\to} uvAxy \overset{*}{\to} uv^2Ax^2y \overset{*}{\to} \cdots \to uv^iAx^iy \to uv^iwx^iy.$$

Also ist auch $uv^i wx^i y \in L$ für $i \geq 0$.

Bemerkung

- Der Spezialfall von Odgen's Lemma, in dem alle Buchstaben von z markiert sind, ist gerade das Pumping–Lemma.

Satz:
Die Chomsky-Hierarchie ist echt, d.h.

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0 \ ,$$

wobei $\mathcal{L}_i, 0 \leq i \leq 3$, Klasse der durch Typ-$i$-Grammatiken erzeugten Sprachen.

Beweis - Teil 1

Es gibt eine kontextfreie Sprache, die nicht regulär ist.

Die Sprache

$$L = \{a^i b^i | i \geq 1\}$$

ist kontextfrei und wird durch die Grammatik

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{0, 1\} \\ R &= \{S \to 01 \mid 0S1\} . \end{aligned}$$

erzeugt. Sie ist aber nicht regulär.
(Beispiel (2) zum Pumping-Lemma, Vorlesung vom 26.1.2010)

Beweis - Teil 2

Es gibt eine kontextsensitive Sprache, die nicht kontextfrei ist.

Die Sprache

$$L = \{a^i b^i c^i \mid i \geq 1\}$$

ist kontextsensitiv.

Beweis

- L kontextsensitiv $\Leftrightarrow$ es gibt NTM mit linearem Speicherbedarf für L
- Eingabe $w \in \{a, b, c\}^*$
- Überprüfe deterministisch, ob $w = a^i b^j c^k$
- Überprüfe deterministisch, ob $j = i$ und $k = i$
- Speicherbedarf: $i + j + k$, also linear
- $\Rightarrow$ L kontextsensitiv

Beweis - Teil 2

Es gibt eine kontextsensitive Sprache, die nicht kontextfrei ist.

Die Sprache

$$L = \{a^i b^i c^i \mid i \geq 1\}$$

ist nicht kontextfrei.

Beweis - Teil 2

Pumping-Lemma für kontextfreie Sprachen

Für jede kontextfreie Sprache L
gibt es eine Konstante $n \in \mathbb{N}$,
so dass sich jedes Wort $z \in L$ mit $|z| \geq n$

so als

- $z = uvwxy$

schreiben lässt, dass

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- für alle $i \geq 0$ das Wort $uv^i wx^i y \in L$ ist.

Beweis - Teil 2

Die Sprache $L = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.

Beweis

- Annahme: L sei kontextfrei. Sei dann n wie im PL gefordert.
- Wähle das Wort $z = a^n b^n c^n \in L$.
- Wir betrachten eine Zerlegung $z = uvwxy$ wie im PL gefordert:
  - $|vx| \geq 1$,
  - $|vwx| \leq n$ und
  - für alle $i \geq 0$ ist das Wort $uv^i wx^i y \in L$.

Beweis - Teil 2

Die Sprache $L = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.

Beweis

- Annahme: L sei kontextfrei. Sei dann n wie im PL gefordert.
- Wähle das Wort $z = a^n b^n c^n \in L$.
- Wir betrachten eine Zerlegung $z = uvwxy$ wie im PL gefordert:
  - $|vx| \geq 1$,
  - $|vwx| \leq n$ und
  - für alle $i \geq 0$ ist das Wort $uv^i wx^i y \in L$.
- Fallunterscheidung, Fall 1: vwx besteht nur aus a und b
  - Dann enthalt vx mindestens ein a oder b.
  - Damit ist $uv^0 wx^0 y = a^i b^j c^n \notin L$ weil entweder $i < n$ oder $j < n$.
  - Dies ist ein Widerspruch zum PL.

Beweis - Teil 2

Die Sprache $L = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.

Beweis

- Annahme: L sei kontextfrei. Sei dann n wie im PL gefordert.
- Wähle das Wort $z = a^n b^n c^n \in L$.
- Wir betrachten eine Zerlegung $z = uvwxy$ wie im PL gefordert:
  - $|vx| \geq 1$,
  - $|vwx| \leq n$ und
  - für alle $i \geq 0$ ist das Wort $uv^i wx^i y \in L$.
- Fallunterscheidung, Fall 2: vwx besteht nur aus b und c
  - Dann enthalt vx mindestens ein b oder c.
  - Damit ist $uv^0 wx^0 y = a^n b^i c^j \notin L$ weil entweder $i < n$ oder $j < n$.
  - Dies ist ein Widerspruch zum PL.

Beweis - Teil 2

Die Sprache $L = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.

Ogden's Lemma für kontextfreie Sprachen

Für jede kontextfreie Sprache L
gibt es eine Konstante $n \in \mathbb{N}$,
so dass für jedes Wort $z \in L$ mit $|z| \geq n$ gilt:

Wenn wir in z mindestens n Buchstaben markieren, so lässt sich z so als $z = uvwxy$ schreiben,

- dass von den mindestens n markierten Buchstaben
  - mindestens einer zu vx gehört und
  - höchstens n zu vwx gehören und
- für alle $i \geq 0$ das Wort $uv^i wx^i y \in L$ ist.

Beweis - Teil 2

Die Sprache $L = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.

Alternativer Beweis mit Odgen's Lemma

- Annahme: L sei kontextfrei.
- Sei dann n wie in Odgen's Lemma gefordert.
- Wähle das Wort $z = a^{n+1} b^{n+1} c^{n+1} \in L$.
- Markiere alle b.
- Damit enthält vwx mindestens ein b aber kein a oder kein c.
- Es enthalte vwx kein c (anderer Fall analog)
- Damit ist $uv^0 wx^0 y = a^i b^j c^n \notin L$ weil entweder $i < n$ oder $j < n$.
- Dies ist ein Widerspruch zu Odgen's Lemma.

Beweis - Teil 3

Es gibt eine semi-entscheidbare Sprache, die nicht kontextsensitiv ist.

Es sei L_u die universelle Sprache.

Wiederholung
Die **universelle Sprache** L_u über $\{0,1\}$ ist definiert durch

$$L_u := \{wv \mid v \in L(T_w)\}.$$

L_u ist also die Menge aller Wörter wv für die T_w bei der Eingabe v hält und v akzeptiert.

Beweis - Teil 3

Es gibt eine semi-entscheidbare Sprache, die nicht kontextsensitiv ist.

Es sei L_u die universelle Sprache.

- L_u ist semi-entscheidbar, aber nicht entscheidbar (Kapitel 3).
- Wegen der Semi-entscheidbarkeit gilt $L_u \in \mathcal{L}_0$.

- Annahme: $L_u \in \mathcal{L}_1$.
- Dann gibt es eine NTM, die L_u mit linearem Speicher erkennt.
- Mit linearem Speicher können nur exponentiell viele verschiedene Konfigurationen auftreten.
- Diese könnte durch eine DTM durch Ausprobieren aller möglichen Konfigurationen simuliert werden.
- Dies wäre ein Widerspruch zur Nichtentscheidbarkeit von L_u.

Nutzlose Variablen

Sei G eine kontextfreie Grammatik. Eine Variable A heißt **nutzlos**, falls es keine Ableitung $S \overset{*}{\rightarrow} w$ gibt, $w \in \Sigma^*$, in der A vorkommt.

Satz:
Für eine kontextfreie Grammatik kann die Menge der nutzlosen Variablen (in polynomialer Zeit) berechnet werden.

Beweis:

- Wir benutzen ein zweistufiges Verfahren.

Schritt 1

Bestimme alle Variablen, die ein Wort erzeugen können

Formal: Berechne $V' = \{A \in V \mid \exists w \in \Sigma^* : A \overset{*}{\to} w\}$

- Initialisiere eine leere Queue Q.
- Füge alle $A \in V$ mit $A \to w$ für ein $w \in \Sigma^*$ in Q und V' ein.
- Entferne der Reihe nach jedes Element A aus Q
  - Ersetze jede Regel
    $$B \to \alpha A \beta \text{ mit } \alpha, \beta \in (V \cup \Sigma)^*$$
    durch die Regeln
    $$B \to \alpha w \beta, \text{ wobei } w \in \Sigma^* \text{ und } A \to w \text{ Regel.}$$
  - Wenn dabei eine Regel der Form
    $$B \to w', \ w' \in \Sigma^*,$$
    entsteht und $B \notin V'$, füge B in Q und V' ein.
- Das Verfahren endet, wenn Q leer ist.

Schritt 1

Karlsruher Institut für Technologie

Bemerkung 1

- Falls $S \notin V'$, breche das Verfahren ab.
- G erzeugt dann die leere Sprache und alle Variablen sind nutzlos.

Bemerkung 2

- Für jede Variable A mit $A \overset{*}{\to} w$ für ein $w \in \Sigma^*$ gilt:
- Per Induktion über die Länge der kürzesten Ableitungsregel der Form $A \overset{*}{\to} w$ kann für A gezeigt werden, dass $A \in V'$.

Beispiel: Schritt 1

Grammatik $G = (\Sigma, V, S, R)$ mit Produktionen R gegeben durch

$$
\begin{aligned}
S &\rightarrow Aa \mid B \mid Cab \\
A &\rightarrow bc \mid A \\
B &\rightarrow Bd \mid Cd \\
C &\rightarrow aBc \\
D &\rightarrow Ab \\
E &\rightarrow SD
\end{aligned}
$$

Beispiel: Schritt 1

Füge alle $A \in V$ mit $A \to w$ für ein $w \in \Sigma^*$ in Q und V' ein.

$$
\begin{array}{lcl}
S & \to & Aa|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & Ab \\
E & \to & SD
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \emptyset \\
Q & = & \emptyset
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & Aa|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & Ab \\
E & \to & SD
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A\} \\
Q & = & \{A\}
\end{array}
$$

Beispiel: Schritt 1

Entferne der Reihe nach jedes Element A aus Q

- Ersetze jede Regel $B \to \alpha A \beta$ mit $\alpha, \beta \in (V \cup \Sigma)^*$ durch die Regeln $B \to \alpha w \beta$, wobei $w \in \Sigma^*$ und $A \to w$ Regel.
- Wenn dabei eine Regel der Form $B \to w'$, $w' \in \Sigma^*$ entsteht und $B \notin V'$, füge B in Q und V' ein.

$$
\begin{array}{lcl}
S & \to & Aa \mid B \mid Cab \\
A & \to & bc \mid A \\
B & \to & Bd \mid Cd \\
C & \to & aBc \\
D & \to & Ab \\
E & \to & SD
\end{array}
\qquad
\begin{array}{lcl}
S & \to & bca \mid B \mid Cab \\
A & \to & bc \mid A \\
B & \to & Bd \mid Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & SD
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A\} \\
Q & = & \{A\}
\end{array}
\qquad
\begin{array}{lcl}
V' & = & \{A, S, D\} \\
Q & = & \{S, D\}
\end{array}
$$

Beispiel: Schritt 1

Entferne der Reihe nach jedes Element A aus Q

- Ersetze jede Regel $B \to \alpha A \beta$ mit $\alpha, \beta \in (V \cup \Sigma)^*$ durch die Regeln $B \to \alpha w \beta$, wobei $w \in \Sigma^*$ und $A \to w$ Regel.
- Wenn dabei eine Regel der Form $B \to w'$, $w' \in \Sigma^*$ entsteht und $B \notin V'$, füge B in Q und V' ein.

$$
\begin{array}{lcl}
S & \to & bca|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & SD
\end{array}
\qquad
\begin{array}{lcl}
V' & = & \{A, S, D\} \\
Q & = & \{S, D\}
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & bca|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & bcaD
\end{array}
\qquad
\begin{array}{lcl}
V' & = & \{A, S, D\} \\
Q & = & \{D\}
\end{array}
$$

Beispiel: Schritt 1

Entferne der Reihe nach jedes Element A aus Q

- Ersetze jede Regel $B \to \alpha A \beta$ mit $\alpha, \beta \in (V \cup \Sigma)^*$ durch die Regeln $B \to \alpha w \beta$, wobei $w \in \Sigma^*$ und $A \to w$ Regel.
- Wenn dabei eine Regel der Form $B \to w'$, $w' \in \Sigma^*$ entsteht und $B \notin V'$, füge B in Q und V' ein.

