$a^n b^n c^n$ ist kontextsensitiv

Beispiel 3.1.11 modifizieren: **$a^n b^n c^n$ kontextsensitiv**

$\Sigma = \{a, b, c\}$

$G = (\Sigma, V, P, X)$
$V = \{X, Y, Z\}$

$$\begin{array}{|lrl|} \hline P: & X & \to \varepsilon \\ & X & \to aXYZ \\ & ZY & \to YZ \\ & aY & \to ab \\ & bY & \to bb \\ & bZ & \to bc \\ & cZ & \to cc \\ \hline \end{array}$$

$G' = (\Sigma, V, P, S)$
$V = \{S, X, Y, Z\}$

$$\begin{array}{|lrl|} \hline P: & S & \to X \mid \varepsilon \\ & X & \to aXYZ \mid aYZ \\ & ZY & \to YZ \\ & aY & \to ab \\ & bY & \to bb \\ & bZ & \to bc \\ & cZ & \to cc \\ \hline \end{array}$$

erzeugen beide die Sprache $L = \{a^n b^n c^n : n \in \mathbb{N}\}$.

G' ist kontextsensitiv (nur harmlose ε-Produktion!).

kontextfreie Sprachen (Typ 2) $\to$ Abschnitt 3.3

- wichtige nächste Stufe nach regulär
- zulässige Produktionen bei Typ 2, $\varepsilon \notin L$: $X \to v$, $v \neq \varepsilon$

Verschärfung: Chomsky-Normalform: $X \to YZ$ und $X \to a$

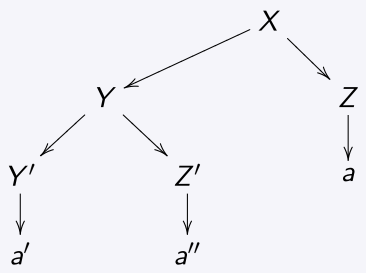
Satz: jede Typ 2 Grammatik ohne ε-Produktionen ist äquivalent zu Chomsky NF Grammatik

Beweisidee zur Transformation in Chomsky NF (Satz 3.3.2):

– ersetze a auf rechten Seiten durch neues Z_a
  neue Produktionen $Z_a \to a$
– eliminiere $X \to Y$ Produktionen (durch shortcuts)
– eliminiere $X \to Y_1 \ldots Y_k$ Produktionen für $k > 2$

Chomsky NF und binäre Bäume

Ableitungsschritt $X \to YZ$ binäre Verzweigung
$X \to a$ keine Verzweigung



Ableitungsbaum mit inneren *Knoten* für X in $X \to YZ$ Anwendungen

Lemma 3.3.3

in Chomsky NF Grammatik G:
$w \in L(G)$ hat Ableitung der Länge $2|w| - 1$.
Beweise induktiv über Länge ℓ der Ableitung von $w \in (V \cup \Sigma)^+$ $\boxed{|w|_V + 2|w|_\Sigma = \ell + 1}$

kontextfreie Sprachen: Abschlusseigenschaften

Abschluss unter Vereinigung, Konkatenation, Stern

(Satz 3.3.7)

Zu gegebenen Typ 2 Grammatiken für L_1, L_2, L finde explizit Typ 2 Grammatiken für $L_1 \cup L_2$, für $L_1 \cdot L_2$, bzw. für L^*

z.B.: seien $G^{(i)} = (\Sigma, V^{(i)}, P^{(i)}, S^{(i)})$ kontextfrei, $V^{(1)} \cap V^{(2)} = \emptyset$

$G = (\Sigma, V, P, S)$ mit $L(G) = L(G^{(1)}) \cdot L(G^{(2)})$:

$$\begin{array}{lll} V & := & V^{(1)} \cup V^{(2)} \cup \{S\} \qquad S \text{ neu} \\ P & := & \{S \to S^{(1)} S^{(2)}\} \cup P^{(1)} \cup P^{(2)} \end{array}$$