$$
\begin{array}{lcl}
S & \to & bca \,|\, B \,|\, Cab \\
A & \to & bc \,|\, A \\
B & \to & Bd \,|\, Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & bcaD
\end{array}
\qquad
\begin{array}{lcl}
V' & = & \{A, S, D\} \\
Q & = & \{D\}
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & bca \,|\, B \,|\, Cab \\
A & \to & bc \,|\, A \\
B & \to & Bd \,|\, Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & bcabcb
\end{array}
\qquad
\begin{array}{lcl}
V' & = & \{A, S, D, E\} \\
Q & = & \{E\}
\end{array}
$$

Beispiel: Schritt 1

Entferne der Reihe nach jedes Element A aus Q

- Ersetze jede Regel $B \to \alpha A \beta$ mit $\alpha, \beta \in (V \cup \Sigma)^*$ durch die Regeln $B \to \alpha w \beta$, wobei $w \in \Sigma^*$ und $A \to w$ Regel.
- Wenn dabei eine Regel der Form $B \to w'$, $w' \in \Sigma^*$ entsteht und $B \notin V'$, füge B in Q und V' ein.

$$
\begin{array}{lcl}
S & \to & bca \,|\, B \,|\, Cab \\
A & \to & bc \,|\, A \\
B & \to & Bd \,|\, Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & bcabcb
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A, S, D, E\} \\
Q & = & \{E\}
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & bca \,|\, B \,|\, Cab \\
A & \to & bc \,|\, A \\
B & \to & Bd \,|\, Cd \\
C & \to & aBc \\
D & \to & bcb \\
E & \to & bcabcb
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A, S, D, E\} \\
Q & = & \{\}
\end{array}
$$

Schritt 2

Bestimme alle Variablen in V', die vom Startsymbol aus „erreicht“ werden können.
Formal: Berechne $\{A \in V' \mid S = A \text{ oder } \exists \alpha, \beta \in (V' \cup \Sigma)^* : S \overset{*}{\to} \alpha A \beta\}$

- Starte mit $V'' = \{S\}$
- Füge zu allen Regeln $A \to \alpha B \beta$ mit $\alpha, \beta \in (V' \cup \Sigma)^*, A \in V'', B \in V'$ die Variable B in V'' ein.
- Wiederhole den letzen Schritt, bis sich V'' nicht mehr ändert.

Per Induktion über die Länge der kürzesten Ableitungsregel der Form $S \to \alpha A \beta,\ \alpha, \beta \in (V' \cup \Sigma)^*$, kann dann wieder die Korrektheit bewiesen werden.

Fazit: Nach Ende von Schritt 2 ist V'' die Menge aller nützlichen Variablen.

Beispiel: Schritt 2

Starte mit $V'' = \{S\}$

$$
\begin{array}{lcl}
S & \to & Aa|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & Ab \\
E & \to & SD
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A, S, D, E\} \\
V'' & = & \{\}
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & Aa|B|Cab \\
A & \to & bc|A \\
B & \to & Bd|Cd \\
C & \to & aBc \\
D & \to & Ab \\
E & \to & SD
\end{array}
$$

$$
\begin{array}{lcl}
V' & = & \{A, S, D, E\} \\
V'' & = & \{S\}
\end{array}
$$

Beispiel: Schritt 2

Füge zu allen Regeln $A \to \alpha B \beta$ mit $\alpha, \beta \in (V' \cup \Sigma)^*, A \in V''$, $B \in V'$ die Variable B in V'' ein.

$$\begin{aligned} S &\to Aa|B|Cab \\ A &\to bc|A \\ B &\to Bd|Cd \\ C &\to aBc \\ D &\to Ab \\ E &\to SD \end{aligned}$$

$$\begin{aligned} V' &= \{A, S, D, E\} \\ V'' &= \{S\} \end{aligned}$$

$$\begin{aligned} S &\to Aa|B|Cab \\ A &\to bc|A \\ B &\to Bd|Cd \\ C &\to aBc \\ D &\to Ab \\ E &\to SD \end{aligned}$$

$$\begin{aligned} V' &= \{A, S, D, E\} \\ V'' &= \{S, A\} \end{aligned}$$

Beispiel: Schritt 2

Wiederhole den letzen Schritt, bis sich V'' nicht mehr ändert.

$$
\begin{aligned}
S &\to Aa|B|Cab \\
A &\to bc|A \\
B &\to Bd|Cd \\
C &\to aBc \\
D &\to Ab \\
E &\to SD
\end{aligned}
\qquad
\begin{aligned}
S &\to Aa|B|Cab \\
A &\to bc|A \\
B &\to Bd|Cd \\
C &\to aBc \\
D &\to Ab \\
E &\to SD
\end{aligned}
$$

$$
\begin{aligned}
V' &= \{A, S, D, E\} \\
V'' &= \{S, A\}
\end{aligned}
\qquad
\begin{aligned}
V' &= \{A, S, D, E\} \\
V'' &= \{S, A\}
\end{aligned}
$$

Korollar

Korollar
Für eine kontextfreie Grammatik G kann (in polynomialer Zeit) entschieden werden, ob $L(G) = \emptyset$ ist.

Beweis:

- $L(G) = \emptyset$ genau dann, wenn S nutzlos.

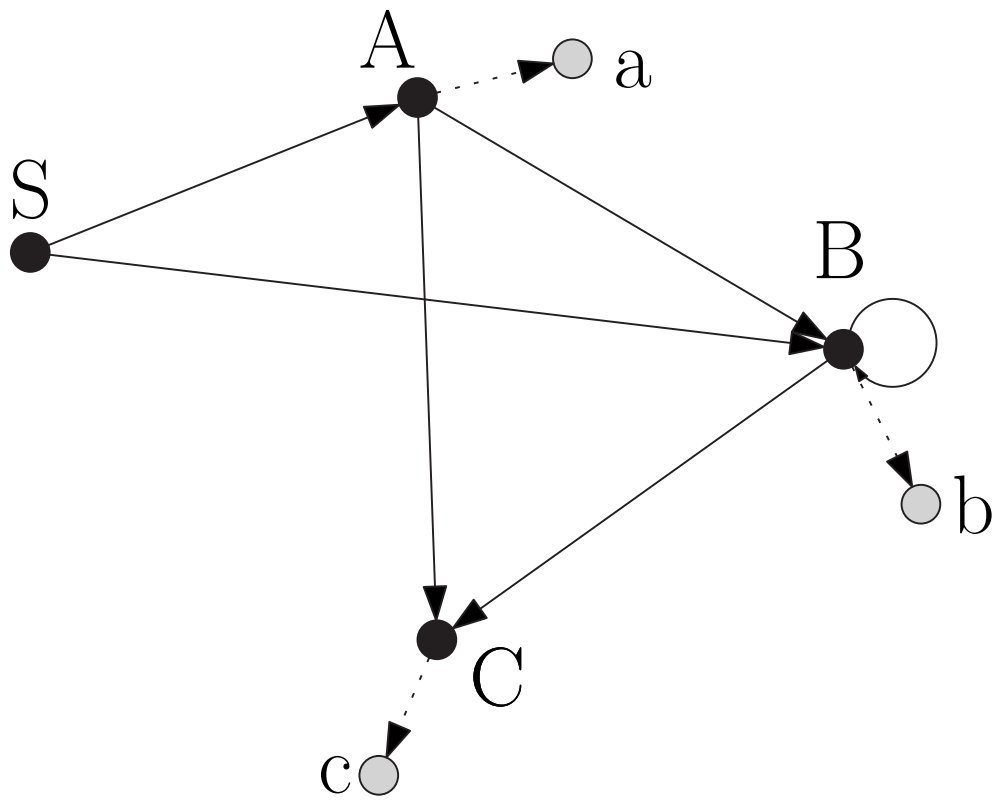
Satz:
Für eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ kann (in polynomialer Zeit) entschieden werden, ob $L(G)$ endlich ist.

Beweis:

- Entferne alle nutzlosen Variablen
- Überführe G in eine äquivalente Grammatik in Chomsky-Normalform.
- Betrachte den gerichteten Graphen (V, E) mit
  - Knotenmenge V ist gleich der Variablenmenge von G
  - Kantenmenge $E = \{(A, B) \mid \exists C \in V : A \to BC \in R \vee A \to CB \in R\}$
- Mit Tiefensuche kann entschieden werden, ob dieser Graph einen Kreis enthält.
- Man kann sich leicht überlegen, dass $L(G)$ genau dann endlich ist, wenn der entsprechende Graph keinen Kreis enthält.

Beispielgraph

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow BC \\
B &\rightarrow BC \\
A &\rightarrow a \\
B &\rightarrow b \\
C &\rightarrow c
\end{aligned}
$$



Satz:
Die Klasse der kontextfreien Sprachen ist abgeschlossen bzgl. Vereinigung, Konkatenation und Kleenschem Abschluss.

Beweis:

- Seien L_1 kontextfreie Sprache mit Grammatik $G_1 = (\Sigma, V_1, S_1, R_1)$
- Seien L_2 kontextfreie Sprache mit Grammatik $G_2 = (\Sigma, V_2, S_2, R_2)$
- o.B.d.A. sei $V_1 \cap V_2 = \emptyset$.

Vereinigung: Die Grammatik

$$\begin{aligned} V &= V_1 \cup V_2 \cup \{S\} \\ S &\quad \text{neues Startsymbol} \\ R &= R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\} \end{aligned}$$

erzeugt $L_1 \cup L_2$.

Satz:
Die Klasse der kontextfreien Sprachen ist abgeschlossen bzgl. Vereinigung, Konkatenation und Kleenschem Abschluss.

Beweis:

- Seien L_1 kontextfreie Sprache mit Grammatik $G_1 = (\Sigma, V_1, S_1, R_1)$
- Seien L_2 kontextfreie Sprache mit Grammatik $G_2 = (\Sigma, V_2, S_2, R_2)$
- o.B.d.A. sei $V_1 \cap V_2 = \emptyset$.

Konkatenation: Die Grammatik

$$\begin{aligned} V &= V_1 \cup V_2 \cup \{S\} \\ S &\quad \text{neues Startsymbol} \\ R &= R_1 \cup R_2 \cup \{S \to S_1 S_2\} \end{aligned}$$

erzeugt $L_1 \cdot L_2$.

Satz:
Die Klasse der kontextfreien Sprachen ist abgeschlossen bzgl. Vereinigung, Konkatenation und Kleenschem Abschluss.

Beweis:

- Seien L_1 kontextfreie Sprache mit Grammatik $G_1 = (\Sigma, V_1, S_1, R_1)$
- Seien L_2 kontextfreie Sprache mit Grammatik $G_2 = (\Sigma, V_2, S_2, R_2)$
- o.B.d.A. sei $V_1 \cap V_2 = \emptyset$.

Kleenscher Abschluss: Die Grammatik

$$\begin{aligned} V &= V_1 \cup \{S\} \\ S &\quad \text{neues Startsymbol} \\ R &= R_1 \cup \{S \to \varepsilon, S \to SS, S \to S_1\} \end{aligned}$$

erzeugt L_1^*.

Satz:
Die Klasse der kontextfreien Sprachen ist nicht abgeschlossen bzgl. Komplementbildung und Durchschnitt.

Beweis Schnitt: Betrachte die kontextfreien Sprachen

$$L_1 = \{a^n b^n | n \geq 1\} \qquad L_2 = \{c\}^*$$
$$L_3 = \{a\}^* \qquad L_4 = \{b^n c^n | n \geq 1\}$$

Nach dem letzen Satz sind dann auch $L_1 \cdot L_2$ und $L_3 \cdot L_4$ kontextfrei.
Es ist dann

$$L := L_1 L_2 \cap L_3 L_4 = \{a^n b^n c^n | n \geq 1\} .$$

Diese Sprache ist nicht kontextfrei.

Satz:
Die Klasse der kontextfreien Sprachen ist nicht abgeschlossen bzgl. Komplementbildung und Durchschnitt.

Beweis Komplementbildung:

- Angenommen, die Klasse der kontextfreien Sprachen wäre bzgl. Komplementbildung abgeschlossen.
- Dann würde für beliebige kontextfreie Sprachen L_1, L_2 gelten $(L_1^c \cup L_2^c)^c = L_1 \cap L_2$ ist wieder kontextfrei.
- Dies ist ein Widerspruch zur ersten Aussage des Satzes.

Greibach Normalform

Greibach Normalform
Eine kontextfreie Grammatik ist in **Greibach-Normalform**, wenn alle Ableitungsregeln von der Form

$$A \to a\alpha \text{ mit } A \in V, a \in \Sigma \text{ und } \alpha \in V^*$$

sind.

Satz:
Für jede kontextfreie Grammatik G, für die $L(G)$ das leere Wort nicht enthält, kann eine (äquivalente) kontextfreie Grammatik G' mit $L(G) = L(G')$ in Greibach-Normalform konstruiert werden.

Beweis - Ersetzung (i)

Folgende Ersetzungen ändern nichts an der erzeugten Sprache:

Ersetzung (i). Eine Regel

$$A \to \alpha_1 B \alpha_2$$

wobei

$$B \to \beta_1, \quad B \to \beta_2, \quad \ldots, \quad B \to \beta_r$$

alle Regeln sind, deren linke Seite B ist, kann durch die Regeln

$$\begin{aligned} A &\to \alpha_1 \beta_1 \alpha_2 \\ A &\to \alpha_1 \beta_2 \alpha_2 \\ &\ldots \\ A &\to \alpha_1 \beta_r \alpha_2 \end{aligned}$$

ersetzt werden.

Beweis - Ersetzung (ii)

Folgende Ersetzungen ändern nichts an der erzeugten Sprache:

Ersetzung (ii). Seien

$$\begin{aligned} A &\to A\alpha_1, \ldots, A \to A\alpha_r \\ A &\to \beta_1, \ldots, A \to \beta_s \end{aligned}$$

alle Regeln, deren linke Seite A ist, wobei β_i nicht mit A beginnen. Dann können die Regeln

$$A \to A\alpha_1, \ldots, A \to A\alpha_r$$

durch die Regeln

$$\begin{aligned} A &\to \beta_1 B, \ldots, A \to \beta_s B \\ B &\to \alpha_1, \ldots, B \to \alpha_r, \\ B &\to \alpha_1 B, \ldots, B \to \alpha_r B \end{aligned}$$

ersetzt werden. Dabei sei B eine neu eingeführte Variable.

Beweis - Definitionen

Annahme G ist in Chomsky-Normalform mit

$$V = \{A_1, \ldots, A_m\}$$
$$\Sigma = \{a_1, \ldots, a_n\}$$

und damit ausschließlich Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Die Grammatik in Greibach-Normalform wird zusätzlich die Variablen $\{B_1, \ldots, B_m\}$ benutzen. Sei

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\}$$

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Während der Umformung sind folgende Invarianten erfüllt.

1.Invariante
Die rechte Seite einer Regel, deren rechte Seite mit einer Variablen beginnt, besteht nur aus Variablen.

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Während der Umformung sind folgende Invarianten erfüllt.

2.Invariante
Die rechte Seite einer Regel, deren rechte Seite mit einer Variablen beginnt, beginnt mit einer Variablen aus $V = \{A_1, \ldots, A_m\}$.

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Während der Umformung sind folgende Invarianten erfüllt.

3.Invariante
Symbole aus Σ kommen nur als erstes Zeichen der rechten Seite einer Regel vor.

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Während der Umformung sind folgende Invarianten erfüllt.

4.Invariante
Die rechte Seite einer Regel, deren linke Seite aus $V = \{A_1, \ldots, A_m\}$ ist und deren rechte Seite mit einer Variablen aus V beginnt, beginnt sogar mit zwei Variablen aus V.

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Während der Umformung sind folgende Invarianten erfüllt.

5.Invariante
Die rechte Seite einer Regel, deren linke Seite aus $V' \setminus V = \{B_1, \ldots, B_m\}$ ist, besteht nur aus Variablen aus V'.

Beweis: Invarianten

$$V' := \{A_1, \ldots, A_m, B_1, \ldots, B_m\} \qquad \Sigma = \{a_1, \ldots, a_n\}$$

zu Beginn mit Regeln der Form

$$A_i \to A_j A_k$$
$$A_i \to a_j \ .$$

Wir formen G zunächst so um, dass außer Invarianten 1-5 noch die nächste Invariante gilt:

6.Invariante
Falls $A_i \to A_j \alpha$ Regel ist, so gilt $j > i$.

Beweis - Übersicht Invarianten

1.Invariante Die rechte Seite einer Regel, deren rechte Seite mit einer Variablen beginnt, besteht nur aus Variablen.

2.Invariante Die rechte Seite einer Regel, deren rechte Seite mit einer Variablen beginnt, beginnt mit einer Variablen aus $V = \{A_1, \ldots, A_m\}$.

3.Invariante Symbole aus Σ kommen nur als erstes Zeichen der rechten Seite einer Regel vor.

4.Invariante Die rechte Seite einer Regel, deren linke Seite aus $V = \{A_1, \ldots, A_m\}$ ist und deren rechte Seite mit einer Variablen aus V beginnt, beginnt sogar mit zwei Variablen aus V.

5.Invariante Die rechte Seite einer Regel, deren linke Seite aus $V' \setminus V = \{B_1, \ldots, B_m\}$ ist, besteht nur aus Variablen aus V'.

6.Invariante Falls $A_i \to A_j\alpha$ Regel ist, so gilt $j > i$.

Beweis: Beispielgrammatik

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\qquad A_2 \to A_3 A_1,\ A_2 \to 1, \\
&\qquad A_3 \to A_1 A_2,\ A_3 \to 0\}
\end{aligned}
$$

Beweis - Verfahren - Schritt 1

Vorher: Grammatik in Chomsky-Normalform
Nachher: 6. Invariante hält: Falls $A_i \to A_j\alpha$ Regel ist, so gilt $j > i$.

Aktion: Dabei wenden wir (in dieser Reihenfolge)

Ersetzung (ii)	zur Ersetzung aller Regeln $A_1 \to A_1\alpha$
Ersetzung (i)	zur Ersetzung aller Regeln $A_2 \to A_1\alpha$
Ersetzung (ii)	zur Ersetzung aller Regeln $A_2 \to A_2\alpha$
Ersetzung (i)	zur Ersetzung aller Regeln $A_3 \to A_1\alpha$
Ersetzung (i)	zur Ersetzung aller Regeln $A_3 \to A_2\alpha$
Ersetzung (ii)	zur Ersetzung aller Regeln $A_3 \to A_3\alpha$
	$\vdots$

Beweis - Schritt 1 - Beispiel

Ersetzung (i)

$$
\begin{aligned}
A &\to \alpha_1 B \alpha_2 \\
B &\to \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
\qquad
\begin{aligned}
A &\to \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \ldots | \alpha_1 \beta_r \alpha_2 \\
B &\to \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\quad A_2 \to A_3 A_1,\ A_2 \to 1, \\
&\quad A_3 \to A_1 A_2,\ A_3 \to 0\}
\end{aligned}
\qquad
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\quad A_2 \to A_3 A_1,\ A_2 \to 1, \\
&\quad A_3 \to A_2 A_3 A_2,\ A_3 \to 0\}
\end{aligned}
$$

Beweis - Schritt 1 - Beispiel

Ersetzung (i)

$$
\begin{aligned}
A &\to \alpha_1 B \alpha_2 \\
B &\to \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\quad A_2 \to A_3 A_1,\ A_2 \to 1, \\
&\quad A_3 \to A_2 A_3 A_2,\ A_3 \to 0\}
\end{aligned}
$$

$$
\begin{aligned}
A &\to \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \ldots | \alpha_1 \beta_r \alpha_2 \\
B &\to \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\quad A_2 \to A_3 A_1,\ A_2 \to 1, \\
&\quad A_3 \to A_3 A_1 A_3 A_2 | 1 A_3 A_2, \\
&\quad A_3 \to 0\}
\end{aligned}
$$

Beweis - Schritt 1 - Beispiel

Ersetzung (ii)

$$
\begin{aligned}
A &\to A\alpha_1, \ldots, A \to A\alpha_r \\
A &\to \beta_1, \ldots, A \to \beta_s
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2A_3, \\
&\quad A_2 \to A_3A_1,\ A_2 \to 1, \\
&\quad A_3 \to A_3A_1A_3A_2, \\
&\quad A_3 \to 0 \,|\, 1A_3A_2\}
\end{aligned}
$$

$$
\begin{aligned}
A &\to \beta_1 B, \ldots, A \to \beta_s B \\
B &\to \alpha_1, \ldots, B \to \alpha_r, \\
B &\to \alpha_1 B, \ldots, B \to \alpha_r B \\
A &\to \beta_1, \ldots, A \to \beta_s
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2A_3, \\
&\quad A_2 \to A_3A_1,\ A_2 \to 1, \\
&\quad A_3 \to 0B_3 \,|\, 1A_3A_2B_3, \\
&\quad B_3 \to A_1A_3A_2 \,|\, A_1A_3A_2B_3 \\
&\quad A_3 \to 0 \,|\, 1A_3A_2\}
\end{aligned}
$$

Beweis - Verfahren - Schritt 2

Vorher: Alle Regeln sind von der Form

$$\begin{aligned} A &\to a\alpha \\ A &\to \alpha \text{ mit } \alpha \in (V')^*, a \in \Sigma \end{aligned}$$

wobei es keine ε-Regeln oder Kettenregeln mit linker Seite $A \in V$ gibt.
Wegen Invariante 6

- gibt es keine Regel $A_m \to \alpha$
- beginnen alle $A_{m-1} \to \alpha$-Regeln mit A_m.

Aktion: Ersetze mit absteigenden k alle Regeln der Form

$$A_k \to \alpha \text{ mit } \alpha \in (V')^*$$

mittels Ersetzung (i).

Nachher: Regeln mit linker Seite in V in Form $A_k \to a\alpha, a \in \Sigma, \alpha \in (V')^*$.

Beweis - Schritt 2 - Beispiel

Ersetzung (i)

$$
\begin{aligned}
A &\rightarrow \alpha_1 B \alpha_2 \\
B &\rightarrow \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \rightarrow A_2 A_3, \\
&\quad A_2 \rightarrow A_3 A_1, \\
&\quad A_2 \rightarrow 1, \\
&\quad A_3 \rightarrow 0 B_3 | 1 A_3 A_2 B_3, \\
&\quad A_3 \rightarrow 0 | 1 A_3 A_2\} \\
&\quad B_3 \rightarrow A_1 A_3 A_2 | A_1 A_3 A_2 B_3
\end{aligned}
$$

$$
\begin{aligned}
A &\rightarrow \alpha_1 \beta_1 \alpha_2 | \alpha_1 \beta_2 \alpha_2 | \ldots | \alpha_1 \beta_r \alpha_2 \\
B &\rightarrow \beta_1 | \beta_2, \ldots | \beta_r
\end{aligned}
$$

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \rightarrow A_2 A_3, \\
&\quad A_2 \rightarrow 0 B_3 A_1 | 1 A_3 A_2 B_3 A_1 \\
&\quad A_2 \rightarrow 0 A_1 | 1 A_3 A_2 A_1 \\
&\quad A_2 \rightarrow 1, \\
&\quad A_3 \rightarrow 0 B_3 | 1 A_3 A_2 B_3, \\
&\quad A_3 \rightarrow 0 | 1 A_3 A_2\} \\
&\quad B_3 \rightarrow A_1 A_3 A_2 | A_1 A_3 A_2 B_3
\end{aligned}
$$

Beweis - Schritt 2 - Beispiel

Ersetzung (i)

$$
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to A_2 A_3, \\
&\quad A_2 \to 0B_3A_1 \,|\, 1A_3A_2B_3A_1 \\
&\quad A_2 \to 0A_1 \,|\, 1A_3A_2A_1 \,|\, 1 \\
&\quad A_3 \to 0B_3 \,|\, 1A_3A_2B_3, \\
&\quad A_3 \to 0 \,|\, 1A_3A_2\} \\
&\quad B_3 \to A_1A_3A_2 \,|\, A_1A_3A_2B_3
\end{aligned}
\qquad
\begin{aligned}
V &= \{A_1, A_2, A_3\} \\
\Sigma &= \{0, 1\} \\
S &= A_1 \\
R &= \{A_1 \to 0B_3A_1A_3 \\
&\quad A_1 \to 1A_3A_2B_3A_1A_3, \\
&\quad A_1 \to 0A_1A_3 \,|\, 1A_3A_2A_1A_3 \,|\, 1A_3, \\
&\quad A_2 \to 0B_3A_1 \,|\, 1A_3A_2B_3A_1 \\
&\quad A_2 \to 0A_1 \,|\, 1A_3A_2A_1 \,|\, 1 \\
&\quad A_3 \to 0B_3 \,|\, 1A_3A_2B_3, \\
&\quad A_3 \to 0 \,|\, 1A_3A_2\} \\
&\quad B_3 \to A_1A_3A_2 \,|\, A_1A_3A_2B_3
\end{aligned}
$$

Beweis - Verfahren - Schritt 3

Vorher: Rechte Seiten von Regeln, deren linke Seite aus $V' \setminus V = \{B_1, \ldots, B_m\}$ ist, beginnen mit einer Variablen aus $\{A_1, \ldots, A_m\}$ (wegen Invarianten 2 und 5).