Bsp: $L_1 = \{a^n b^n : n \in \mathbb{N}\} \cdot \{c\}^*$, $L_2 = \{a\}^* \cdot \{b^m c^m : m \in \mathbb{N}\}$
kontextfrei, nicht jedoch $L_1 \cap L_2 = \{a^n b^n c^n : n \in \mathbb{N}\}$ (später)

Kein Abschluss unter Durchschnitt/Komplement: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

noch ein Pumping Lemma → Abschnitt 3.3.3

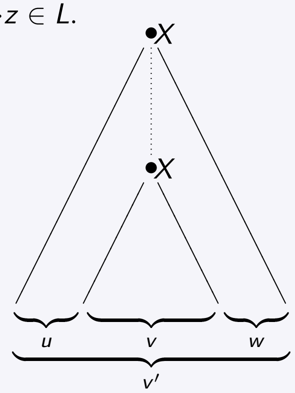
$L \subseteq \Sigma^*$ kontextfrei $\Rightarrow$

existiert $n \in \mathbb{N}$ sodass sich jedes $x \in L$ mit $|x| \geqslant n$ zerlegen lässt in $x = yuvwz$, $uw \neq \varepsilon$, $|uvw| \leqslant n$, und für alle $m \in \mathbb{N}$

$$y \cdot u^m \cdot v \cdot w^m \cdot z = y \cdot \underbrace{u \cdots u}_{m \text{ mal}} \cdot v \cdot \underbrace{w \cdots w}_{m \text{ mal}} \cdot z \in L.$$

Beweis (Satz 3.3.8):
$L = L(G)$, G in Chomsky NF, $n := 2^{|V|}$.
Für $x \in L(G), |x| \geqslant n$, hat jeder Ableitungsbaum zwei geschachtelte Vorkommen desselben X

Finde uvw wie in Skizze
Beispiel: $\{a^n b^n c^n : n \in \mathbb{N}\}$ nicht kontextfrei



Erinnerung: Wortprobleme als Entscheidungsprobleme

Wortproblem zu $L \subseteq \Sigma^*$:

Eingabe: $w \in \Sigma^*$ Entscheide, ob $w \in L$

Lösung durch Algorithmus A mit

$$w \xrightarrow{A} \begin{cases} \text{“ja”} & \text{falls } w \in L \quad \textit{akzeptieren} \\ \text{“nein”} & \text{falls } w \notin L \quad \textit{verwerfen} \end{cases}$$

definite Entscheidung
im Gegensatz zu: Akzeptieren wie bei NFA
Erzeugen/Ableiten wie in Grammatik

CYK Algorithmus → Abschnitt 3.3.4

effizienter Algorithmus für das kontextfreie Wortproblem

für G in Chomsky NF (Produktionen $X \to a$ und $X \to YZ$)
berechne zu $w = a_1 \dots a_n$ systematisch für alle Teilwörter $w_{i,j} = a_i \dots a_j$ $(1 \leqslant i \leqslant j \leqslant n)$

$$\boxed{V(i,j) := \{X \in V : X \to_G^* w_{i,j}\}}$$

dynamisches Programmieren

rekursive Auswertung für $i < j$: (mit wachsender Länge $j - i + 1$)

$X \to_G^* w_{i,j}$ gdw für ein k mit $i \leqslant k < j$ und ein $X \to YZ$ ist $Y \to_G^* w_{i,k}$ und $Z \to_G^* w_{k+1,j}$

Cocke, Younger, Kasami

CYK Algorithmus: Wortproblem in $\sim |w|^3$ Schritten entscheidbar.

das Wichtigste aus Kapitel 3

Grammatiken und Erzeugungsprozesse

Niveaus der **Chomsky-Hierarchie**

Normalform und **Pumping Lemma** für **kontextfreie Sprachen**

Kapitel 4: Berechnungsmodelle:
Turingmaschinen (DTM/NTM)
Kellerautomaten (PDA)
Endliche Automaten (DFA/NFA) ✓

Berechnungsmodelle

- prinzipielle Fragen:
  Was lässt sich berechnen? (z.B. Wortprobleme)
- qualitativ-quantitative Fragen:
  Wie schwer ist ein algorithmisches Problem?
  Komplexitätshierarchien? (z.B. in Chomsky-Hierarchie)