Aktion: Ersetze Regeln $B_i \to A_j\alpha$, $\alpha \in (\Sigma \cup V')^*$ mit Ersetzung (i).

Nachher: G ist in Greibach-Normalform.

Beweis - Schritt 3 - Beispiel

Ersetzung (i)

$$
\begin{aligned}
A_1 &\to 0B_3A_1A_3 \\
A_1 &\to 1A_3A_2B_3A_1A_3, \\
A_1 &\to 0A_1A_3 \mid 1A_3A_2A_1A_3 \mid 1A_3, \\
A_2 &\to 0B_3A_1 \mid 1A_3A_2B_3A_1 \\
A_2 &\to 0A_1 \mid 1A_3A_2A_1 \mid 1 \\
A_3 &\to 0B_3 \mid 1A_3A_2B_3, \\
A_3 &\to 0 \mid 1A_3A_2\}
\end{aligned}
\qquad
\begin{aligned}
A_1 &\to 0B_3A_1A_3 \\
A_1 &\to 1A_3A_2B_3A_1A_3, \\
A_1 &\to 0A_1A_3 \mid 1A_3A_2A_1A_3 \mid 1A_3, \\
A_2 &\to 0B_3A_1 \mid 1A_3A_2B_3A_1 \\
A_2 &\to 0A_1 \mid 1A_3A_2A_1 \mid 1 \\
A_3 &\to 0B_3 \mid 1A_3A_2B_3, \\
A_3 &\to 0 \mid 1A_3A_2\}
\end{aligned}
$$

$$B_3 \to A_1A_3A_2 \mid A_1A_3A_2B_3$$

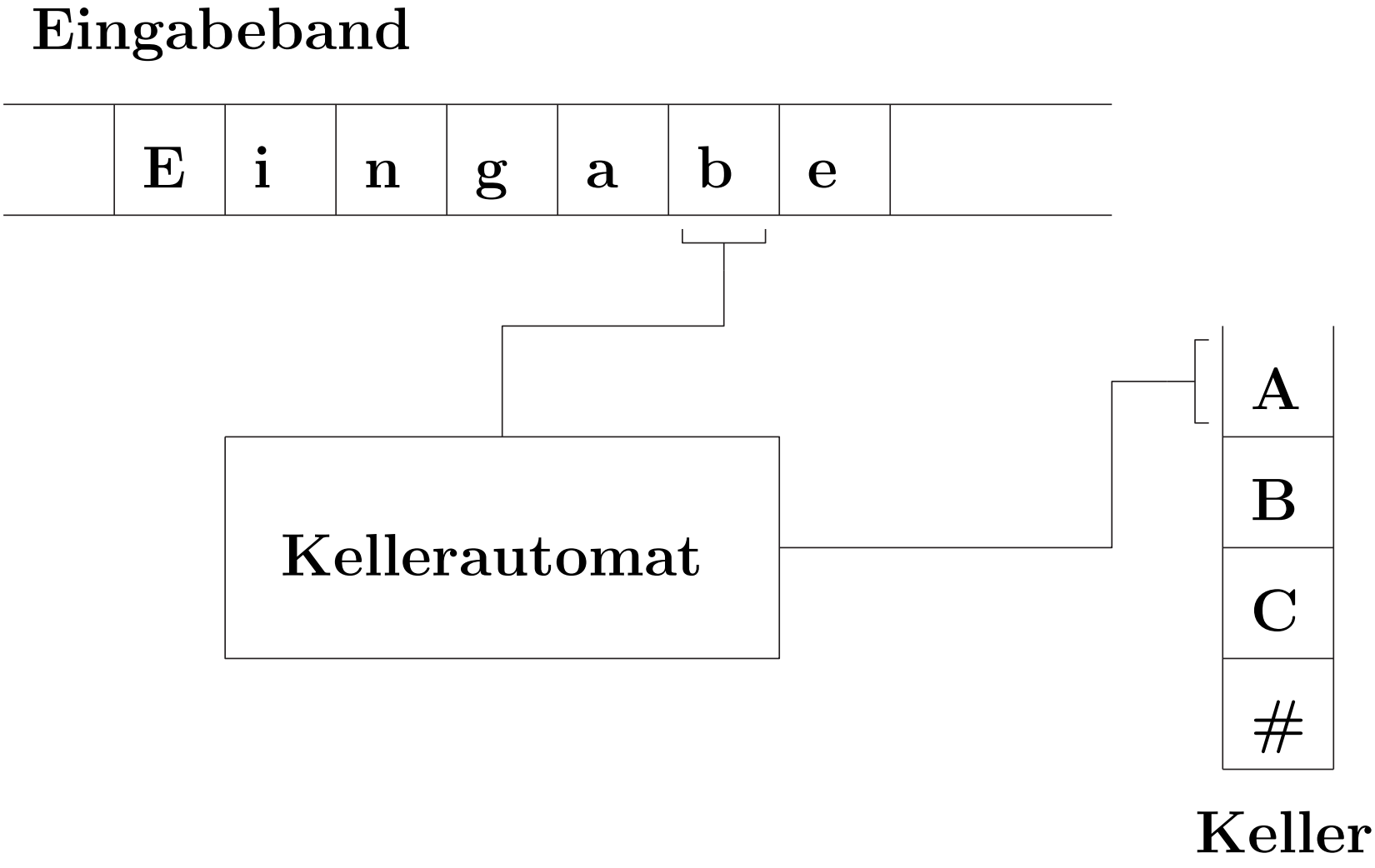
$$
\begin{aligned}
B_3 &\to 1A_3A_3A_2 \mid 0B_3A_1A_3A_3A_2 \\
B_3 &\to 1A_3A_2A_1A_3A_3A_2 \mid 0A_1A_3A_3A_2 \\
B_3 &\to 1A_3A_2B_3A_1A_3A_3A_2 \mid 1A_3A_3A_2B_3 \\
B_3 &\to 0B_3A_1A_3A_3A_2B_3 \mid 1A_3A_2A_1A_3A_3A_2B_3 \\
B_3 &\to 0A_1A_3A_3A_2B_3 \\
B_3 &\to 1A_3A_2B_3A_1A_3A_3A_2B_3
\end{aligned}
$$

Kellerautomaten

Ein (nichtdeterministischer) **Kellerautomat** (NPDA bzw. PDA, Push-down Automaton) besteht aus $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, wobei

- Q endliche Zustandsmenge
- Σ endliches Eingabealphabet
- Γ endliches STACK-Alphabet
- $q_0 \in Q$ Anfangszustand
- $Z_0 \in \Gamma$ Initialisierung des STACK
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$, d.h.
  - $\delta(q, a, Z) \subseteq \{(q, \gamma) : q \in Q, \gamma \in \Gamma^*\}$
  - $\delta(q, \varepsilon, Z) \subseteq \{(q, \gamma) : q \in Q, \gamma \in \Gamma^*\}$
- $F \subseteq Q$ Menge der akzeptierenden Endzustände, $F = \emptyset$ ist möglich.

Kellerautomaten - Visualisierung



Kellerautomaten - Arbeitsweise

Eine **Konfiguration eines PDA** ist ein Tripel (q, w, α) mit

- $q \in Q$,
- $w \in \Sigma^*$ der Teil der Eingabe, der noch nicht gelesen wurde,
- $\alpha \in \Gamma^*$ STACK-Inhalt.

Zu Konfiguration $(q, w_1 \dots w_k, Z_1 \dots Z_m)$ gibt es die **Nachfolgekonfigurationen**:

$(q', w_2 \dots w_k, Z'_1 \dots Z'_r Z_2 \dots Z_m)$ für alle $(q', Z'_1 \dots Z'_r) \in \delta(q, w_1, Z_1)$

und

$(q', w_1 \dots w_k, Z'_1 \dots Z'_r Z_2 \dots Z_m)$ für alle $(q', Z'_1 \dots Z'_r) \in \delta(q, \varepsilon, Z_1)$.

Kellerautomaten - Arbeitsweise

Karlsruher Institut für Technologie

Ein PDA **akzeptiert** ein $w \in \Sigma^*$ **durch leeren Stack**, wenn es eine zulässige Folge von Konfigurationen aus der Anfangskonfiguration (q_0, w, Z_0) in eine Konfiguration $(q, \varepsilon, \varepsilon)$, $q \in Q$, gibt.

Ein PDA **akzeptiert** ein $w \in \Sigma^*$ **durch einen akzeptierenden Endzustand**, wenn es eine zulässige Folge von Konfigurationen aus der Anfangskonfiguration (q_0, w, Z_0) in eine Konfiguration (q, ε, γ) mit $q \in F$ und $\gamma \in \Gamma^*$ gibt.

Ein PDA ist **deterministisch** (DPDA), falls

$$|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$$

für alle $q \in Q$, $a \in \Sigma$, $Z \in \Gamma$.

Kellerautomaten - Beispiel

Ein DPDA für $L = \{w\#w^R | w \in \{0,1\}^*\}$. Informelle Beschreibung:

- Betrachte beliebiges Wort $w_1 \dots w_n\#w_n \dots w_1 \in L$.

Phase 1

- Lies $w_1 \dots w_n$ und schreibe jeweils w_i auf den STACK bis # gelesen.

Phase 2

- Lies $w_n \dots w_1$ und vergleiche den jeweils gelesenen Buchstaben mit dem jeweils obersten Buchstaben auf dem STACK.
  - Gleichheit: Nimm obersten Buchstaben vom STACK
  - Sonst: Stoppe in nichtakzeptierenden Zustand

Phase 3

- Ist nur noch Z_0 auf dem STACK
  - Entferne Z_0
  - Akzeptiere die Eingabe „mit leerem STACK“

Kellerautomaten - Beispiel

Ein DPDA für $L = \{w\#w^R | w \in \{0,1\}^*\}$.

$$(Q = \{q_0, q_1, q_2\},\ \Sigma = \{0,1,\#\},\ \Gamma = \Sigma \cup \{Z_0\}),\ q_0,\ Z_0,\ \delta,\ \emptyset)$$

$\delta(q_0, 0, Z_0)$	=	$(q_1, 0Z_0)$	Phase 1
$\delta(q_0, 1, Z_0)$	=	$(q_1, 1Z_0)$	
$\delta(q_1, 0, 0)$	=	$(q_1, 00)$	
$\delta(q_1, 0, 1)$	=	$(q_1, 01)$	
$\delta(q_1, 1, 0)$	=	$(q_1, 10)$	
$\delta(q_1, 1, 1)$	=	$(q_1, 11)$	
$\delta(q_1, \#, 0)$	=	$(q_2, 0)$	Trennzeichen gelesen $\Rightarrow$ Zu Phase 2
$\delta(q_1, \#, 1)$	=	$(q_2, 1)$	
$\delta(q_2, 0, 0)$	=	(q_2, ε)	Phase 2
$\delta(q_2, 1, 1)$	=	(q_2, ε)	
$\delta(q_2, \varepsilon, Z_0)$	=	(q_2, ε)	Phase 3

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0)
\end{aligned}
$$

$$
\begin{aligned}
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11)
\end{aligned}
$$

$$
\begin{aligned}
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1)
\end{aligned}
$$

$$
\begin{aligned}
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon)
\end{aligned}
$$

$$
\delta(q_2, \varepsilon, Z_0) = (q_2, \varepsilon)
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$
q_1	#100	$100Z_0$

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$
q_1	#100	$100Z_0$
q_2	100	$100Z_0$
q_2	00	$00Z_0$

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$
q_1	#100	$100Z_0$
q_2	100	$100Z_0$
q_2	00	$00Z_0$
q_2	0	$0Z_0$

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$
q_1	#100	$100Z_0$
q_2	100	$100Z_0$
q_2	00	$00Z_0$
q_2	0	$0Z_0$
q_2		Z_0

Beispiel - Berechnung für Eingabe 001#100

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= (q_1, 0Z_0) \\
\delta(q_0, 1, Z_0) &= (q_1, 1Z_0) \\
\\
\delta(q_1, 0, 0) &= (q_1, 00) \\
\delta(q_1, 0, 1) &= (q_1, 01) \\
\delta(q_1, 1, 0) &= (q_1, 10) \\
\delta(q_1, 1, 1) &= (q_1, 11) \\
\\
\delta(q_1, \#, 0) &= (q_2, 0) \\
\delta(q_1, \#, 1) &= (q_2, 1) \\
\\
\delta(q_2, 0, 0) &= (q_2, \varepsilon) \\
\delta(q_2, 1, 1) &= (q_2, \varepsilon) \\
\\
\delta(q_2, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}
$$

Zustand	Eingabe	Stack
q_0	001#100	Z_0
q_1	01#100	$0Z_0$
q_1	1#100	$00Z_0$
q_1	#100	$100Z_0$
q_2	100	$100Z_0$
q_2	00	$00Z_0$
q_2	0	$0Z_0$
q_2		Z_0

akzeptiert durch leeren Stack

Kellerautomaten - Beispiel 2

Ein NPDA für $L = \{ww^R | w \in \{0,1\}^*\}$. Informelle Beschreibung:

- In diesem Fall fehlt das Trennzeichen.
- Der NPDA funktioniert wie der DPDA aus dem letzten Beispiel
- Der Übergang in Phase 2 funktioniert allerdings nichtdeterministisch.

Bemerkung:

- Für die Sprache $L = \{ww^R | w \in \{0,1\}^*\}$ gibt es keinen DPDA.
- NPDAs können also mehr als DPDAs.

Kellerautomaten - Beispiel 2

Ein NPDA für $L = \{ww^R | w \in \{0,1\}^*\}$.

$$(Q = \{q_0, q_1, q_2\},\ \Sigma = \{0,1,\#\},\ \Gamma = \Sigma \cup \{Z_0\}),\ q_0,\ Z_0,\ \delta,\ \emptyset)$$

$\delta(q_0, 0, Z_0)$	=	$\{(q_1, 0Z_0)\}$	Phase 1
$\delta(q_0, 1, Z_0)$	=	$\{(q_1, 1Z_0)\}$	
$\delta(q_1, 0, 0)$	=	$\{(q_1, 00), (q_2, \epsilon)\}$	
$\delta(q_1, 0, 1)$	=	$\{(q_1, 01)\}$	
$\delta(q_1, 1, 0)$	=	$\{(q_1, 10)\}$	
$\delta(q_1, 1, 1)$	=	$\{(q_1, 11), (q_2, \epsilon)\}$	
$\delta(q_2, 0, 0)$	=	$\{(q_2, \varepsilon)\}$	Phase 2
$\delta(q_2, 1, 1)$	=	$\{(q_2, \varepsilon)\}$	
$\delta(q_2, \varepsilon, Z_0)$	=	$\{(q_2, \varepsilon)\}$	Phase 3

Satz:
Zu einem PDA, der eine Sprache L durch einen akzeptierenden Endzustand akzeptiert, kann ein PDA konstruiert werden, der L mit leerem STACK akzeptiert.

Beweis - Beschreibung

- Sei $\mathcal{A}_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$ PDA, der L durch Übergang in einen Zustand aus F_1 akzeptiert.
- Wir konstruieren dazu einen PDA $\mathcal{A}_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_0^2, Z_0^2)$, der L durch leeren STACK akzeptiert.
- Sei q_E ein neuer Zustand
- Sei Z_0^2 ein neues Stack-Symbol

Idee der Konstruktion von $\mathcal{A}_2$.

- Lege zu Beginn Z_0^2 vor Z_0^1 auf den Stack, so dass der Stack nicht „versehentlich“ geleert werden kann.
- Dann Verfahre wie in $\mathcal{A}_1$.
- Wenn Zustand in F_1 erreicht wird: Gehe zu q_E und leere den Stack

Beweis - Konstruktion

- $\mathcal{A}_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$ akzeptiert durch Endzustand
- $\mathcal{A}_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_0^2, Z_0^2)$, akzeptiert durch leeren Stack
- Sei q_E ein neuer Zustand
- Sei Z_0^2 ein neues Stack-Symbol

$$
\begin{aligned}
Q_2 &= Q_1 \cup \{q_0^2, q_E\} \\
\Gamma_2 &= \Gamma_1 \cup \{Z_0^2\} \\
\delta_2(q_0^2, \varepsilon, Z_0^2) &= \{(q_0^1, Z_0^1 Z_0^2)\} \\
\delta_2(q, a, Z) &= \delta_1(q, a, Z) \text{ für } q \in Q_1,\ a \neq \varepsilon,\ Z \in \Gamma_1 \\
&\qquad\qquad q \in Q_1 \setminus F_1,\ a = \varepsilon,\ Z \in \Gamma_1 \\
\delta_2(q, \varepsilon, Z) &= \delta_1(q, \varepsilon, Z) \cup \{(q_E, \varepsilon)\} \text{ für } q \in F_1,\ Z \in \Gamma_2 \\
\delta_2(q_E, \varepsilon, Z) &= \{(q_E, \varepsilon)\} \text{ für } Z \in \Gamma_2 \\
\delta(\cdot) &= \emptyset \text{ sonst}
\end{aligned}
$$

Satz:
Zu einem PDA, der eine Sprache L mit leerem STACK akzeptiert, kann ein PDA konstruiert werden, der L durch einen akzeptierenden Endzustand akzeptiert.

Beweis - Beschreibung

- Sei $\mathcal{A}_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_0^1, Z_0^1)$, ein PDA der $w \in L$ mit leerem STACK akzeptiert
- Wir konstruieren dazu einen PDA $\mathcal{A}_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_0^2, Z_0^2, F_2)$, der genau die $w \in L$ durch Übergang in einen Zustand $q \in F_2$ akzeptiert.
- Sei Z_0^2 ein neues Stack-Symbol
- Sei q_F ein neuer (End-)Zustand
- Sei q_0^2 ein neuer (Anfangs-)Zustand

Idee der Konstruktion von $\mathcal{A}_2$.

- Lege zu Beginn Z_0^2 vor Z_0^1 auf den Stack, und lösche Z_0^2 nur, wenn die Abarbeitung von $\mathcal{A}_1$ durch leeren Stack akzeptiert hätte.
- Gehe in Endzustand q_F, wenn $\mathcal{A}_1$ durch leeren Stack akzeptiert hätte.

Beweis - Konstruktion

- $\mathcal{A}_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$ akzeptiert durch leeren Stack
- $\mathcal{A}_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_0^2, Z_0^2)$, akzeptiert durch Endzustand
- q_0^2 neuer Anfangszustand
- q_F neuer (End-)Zustand
- Z_0^2 ein neues Stack-Symbol

$$
\begin{aligned}
Q_2 &= Q_1 \cup \{q_0^2, q_F\}, \\
F_2 &= \{q_F\} \\
\Gamma_2 &= \Gamma_1 \cup \{Z_0^2\}
\end{aligned}
$$

$$
\begin{aligned}
\delta_2(q_0^2, a, X) &= \begin{cases} \{q_0^1, Z_0^1 Z_0^2\} & \text{falls } a = \varepsilon \text{ und } X = Z_0^2 \\ \emptyset & \text{sonst} \end{cases} \\
\delta_2(q, a, Z) &= \delta_1(q, a, Z), \text{ falls } q \in Q_1, a \in \Sigma \cup \{\varepsilon\} \text{ und } Z \in \Gamma_1 \\
\delta_2(q, \varepsilon, Z_0^2) &= \{(q_F, \varepsilon)\} \text{ für } q \in Q_1.
\end{aligned}
$$

Satz:
Für eine Grammatik G in Greibach-Normalform kann ein PDA konstruiert werden, der $L(G)$ mit leerem STACK akzeptiert.

Beweis

- Sei $G = (\Sigma, V, S, R)$ eine Grammatik in Greibach Normalform
- Konstruiere gewünschten Automaten $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

$$
\begin{aligned}
Q &:= \{q_0\} \\
\Gamma &:= V \\
Z_0 &:= S \\
\delta(q_0, a, A) &:= \{(q_0, \alpha) | (A \to a\alpha) \in R\}
\end{aligned}
$$

Per Induktion über die Länge i einer Ableitung beweisen wir:

- $S \overset{*}{\to} w_1 \dots w_i A_1 \dots A_m \quad \Leftrightarrow \quad \mathcal{A}$ kann beim Lesen von $w_1 \dots w_i$ den STACK-Inhalt $A_1 \dots A_m$ erzeugen. Möglicherweise ist $A_1 \dots A_m = \epsilon$.

Daraus folgt:

- $\mathcal{A}$ erkennt $w_1 \dots w_n$ mit leerem STACK $\Leftrightarrow S \overset{*}{\to} w_1 \dots w_n$ in G

Beweis

$$\begin{aligned} Q &:= \{q_0\} \quad \Gamma := V \quad Z_0 := S \\ \delta(q_0, a, A) &:= \{(q_0, \alpha) | (A \to a\alpha) \in R\} \end{aligned}$$

Induktionsanfang ist mit $i = 0$ trivialerweise erfüllt.
Induktionsschritt:
Sei $i \geq 1$ und „$\overset{j}{\to}$“ stehe für eine Ableitung der Länge j. Dann gilt

$$S \overset{i}{\to} w_1 \dots w_i A_1 \dots A_m \iff \begin{array}{l} \exists A' \in V, r \in \{1, \dots, m\} \text{ mit} \\ S \overset{i-1}{\to} w_1 \dots w_{i-1} A' A_r \dots A_m \\ \to w_1 \dots w_i A_1 \dots A_m. \end{array}$$

Mit Induktionsvoraussetzung ist dies äquivalent zu

$\exists A' \in V, r \in \{1, \dots, m\}$ so, dass

- $\mathcal{A}$ das Wort $w_1 \dots w_{i-1}$ lesen und dabei STACK-Inhalt $A' A_r \dots A_m$ erzeugen kann, und
- $A' \to w_i A_1 \dots A_{r-1}$ Regel von G ist.

Beweis

$$\begin{aligned} Q &:= \{q_0\} \quad \Gamma := V \quad Z_0 := S \\ \delta(q_0, a, A) &:= \{(q_0, \alpha) | (A \to a\alpha) \in R\} \end{aligned}$$

Induktionsanfang ist mit $i = 0$ trivialerweise erfüllt.
Induktionsschritt:
Sei $i \geq 1$ und „$\overset{j}{\to}$“ stehe für eine Ableitung der Länge j.
Mit Induktionsvoraussetzung ist dies äquivalent zu

$\exists A' \in V, r \in \{1, \ldots, m\}$ so, dass

- $\mathcal{A}$ das Wort $w_1 \ldots w_{i-1}$ lesen und dabei STACK-Inhalt $A' A_r \ldots A_m$ erzeugen kann, und
- $A' \to w_i A_1 \ldots A_{r-1}$ Regel von G ist.

Dies ist genau dann erfüllt, wenn $\mathcal{A}$ das Wort $w_1 \ldots w_i$ lesen und dabei den STACK-Inhalt $A_1 \ldots A_m$ erzeugen kann.

Satz:
Jede durch einen PDA (mit leerem STACK oder durch akzeptierende Endzustände) akzeptierte Sprache ist kontextfrei.

Beweis

- Sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ PDA, der $L_{\mathcal{A}}$ durch leeren STACK akzeptiert.
- Wir geben eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ mit $L_{\mathcal{A}} = L(G)$ an.

Die Konstruktion von G heißt **Tripelkonstruktion**.

- Setze $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- Sei S Startsymbol.

Ziel: Aus $[q, X, p]$ sollen genau die $w \in \Sigma^*$ ableitbar sein, für die es eine Abarbeitung von $\mathcal{A}$ gibt,

- die im Zustand q mit oberstem STACK-Symbol X beginnt und
- nach Lesen von w im Zustand p mit leerem STACK endet.

Beweis

- Sei $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ PDA, der $L_{\mathcal{A}}$ durch leeren STACK akzeptiert.
- Wir geben eine kontextfreie Grammatik $G = (\Sigma, V, S, R)$ mit $L_{\mathcal{A}} = L(G)$ an.

Die Konstruktion von G heißt **Tripelkonstruktion**.

- Setze $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- Sei S Startsymbol.

Die Regelmenge R ist gegeben durch

- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$ für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$, falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Beweis

Für eine Folge von Konfigurationen (q, w, X) nach (p, w', Y) schreiben wir auch

$$(q, w, X) \overset{*}{\vdash} (p, w', Y)$$

beziehungsweise

$$(q, w, X) \overset{k}{\vdash} (p, w', Y)$$

für eine Folge von genau k Konfigurationen.