Algorithmus =
(Berechnungs-)Verfahren

nach **Al Chwarismi**
(Bagdad, um 800),
latinisiert zu Algoritmi

Kellerautomaten (PDA) → Abschnitt 4.1

PDA = NFA + Kellerspeicher (stack, push-down storage)

Konfiguration jeweils bestimmt durch
- Zustand
- Position in der Eingabe
- Kellerinhalt

erlaubte (nichtdeterministische) Übergänge abhängig von
- Zustand
- oberstem Kellersymbol
- nächstem Eingabesymbol

Übergang resultiert in
- Zustandswechsel
- (optional) Vorrücken in Eingabe
- pop und push im Keller:
  – Entfernen des obersten Kellersymbols (pop)
  – Einschreiben eines Wortes in Keller (push)

PDA $\mathcal{P} = (\Sigma, Q, q_0, \Delta, A, \Gamma, \#)$:

Σ	Eingabealphabet	Q	Zustandsmenge
Γ	Kelleralphabet	$q_0 \in Q$	Anfangszustand
$\# \in \Gamma$	Anfangs-Kellersymbol	$A \subseteq Q$	akzeptierende Zustände

endliche Übergangsrelation: $\Delta \subseteq Q \times \Gamma \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$

Konfigurationen: $C = (q, v, \alpha) \in Q \times \Sigma^* \times \Gamma^*$:
- $q \in Q$ aktueller Zustand,
- $v \in \Sigma^*$ Restabschnitt des Eingabewortes,
- $\alpha \in \Gamma^*$ aktueller Kellerinhalt.

Startkonfiguration auf Eingabe w: $C_0[w] = (q_0, w, \#)$

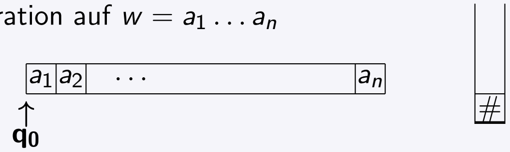
Nachfolgekonfigurationen zu $C = (q, v, \alpha)$, $\alpha = \gamma\, \alpha_{\text{rest}}$,
$\gamma \in \Gamma$ oberstes Kellersymbol, Keller nicht leer:

$$C' = (q', v', \alpha') \text{ mit } \left.\begin{array}{l} v = xv' \\ \alpha' = \beta\, \alpha_{\text{rest}} \end{array}\right\} \text{ für ein } (q, \gamma, x, \beta, q') \in \Delta.$$

PDA Berechnungen

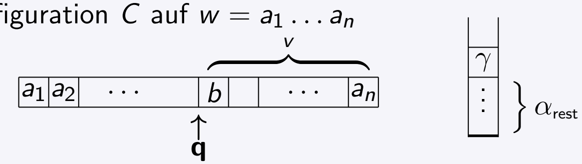
Startkonfiguration auf $w = a_1 \dots a_n$

$\mathbf{C_0[w]}$



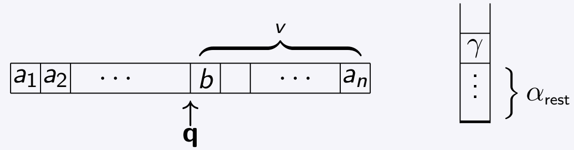
typische Konfiguration C auf $w = a_1 \dots a_n$

$\mathbf{C}$



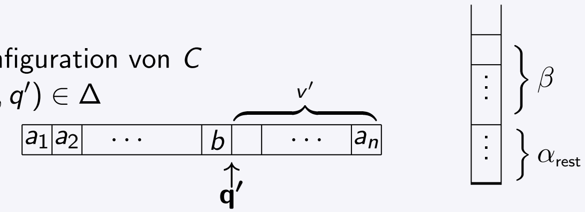
PDA Berechnungen

$\mathbf{C}$



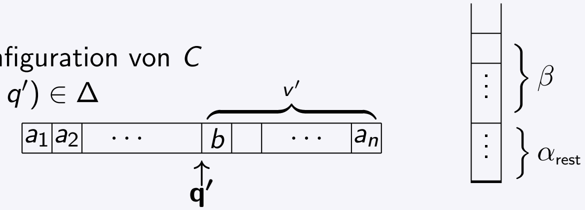
Nachfolgekonfiguration von C
zu $(q, \gamma, b, \beta, q') \in \Delta$