Beweis

Wir werden per Induktion beweisen, dass für alle $p, q \in Q$, $X \in \Gamma$ und $w \in L$ gilt:

$$[q, X, p] \overset{*}{\to} w \text{ in } G \iff (q, w, X) \overset{*}{\vdash} (p, \epsilon, \epsilon)$$

Aus dieser Behauptung folgt dann

$$\begin{aligned} w \in L_{\mathcal{A}} \iff & \exists p \in Q \text{ mit } (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon), \text{ wobei} \\ & (q_0, w, Z_0) \text{ Anfangskonfiguration von } \mathcal{A} \text{ ist} \\ \iff & \exists p \in Q \text{ mit } [q_0, Z_0, p] \overset{*}{\to} w \\ \iff & \exists p \in Q \text{ mit } S \to [q_0, Z_0, p] \overset{*}{\to} w \\ \iff & w \in L(G) \end{aligned}$$

Beweis

Richtung

$$[q, X, p] \overset{*}{\to} w \text{ in } G \implies (q, w, X) \overset{*}{\vdash} (p, \epsilon, \epsilon)$$

Beschreibung:

- Induktion über die Länge k einer Ableitung $[q, X, p] \overset{k}{\to} w$ in G

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Induktionsanfang:

- Für $k = 1$ gilt, dass $[q, X, p] \to w$ eine Regel in G ist.
- Also ist $(p, \varepsilon) \in \delta(q, w, X)$ und $|w| \leq 1$.
- Also gibt es die Abarbeitung $(q, w, X) \overset{1}{\vdash} (p, \varepsilon, \varepsilon)$ in $\mathcal{A}$.

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Induktionsschritt:

- Betrachte eine Ableitung $[q, X, p] \overset{k}{\to} w$.
- Schreibe diese als

$$[q, X, p] \to a[q_1, Y_1, q_2][q_2, Y_2, q_3] \dots [q_m, Y_m, q_{m+1}] \overset{k-1}{\to} w,$$

wobei $q_{m+1} = p$ und $w = aw_1 \dots w_m$, mit $w_i \in \Sigma^*$, $a \in \Sigma$ und $[q_j, Y_j, q_{j+1}] \overset{k'}{\to} w_j$ mit $k' \leq k - 1$ für alle $1 \leq j \leq m$.

Beweis

- Betrachte eine Ableitung $[q, X, p] \xrightarrow{k} w$.
- Schreibe diese als

$$[q, X, p] \to a[q_1, Y_1, q_2][q_2, Y_2, q_3] \dots [q_m, Y_m, q_{m+1}] \xrightarrow{k-1} w,$$

wobei $q_{m+1} = p$ und $w = aw_1 \dots w_m$, mit $w_i \in \Sigma^*$, $a \in \Sigma$ und $[q_j, Y_j, q_{j+1}] \xrightarrow{k'} w_j$ mit $k' \leq k - 1$ für alle $1 \leq j \leq m$.

Beweis

- Betrachte eine Ableitung $[q, X, p] \xrightarrow{k} w$.
- Schreibe diese als

$$[q, X, p] \to a[q_1, Y_1, q_2][q_2, Y_2, q_3] \dots [q_m, Y_m, q_{m+1}] \overset{k-1}{\to} w,$$

wobei $q_{m+1} = p$ und $w = aw_1 \dots w_m$, mit $w_i \in \Sigma^*$, $a \in \Sigma$ und $[q_j, Y_j, q_{j+1}] \xrightarrow{k'} w_j$ mit $k' \leq k - 1$ für alle $1 \leq j \leq m$.

- Induktionsvoraussetzung: $(q_j, w_j, Y_j) \overset{*}{\vdash} (q_{j+1}, \varepsilon, \varepsilon)$ für alle $1 \leq j \leq m$.
- Also $(q_j, w_j, Y_j \dots Y_m) \overset{*}{\vdash} (q_{j+1}, \varepsilon, Y_{j+1} \dots Y_m)$ für alle $1 \leq j \leq m$.
- Damit

$$\begin{aligned}
(q, w, X) &\vdash (q_1, w_1 \dots w_m, Y_1 \dots Y_m) \\
&\overset{*}{\vdash} (q_2, w_2 \dots w_m, Y_2 \dots Y_m) \\
&\overset{*}{\vdash} (q_3, w_3 \dots w_m, Y_3 \dots Y_m) \\
&\overset{*}{\vdash} \dots \overset{*}{\vdash} (q_m, w_m, Y_m) \overset{*}{\vdash} (q_{m+1}, \varepsilon, \varepsilon) = (p, \varepsilon, \varepsilon)
\end{aligned}$$

Beweis

Richtung

$$[q, X, p] \overset{*}{\rightarrow} w \text{ in } G \quad \Longleftarrow \quad (q, w, X) \overset{*}{\vdash} (p, \epsilon, \epsilon)$$

Beschreibung:

- Induktion über die Länge k einer Abarbeitung $(q, w, X) \overset{k}{\vdash} (p, \epsilon, \epsilon)$

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Induktionsanfang:

- Für $k = 1$ folgt aus $(q, w, X) \vdash (p, \varepsilon, \varepsilon)$, dass
  - $w \in \Sigma \cup \{\varepsilon\}$ und
  - $(p, \varepsilon) \in \delta(q, w, X)$.
- Dann ist $[q, X, p] \to w$ eine Regel von G.

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Induktionsschritt:

- Betrachte eine Abarbeitung $(q, X, w) \overset{k}{\vdash} (p, \varepsilon, \varepsilon)$
- Zerlege $w = aw'$ wobei
  - $a = \varepsilon$, falls der erste Schritt von $\mathcal{A}$ ein ε-Übergang ist
  - $a \in \Sigma$, also der erste Buchstabe von w, sonst.
- Sei $(q_1, w', Y_1 \dots Y_m)$ die Konfiguration von $\mathcal{A}$ nach dem 1. Schritt.
- Dann gilt

$$(q, aw', X) \vdash (q_1, w', Y_1 \dots Y_m) \overset{k'}{\vdash} (p, \varepsilon, \varepsilon)$$

mit $k' \leq k - 1$.

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Sei

$$w' = w_1 \dots w_m \text{ Zerlegung von } w \text{ mit } w_j \in \Sigma^*$$

so, dass $\mathcal{A}$ startend mit der Konfiguration

$$(q_1, w', Y_1 \dots Y_m)$$

bei der betrachteten Abarbeitung gerade nach dem Lesen von $w_1 \dots w_j$ zum ersten Mal den STACK-Inhalt $Y_{j+1} \dots Y_m$ erzeugt. Sei q_{j+1} der zu diesem Zeitpunkt erreichte Zustand.

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Dann gilt: $q_{m+1} = p$ und

$$(q_j, w_j \dots w_m, Y_j \dots Y_m) \overset{k'}{\vdash} (q_{j+1}, w_{j+1} \dots w_m, Y_{j+1} \dots Y_m),$$

$k' \leq k - 1$, und während der gesamten Abarbeitung liegt $Y_{j+1} \dots Y_m$ ungelesen auf dem STACK.
Also gilt auch

$$(q_j, w_j, Y_j) \overset{k'}{\vdash} (q_{j+1}, \varepsilon, \varepsilon).$$

Beweis

- $V := \{[q, X, p] | p, q \in Q,\ X \in \Gamma\} \cup \{S\}$.
- $S \to [q_0, Z_0, q]$ für alle $q \in Q$
- $[q, X, q_{m+1}] \to a[q_1, Y_1, q_2] \dots [q_m, Y_m, q_{m+1}]$
  für alle Möglichkeiten $q_2, \dots, q_{m+1} \in Q$,
  falls $(q_1, Y_1 \dots Y_m) \in \delta(q, a, X)$.

Also gilt auch

$$(q_j, w_j, Y_j) \overset{k'}{\vdash} (q_{j+1}, \varepsilon, \varepsilon).$$

Nach Induktionsvoraussetzung folgt daraus, dass $[q_j, Y_j, q_{j+1}] \overset{*}{\to} w_j$ in G existiert. Damit erhalten wir, dass auch

$$[q, X, p] \to a[q_1, Y_1, q_2][q_2, Y_2, q_3] \dots [q_m, Y_m, q_{m+1}] \overset{*}{\to} aw_1 \dots w_m = w$$

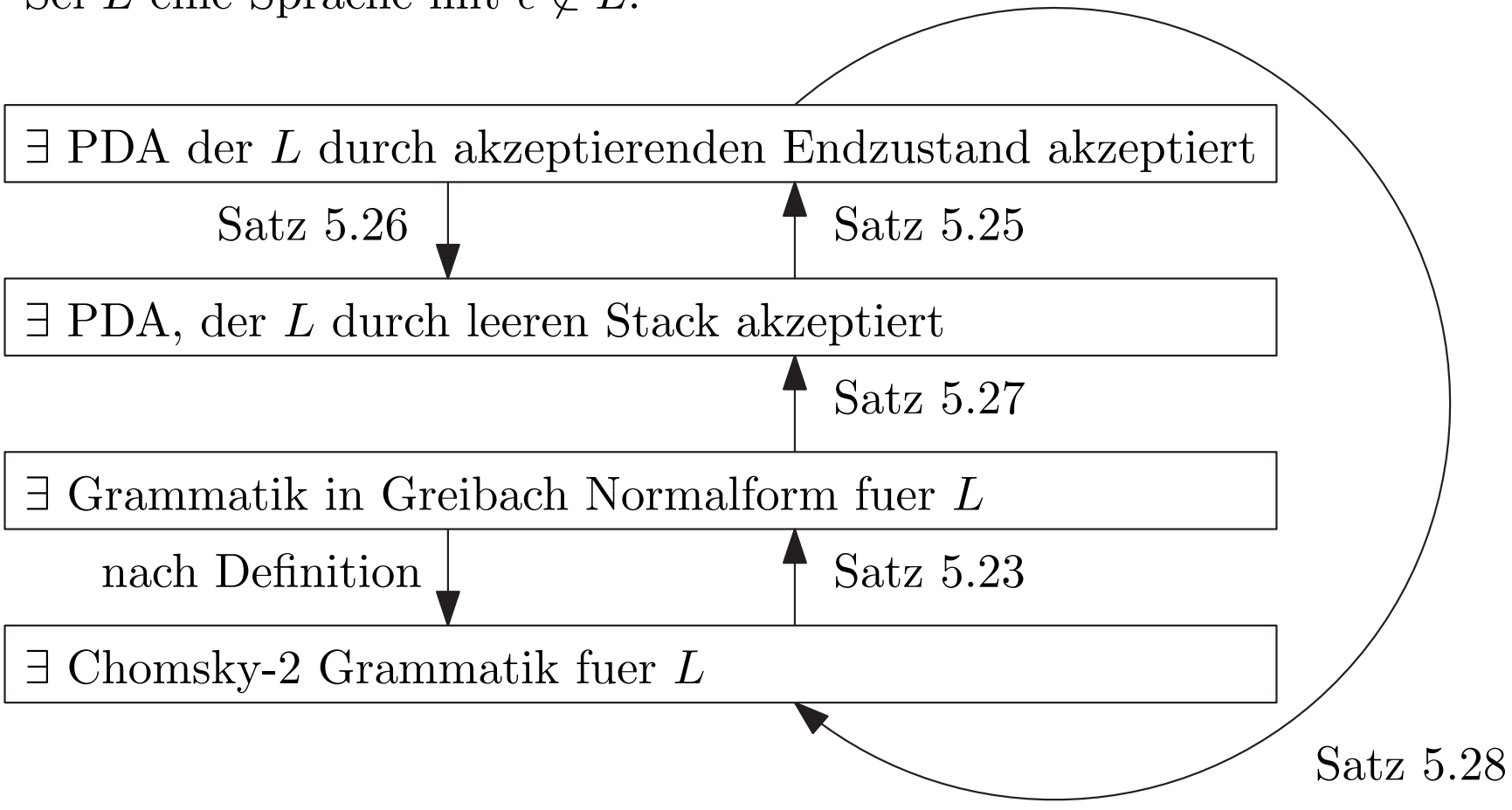
in G existiert.

Korollar

Die Klasse der von nichtdeterministischen Kellerautomaten akzeptierten Sprachen ist gleich der Klasse der kontextfreien Sprachen.

Übersicht Chomsky-2

Sei L eine Sprache mit $\epsilon \notin L$.



Exkurs

Wofür braucht man eigentlich Grammatiken und Berechnungsmodelle wie endliche Automaten oder Turingmaschinen?

- Die Chomsky-Hierarchie wurde von dem Linguisten Noam Chomsky entworfen. Ursprünglich war sie als Mittel zur Beschreibung natürlicher Sprachen gedacht (hat sich nicht erfüllt).
- Grammatiken und Automaten sind fundamental für die Beschreibung von Progammiersprachen.
- XML basiert auf sogenannten Dokumenttypdefinitionen (DTD). Diese sind kontextfreie Grammatiken.