$\mathbf{C'}$



Nachfolgekonfiguration von C
zu $(q, \gamma, \varepsilon, \beta, q') \in \Delta$

$\mathbf{C'}$



PDA Berechnungen

(terminierende) Berechnung von $\mathcal{P}$ auf Eingabe $w \in \Sigma^*$:

Konfigurationsfolge $C_0 \dots C_f$, wobei

- $C_0 = C_0[w] = (q_0, w, \#)$,
- C_{i+1} eine Nachfolgekonfiguration von C_i, $0 \leqslant i < f$,
- C_f *Endkonfiguration* ohne anwendbare Transition

$\boxed{\textbf{Notation: } \mathbf{C_0[w]} \xrightarrow{\mathcal{P}} \mathbf{C_f}}$

akzeptierende Berechnung: $C_f = (q, \varepsilon, \varepsilon)$ mit $q \in A$

von $\mathcal{P}$ akzeptierte Sprache:

$L(\mathcal{P}) = \{ w \in \Sigma^* \colon C_0[w] \xrightarrow{\mathcal{P}} (q, \varepsilon, \varepsilon) \text{ für ein } q \in A \}$

Beispiel: PDA für Klammersprache

$\mathcal{P} = (\{(,)\}, Q, q_0, \Delta, \{q_0\}, \Gamma, \#)$,
$Q = A = \{q\}$, ein Zustand $q = q_0$
$\Gamma = \{|, \#\}$

Transitionen:

$(q, \#, (, \|\#, q)$	verarbeitet "(" und addiert "|" im Keller
$(q, \|, (, \|\|, q)$	verarbeitet "(" und addiert "|" im Keller
$(q, \|,), \varepsilon, q)$	verarbeitet ")" und löscht ein "|" im Keller
$(q, \#, \varepsilon, \varepsilon, q)$	ε-Transition, die $\#$ löscht

Idee: Kellerspeicher als Zähler für $|u|_(- |u|_)$

Satz: kontextfrei = PDA-erkennbar — Satz 4.1.5

Für $L \subseteq \Sigma^*$ sind äquivalent: (i) L kontextfrei.
(ii) $L = L(\mathcal{P})$ für einen PDA $\mathcal{P}$.

Beweis

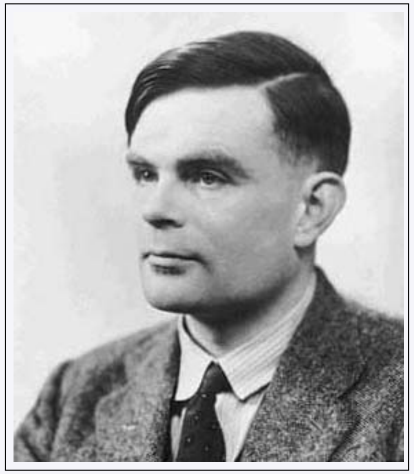
(i) $\Rightarrow$ (ii): aus $L = L(G)$, $G = (\Sigma, V, P, S)$ kontextfrei,
gewinne PDA $\mathcal{P} = (\Sigma, Q, q_0, \Delta, A, \Gamma, \#)$

$Q = A = \{q\}$, $q_0 = q$
$\Gamma = V \cup \Sigma$, $\# = S$
Transitionen: $(q, X, \varepsilon, \alpha, q)$ für Produktionen $X \to \alpha$ von G
$(q, a, a, \varepsilon, q)$ für jedes $a \in \Sigma$.