Zwischenfazit zu kontextfreien Grammatiken

- Es kann in polynomialer Laufzeit entschieden werden, ob zu einer kontextfreien Grammatik G die Sprache $L(G)$ leer bzw. endlich ist.
- Das Wortproblem für kontextfreie Grammatiken ist in polynomialer Laufzeit entscheidbar.
- Für kontextfreie Grammatiken G, G_1 und G_2 sind die Sprachen
  - $L(G)^*$,
  - $L(G_1) \cup L(G_2)$ und
  - $L(G_1) \cdot L_(G_2)$

  kontextfrei.

Satz:
Das Problem für kontextfreie Grammatiken G_1 und G_2 zu entscheiden, ob $L(G_1) \cap L(G_2) = \emptyset$ ist, ist nicht entscheidbar.

Satz:
Das Problem für kontextfreie Grammatiken G_1 und G_2 zu entscheiden, ob $L(G_1) \cap L(G_2) = \emptyset$ ist, ist nicht entscheidbar.

Beweisskizze:

- Wir beweisen, dass aus der Entscheidbarkeit von $L(G_1) \cap L(G_2) = \emptyset$ die Entscheidbarkeit des Post'schen Korrespondenzproblem (PKP) folgt.
- Dies ist ein Widerspruch zur Nichtentscheidbarkeit des PKP.
- Wir geben für jede PKP-Instanz K kontextfreie Grammatiken G_1 und G_2 an, so dass es ein Wort $w \in L(G_1) \cap L(G_2)$ genau dann gibt, wenn es eine Lösung für K gibt.

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblems
Gegeben ist endliche Folge von Wortpaaren

$$K = ((x_1, y_1), \ldots, (x_k, y_k))$$

über einem endlichen Alphabet Σ. Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \ldots, i_\ell \in \{1, \ldots, k\}$ gibt, so dass $x_{i_1} \ldots x_{i_\ell} = y_{i_1} \ldots y_{i_\ell}$ gilt.

Beweis

- Gegeben sei PKP-Instanz $K = ((x_1, y_1), \ldots, (x_k, y_k))$
- Es sei $\Sigma = \{x_1, \ldots, x_l, y_1, \ldots, y_k, a_1, \ldots, a_k\}$ mit neuen Symbolen $a_1, \ldots, a_k$.
- Es sei $G_1 = (\Sigma, V_1 = \{S_1\}, S_1, R_1)$ mit Regeln

$$S_1 \to a_i x_i \text{ und } S_1 \to a_i S_1 x_i \text{ für alle } 1 \leq i \leq k;$$

- Es sei $G_2 = (\Sigma, V_2 = \{S_2\}, S_2, R_2)$ mit Regeln

$$S_2 \to a_i y_i \text{ und } S_2 \to a_i S_2 y_i \text{ für alle } 1 \leq i \leq k.$$

Dann gilt

$$\begin{aligned} L(G_1) &= \{a_{i_n} \cdots a_{i_1} x_{i_1} \cdots x_{i_n} \mid n \in \mathbb{N},\ 1 \leq i_j \leq k\} \\ L(G_2) &= \{a_{i_n} \cdots a_{i_1} y_{i_1} \cdots y_{i_n} \mid n \in \mathbb{N},\ 1 \leq i_j \leq k\}\ . \end{aligned}$$

Beweis

$$
\begin{aligned}
K &= ((x_1, y_1), \ldots, (x_k, y_k)) \\
L(G_1) &= \{a_{i_n} \cdots a_{i_1} x_{i_1} \cdots x_{i_n} \mid n \in \mathbb{N},\ 1 \leq i_j \leq k\} \\
L(G_2) &= \{a_{i_n} \cdots a_{i_1} y_{i_1} \cdots y_{i_n} \mid n \in \mathbb{N},\ 1 \leq i_j \leq k\}\ .
\end{aligned}
$$

Es folgt

$$
\begin{aligned}
K \text{ hat Lösung} &\Leftrightarrow \exists i_1, \ldots, i_n \text{ mit } x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n} \\
&\Leftrightarrow \exists i_1, \ldots, i_n \text{ mit } a_{i_n} \cdots a_{i_1} x_{i_1} \cdots x_{i_n} = a_{i_n} \cdots a_{i_1} y_{i_1} \cdots y_{i_n} \\
&\Leftrightarrow \exists w \in L(G_1) \cap L(G_2) \\
&\Leftrightarrow L(G_1) \cap L(G_2) \neq \emptyset
\end{aligned}
$$

Eine Grammatik G ist eindeutig, wenn es für jedes $w \in L(G)$ genau einen Syntaxbaum gibt.

Satz:
Das Problem, für eine kontextfreie Grammatik G zu entscheiden, ob sie eindeutig ist, ist nicht entscheidbar.

Beweisskizze

- Annahme: Es sei entscheidbar, ob eine kontextfreie Grammatik eindeutig ist.
- Dann könnten wir das PKP entscheiden.
- Dies ist ein Widerspruch.

Beweis

- Gegeben sei PKP-Instanz $K = ((x_1, y_1), \ldots, (x_k, y_k))$.
- Seien $G_1 = (\Sigma, V_1, S_1, R_1)$ und $G_2 = (\Sigma, V_2, S_2, R_2)$ wie im letzten Beweis.
- Wir konstruieren eine neue Grammatik $G = (\Sigma, V, S, R)$, die mehrdeutig ist, gdw, $L(G_1) \cap L(G_2) \neq \emptyset$:

$$\begin{aligned} V &= V_1 \cup V_2 \cup \{S\} \text{ wobei } S \text{ neues Startsymbol} \\ R &= R_1 \cup R_2 \cup \{S \to S_1 | S_2\} \end{aligned}$$

- Da G_1 und G_2 eindeutig sind, existiert $w \in L(G_1) \cap L(G_2)$ genau dann, wenn es in G Ableitungen $S \to S_1 \overset{*}{\to} w$ und $S \to S_2 \overset{*}{\to} w$ gibt, also G mehrdeutig ist.

Sprache der korrekten Rechenwege

- Sei $\mathcal{M} = (Q, \Sigma, \Gamma, \sqcup, q_0, \delta, F)$ eine TM.
- Eine Berechnung von $\mathcal{M}$ kann durch die Folge der durchlaufenen

  Konfigurationen $\alpha q \beta$ mit $\alpha, \beta \in \Gamma^*$ und $q \in Q$

  beschrieben werden.

- $\alpha q \beta$ bedeutet, dass
  - auf dem Band das Wort $\alpha\beta$, umgeben von Blanksymbolen, steht,
  - die Turingmaschine im Zustand q ist
  - und der Lese-/Schreibkopf auf die Stelle des Bandes, an der das erste Symbol von β steht, zeigt.

- Wenn $w_1, w_2, \ldots, w_n$ die Abfolge der Konfigurationen einer Berechnung von $\mathcal{M}$ ist, so kann dieser Rechenweg durch das Wort $w_1 \# w_2 \# \ldots \# w_n \#$, mit $\# \notin \Gamma$ Trennsymbol, kodiert werden.

Sprache der korrekten Rechenwege

- Allerdings lässt sich die Sprache aller Wörter, die in dieser Weise die korrekten Rechenwege einer TM kodieren, nicht unbedingt durch kontextfreie Grammatiken beschreiben.
- Daher wird ein „Trick“ angewendet und jede zweite Konfiguration gespiegelt kodiert.

Sprache der korrekten Rechenwege

Die **Sprache $B_{\mathcal{M}}$ der korrekten Rechenwege einer TM $\mathcal{M}$** besteht aus allen Worten

$$w_1 \# w_2^R \# w_3 \# w_4^R \ldots w_n^R \#, \text{ falls } n \text{ gerade und}$$

$$w_1 \# w_2^R \# w_3 \# w_4^R \ldots w_n \#, \text{ falls } n \text{ ungerade,}$$

wobei

- die w_i, $1 \leq i \leq n$, Konfigurationen von $\mathcal{M}$ sind,
- w_1 eine Anfangskonfiguration,
- w_n eine akzeptierende Konfiguration und
- für alle $1 \leq i \leq n-1$ die Konfiguration w_{i+1} die direkte Nachfolgekonfiguration von w_i bei einer korrekten Berechnung von $\mathcal{M}$

ist.

Lemma

Für alle Turingmaschinen $\mathcal{M}$ ist $B_{\mathcal{M}}$ der Durchschnitt zweier Sprachen

- $L_1 = L(G_1)$
- $L_2 = L(G_2)$,

wobei G_1 und G_2 kontextfreie Grammatiken sind.

Beweis

Wir konstruieren L_1 und L_2 aus den Sprachen

$$\begin{aligned} L &:= \{u\#v^R \mid v \text{ ist direkte Nachfolgekonfiguration von } u \text{ für } \mathcal{M}\} \\ L' &:= \{v^R\#u \mid u \text{ ist direkte Nachfolgekonfiguration von } v \text{ für } \mathcal{M}\} \end{aligned}$$

Falls L und L' kontextfrei sind, so sind auch

$$\begin{aligned} L_1 &:= (L\{\#\})^*(\{\varepsilon\} \cup \Gamma^* F \Gamma^* \{\#\}) \\ L_2 &:= \{q_0\}\Sigma^*\{\#\}(L'\{\#\})^*(\{\varepsilon\} \cup \Gamma^* F \Gamma^* \{\#\}) \end{aligned}$$

kontextfrei, wobei

- Γ Bandalphabet,
- Σ Eingabealphabet,
- q_0 Anfangszustand und
- F Endzustandsmenge

von $\mathcal{M}$.

Beweis

Offensichtlich haben alle Wörter aus L_1 die Form

$$w_1 \# w_2^R \# \ldots w_{2i-1} \# w_{2i}^R \# \text{ oder}$$

$$w_1 \# w_2^R \# \ldots w_{2i-1} \# w_{2i}^R \# w_{2i+1} \#$$

mit

- w_j Konfiguration von $\mathcal{M}$
- w_{2j} direkte Nachfolgekonfiguration von w_{2j-1}

für alle $1 \leq j \leq i$ und w_{2i+1} akzeptierende Konfiguration, falls vorhanden.

Beweis

Analog haben alle Wörter aus L_2 die Form

$$w_1 \# w_2^R \# \ldots w_{2i-1} \# w_{2i}^R \# \text{ oder}$$

$$w_1 \# w_2^R \# \ldots w_{2i-2}^R \# w_{2i-1} \#$$

mit

- w_j Konfiguration von $\mathcal{M}$
- w_1 Anfangskonfiguration
- w_{2j+1} direkte Nachfolgekonfiguration von w_{2j}

für alle $1 \leq j \leq i-1$ und w_{2i} akzeptierende Konfiguration, falls vorhanden.

Dann ist $B_{\mathcal{M}} = L_1 \cap L_2$.

Beweis

Wir geben nun eine kontextfreie Grammatik G für L an mit Startvariable S und zusätzlicher Variable A.

G enthalte folgende Regeln:

(i) alle Regeln $S \to aSa$, $a \in \Gamma \setminus \{\sqcup\}$;

(ii) für alle Übergänge $\delta(q, a) = (q', b, R)$ von $\mathcal{M}$ die Regeln $S \to qaAq'b$;

(iii) für alle Übergänge $\delta(q, a) = (q', b, L)$ von $\mathcal{M}$ die Regeln $S \to xqaAbxq'$, wobei x Symbol links von a beim Lesen von a im Zustand q;

(iv) für alle Übergänge $\delta(q, a) = (q', b, N)$ von $\mathcal{M}$ die Regeln $S \to qaAbq'$;

(v) für alle $a \in \Gamma$ die Regeln $A \to aAa$;

(vi) die Regel $A \to \#$.

Beweis

- Analog kann eine kontextfreie Grammatik G' für L' angegeben werden.
- Es ist leicht zu zeigen, dass $L(G) = L$ und $L(G') = L'$ ist.
- Damit ist die Behauptung bewiesen.

Bemerkung

Falls $\mathcal{M}$ in jeder Berechnung nur höchstens einen Rechenschritt ausführt, ist $B_{\mathcal{M}}$ sogar selbst kontextfrei.

Lemma

Sei $\mathcal{M}$ eine TM, die auf jeder Eingabe mindestens zwei Rechenschritte ausführt. Dann ist die Sprache $B_{\mathcal{M}}$ genau dann kontextfrei, wenn $L(\mathcal{M})$ endlich ist.

Beweis

$B_{\mathcal{M}}$ ist kontextfrei $\Leftarrow$ $L(\mathcal{M})$ endlich ist

- Sei $L(\mathcal{M})$ endlich
- Zu jeder Eingabe aus $L(\mathcal{M})$ gibt es genau eine akzeptierende Berechnung.
- Damit ist $B_{\mathcal{M}}$ auch endlich,
- Jede endliche Sprache ist regulär, also auch kontextfrei.

Beweis

$B_{\mathcal{M}}$ ist kontextfrei $\Rightarrow$ $L(\mathcal{M})$ endlich ist

- Angenommen $L(\mathcal{M})$ sei unendlich und $B_{\mathcal{M}}$ wäre kontextfrei.
- Da $L(\mathcal{M})$ unendlich ist, gibt es zu der Konstanten n aus Ogden's Lemma ein $w \in B_{\mathcal{M}}$ mit $w = w_1 \# w_2^R \# \ldots$ und $|w_2^R| \geq n$.
- Wenn alle Symbole aus $\# w_2^R \#$ markiert werden, muss es eine Zerlegung $uvwxy$ von w geben, sodass vx mindestens einen und vwx höchstens n markierte Buchstaben enthält und $uv^i wx^i y \in B_{\mathcal{M}}$ für alle $i \geq 0$.
- Da $\mathcal{M}$ mindestens zwei Berechnungsschritte ausführt, existieren die Konfigurationen w_1, w_2 und w_3.
- Entsprechend der Zerlegung von w enthalten $\# w_2^R \#$ und vx mindestens einen gemeinsamen Buchstaben, und nur eines der Worte w_1 und w_3 hat ebenfalls gemeinsame Buchstaben mit vx.

Beweis

$B_{\mathcal{M}}$ ist kontextfrei $\Rightarrow$ $L(\mathcal{M})$ endlich ist

- Da $\mathcal{M}$ mindestens zwei Berechnungsschritte ausführt, existieren die Konfigurationen w_1, w_2 und w_3.
- Entsprechend der Zerlegung von w enthalten $\#w_2^R\#$ und vx mindestens einen gemeinsamen Buchstaben, und nur eines der Worte w_1 und w_3 hat ebenfalls gemeinsame Buchstaben mit vx.
- Wenn w_1 keinen gemeinsamen Buchstaben mit vx hat, ist $uv^2wx^2y \notin B_{\mathcal{M}}$, da die Berechnung für die Anfangskonfiguration w_1 eindeutig ist.
- Aus demselben Grund ist $uv^2wx^2y \notin B_{\mathcal{M}}$, falls $w_1\#$ Präfix von uv^2 ist.
- Falls v ein Teilwort von w_1 wäre, müsste x ein Teilwort von w_2^R sein, damit für großes i das Wort $uv^iwx^iy \in B_{\mathcal{M}}$ ist, da zwei aufeinanderfolgende Konfigurationen etwa gleich lang sind.
- Dann wäre aber w_3 als Nachfolgekonfiguration zu kurz, uv^iwx^iy also keine Kodierung eines korrekten Rechenweges von $\mathcal{M}$.
- Dies ist ein Widerspruch.

Lemma
Für jede TM $\mathcal{M}$ ist das Komplement $(B_{\mathcal{M}})^c$ von $B_{\mathcal{M}}$ kontextfrei.

Beweisskizze

Wir geben L_1, L_2 und L_3 kontextfrei an, dass $(B_{\mathcal{M}})^c = L_1 \cup L_2 \cup L_3$.
Wir nennen

$$w := \begin{cases} w_1 \# w_2^R \# \dots w_n^R \# & \text{für gerades } n \\ w_1 \# w_2^R \# \dots w_n \# & \text{für ungerades } n \end{cases}$$

wohlgeformt, wobei w_i Konfiguration für alle i, $1 \leq i \leq n$, w_1 Anfangskonfiguration und w_n akzeptierende Konfiguration.

$$L_1 := \{w \mid w \text{ nicht wohlgeformt}\}$$

$$L_2 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ ungerade, sodass } w_i \# w_{i+1}^R \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

$$L_3 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ gerade, sodass } w_i^R \# w_{i+1} \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

Beweisskizze

$$L_1 := \{w \mid w \text{ nicht wohlgeformt}\}$$

$$L_2 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ ungerade, sodass } w_i \# w_{i+1}^R \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

$$L_3 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ gerade, sodass } w_i^R \# w_{i+1} \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

Es bleibt zu zeigen, dass L_1, L_2 und L_3 kontextfrei sind.

Beweisskizze

$$L_1 := \{w \mid w \text{ nicht wohlgeformt}\}$$

$$L_2 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ ungerade, sodass } w_i \# w_{i+1}^R \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

$$L_3 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ gerade, sodass } w_i^R \# w_{i+1} \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

L_1 ist das Komplement des regulären Ausdrucks

$$q_0 \Sigma^* \# (\Gamma^* Q \Gamma^* \#)^* \Gamma^* F \Gamma^* \#$$

und ist damit sogar regulär.

Beweisskizze

$$L_1 := \{w \mid w \text{ nicht wohlgeformt}\}$$

$$L_2 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ ungerade, sodass } w_i \# w_{i+1}^R \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

$$L_3 := \left\{ w \;\middle|\; \begin{array}{l} w \text{ wohlgeformt und es gibt } i \text{ gerade, sodass } w_i^R \# w_{i+1} \\ \text{Teilwort von } w, \text{ für das } w_{i+1} \text{ nicht Nachfolgekonfiguration} \\ \text{von } w_i \text{ ist.} \end{array} \right\}$$

Für L_1 und L_2 können kontextfreie Grammatiken angegeben werden, wobei die Regeln ähnlich wie im Beweis zum vorletzten Lemma hergeleitet werden.

Satz:
Seien G, G_1, G_2 kontextfreie Grammatiken über Σ. Es ist nicht entscheidbar, ob

(i) $(L(G))^c$ kontextfrei ist,

(ii) $L(G_1) \cap L(G_2)$ kontextfrei ist,

(iii) $L(G) = \Sigma^*$,

(iv) $L(G_1) = L(G_2)$,

(v) $L(G_1) \subseteq L(G_2)$.

Beweis

(i) Es ist nicht entscheidbar, ob $(L(G))^c$ kontextfrei ist.

Beweis:
Angenommen, es gäbe eine TM $\mathcal{M}'$, die für G entscheidet, ob $(L(G))^c$ kontextfrei ist. Wir zeigen, dass es dann auch eine TM $\mathcal{M}''$ gäbe, die

$$L := \{\langle \mathcal{M} \rangle \,|\, L(\mathcal{M}) \text{ endlich}\}$$

entscheidet. Dies wäre ein Widerspruch zum Satz von Rice.

- O.B.d.A. enthalte L nur $\langle \mathcal{M} \rangle$, für die $\mathcal{M}$ mindestens zwei Rechenschritte ausführt.
- Die TM $\mathcal{M}''$ berechnet für jede Eingabe $\langle \mathcal{M} \rangle$ eine kontextfreie Grammatik $G_{\mathcal{M}}$ für $(B_{\mathcal{M}})^c$.
- Dann simuliert $\mathcal{M}''$ die TM $\mathcal{M}'$ auf der Eingabe $G_{\mathcal{M}}$ und entscheidet damit, ob $B_{\mathcal{M}}$ kontextfrei ist.
- Dies ist äquivalent dazu, dass $L(\mathcal{M})$ endlich ist.

Beweis

(ii) Es ist nicht entscheidbar, ob $L(G_1) \cap L(G_2)$ kontextfrei ist.

Beweis:

- Angenommen, es gäbe eine TM $\mathcal{M}'$, die für G_1 und G_2 entscheidet, ob $L(G_1) \cap L(G_2)$ kontextfrei ist.
- Wir betrachten eine TM $\mathcal{M}''$, die auf Eingaben $\langle \mathcal{M} \rangle$ kontextfreie Grammatiken $G_{1,\mathcal{M}}$ und $G_{2,\mathcal{M}}$ berechnet mit $L(G_{1,\mathcal{M}}) \cap L(G_{2,\mathcal{M}}) = B_{\mathcal{M}}$.
- Dann simuliert $\mathcal{M}''$ die TM $\mathcal{M}'$ auf den Eingaben $G_{1,\mathcal{M}}$ und $G_{2,\mathcal{M}}$ und entscheidet damit, ob $B_{\mathcal{M}}$ kontextfrei ist, also wieder ob $L(\mathcal{M})$ endlich ist.
- Dies ist wie in (i) ein Widerspruch zum Satz von Rice.

Beweis

(iii) Es ist nicht entscheidbar, ob $L(G) = \Sigma^*$.

Beweis:

- Angenommen, es gäbe eine TM $\mathcal{M}'$, die für G und Σ entscheidet, ob $L(G) = \Sigma^*$ ist.
- Wir betrachten eine TM $\mathcal{M}''$, die für $\langle \mathcal{M} \rangle$ entscheidet, ob $L(\mathcal{M}) = \emptyset$ ist.
- Dies wäre wieder ein Widerspruch zum Satz von Rice.
- $\mathcal{M}''$ berechnet für $\langle \mathcal{M} \rangle$eine kontextfreie Grammatik $G_{\mathcal{M}}$ für $(B_{\mathcal{M}})^c$.
- Dann simuliert $\mathcal{M}''$ die TM $\mathcal{M}'$ auf $G_{\mathcal{M}}$ und entscheidet, ob $(B_{\mathcal{M}})^c = (\Gamma \cup \{\#\} \cup Q)^*$ ist.
- Dies ist äquivalent dazu, dass $B_{\mathcal{M}} = \emptyset$ und damit $L(\mathcal{M}) = \emptyset$ ist.

Beweis

(iv) Es ist nicht entscheidbar, ob $L(G_1) = L(G_2)$.

Beweis:

- Die Entscheidung, ob $L(G_1) = \Sigma^*$ ist, ist ein Spezialfall von $L(G_1) = L(G_2)$.
- Damit ist $L(G_1) = L(G_2)$ ebenfalls unentscheidbar.

Beweis

(v) Es ist nicht entscheidbar, ob $L(G_1) \subseteq L(G_2)$.

Beweis:

- Die Entscheidung, ob $\Sigma^* = L(G_2)$ ist, ist ein Spezialfall von $L(G_1) \subseteq L(G_2)$.
- Damit ist auch $L(G_1) \subseteq L(G_2)$ unentscheidbar.

Satz:
Die Sprache $L = \{a^i b^j c^k | i = j \text{ oder } j = k\}$ ist inhärent mehrdeutig.

Beweisskizze

Zunächst stellen wir fest, dass

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

kontextfrei ist. Es gilt $L = L_1 \cup L_2$, wobei

$$L_1 := \{a^i b^i | i \geq 0\} \cdot \{c\}^*$$

$$L_2 := \{a\}^* \cdot \{b^i c^i | i \geq 0\}.$$

Der Schnitt

$$L_1 \cap L_2 = \{a^i b^i c^i | i \geq 0\}$$

ist jedoch nicht kontextfrei.

Beweisskizze

Zunächst stellen wir fest, dass

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

kontextfrei ist. Es gilt $L = L_1 \cup L_2$, wobei

$$L_1 := \{a^i b^i | i \geq 0\} \cdot \{c\}^*$$

$$L_2 := \{a\}^* \cdot \{b^i c^i | i \geq 0\}.$$

Der Schnitt

$$L_1 \cap L_2 = \{a^i b^i c^i | i \geq 0\}$$

ist jedoch nicht kontextfrei.

Beweisidee Mehrdeutigkeit von L:
Zeige für gewisse Wörter $w \in L_1 \cap L_2$, dass jede beliebige kontextfreie Grammatik G zu L zwei Syntaxbäume für w.

Beweisskizze

Zunächst stellen wir fest, dass

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

kontextfrei ist. Es gilt $L = L_1 \cup L_2$, wobei

$$L_1 := \{a^i b^i \mid i \geq 0\} \cdot \{c\}^*$$

$$L_2 := \{a\}^* \cdot \{b^i c^i \mid i \geq 0\}.$$

Der Schnitt

$$L_1 \cap L_2 = \{a^i b^i c^i \mid i \geq 0\}$$

ist jedoch nicht kontextfrei.

Ausgehend von $a^n b^n c^{n+n!}$ und $a^{n+n!} b^n c^n$ zeige mit Ogden's Lemma, dass es zu $a^{n+n!} b^{n+n!} c^{n+n!}$ zwei Syntaxbäume gibt.