(ii) $\Rightarrow$ (i): aus $L = L(\mathcal{P})$, $\mathcal{P} = (\Sigma, Q, q_0, \Delta, \Gamma, \#)$,
gewinne kontextfreies $G = (\Sigma, V, P, S)$

Idee: Ableitungsschritte von $G \approx$ Berechnungsschritte von $\mathcal{P}$

Turing: prinzipielle Berechenbarkeit — $\to$ Abschnitt 4.2



Alan M. Turing (1912 – 1954)

Pionier der modernen Theorie der Berechenbarkeit

prinzipielle Grenzen und Möglichkeiten

1936 publiziert: "*On Computable Numbers*" mit mathematischer Abstraktion seiner Zuarbeiter (sog. 'computer')

Heutzutage "Turingmaschine" allgemein akzeptiert als Modell für Digitalrechner (PCs)

Turingmaschinen: DTM — $\to$ Abschnitt 4.2

DTM = DFA + unbeschränkter Lese/Schreibzugriff

Eingabe-/Arbeitsspeicher: unbeschränkte Folge von Zellen als "Band" mit Lese/Schreibkopf

Konfiguration bestimmt durch
- Zustand ($q \in Q$)
- Position auf dem Band
- Bandbeschriftung

Übergang in Nachfolgekonfiguration abhängig von
- Zustand
- aktuell gelesenem Bandsymbol

Übergang resultiert in
- Zustandswechsel
- Schreiben
- Kopfbewegung $(<, \circ, >)$

DTM $\mathcal{M} = (\Sigma, Q, q_0, \delta, q^+, q^-)$

Q Zustandsmenge
$q_0 \in Q$ Anfangszustand
$q^+/q^- \in Q$ akzeptierender/verwerfender Endzustand, $q^- \neq q^+$
δ Übergangsfunktion

$$\delta\colon Q \times (\Sigma \cup \{\Box\}) \to (\Sigma \cup \{\Box\}) \times \{<, \circ, >\} \times Q$$

Konfigurationen:
$C = (\alpha, q, x, \beta) \in (\Sigma \cup \{\Box\})^* \times Q \times (\Sigma \cup \{\Box\}) \times (\Sigma \cup \{\Box\})^*$
α: Bandinhalt links vom Kopf
x: Bandinhalt in Kopfposition
β: Bandinhalt rechts vom Kopf
q: aktueller Zustand

Startkonfiguration auf Eingabe w: $C_0[w] := (\varepsilon, q_0, \Box, w)$

Nachfolgekonfiguration: $C \longmapsto C'$ gemäß δ ...

Endkonfigurationen: $q \in \{q^+, q^-\}$, akzeptierend/verwerfend

DTM: Akzeptieren und Entscheiden → Abschnitt 4.3

von DTM $\mathcal{M}$ akzeptierte Sprache

$L(\mathcal{M}) = \{w \in \Sigma^*: \text{ M akzeptiert } w\ \} = \{w \in \Sigma^*: w \xrightarrow{\mathcal{M}} q^+\}$

Entscheidung (des Wortproblems) von L

$\mathcal{M}$ *entscheidet* L falls für alle $w \in \Sigma^*$:

$$w \xrightarrow{\mathcal{M}} \begin{cases} q^+ & \text{für } w \in L \\ q^- & \text{für } w \notin L \end{cases}$$

definit!

L entscheidbar (rekursiv):
L von einer DTM entschieden

L semi-entscheidbar (rekursiv aufzählbar):
L von einer DTM akzeptiert

Beispiel: DTM für Palindrom

δ	$\Box$	0	1
q_0	$(\Box, >, q^?)$		
$q^?$	$(\Box, \circ, q^+)$	$(\Box, >, q^{\to 0})$	$(\Box, >, q^{\to 1})$
$q^{\to 0}$	$(\Box, <, q^{\leftarrow 0})$	$(0, >, q^{\to 0})$	$(1, >, q^{\to 0})$
$q^{\to 1}$	$(\Box, <, q^{\leftarrow 1})$	$(0, >, q^{\to 1})$	$(1, >, q^{\to 1})$
$q^{\leftarrow 0}$	$(\Box, \circ, q^+)$	$(\Box, <, q^{\leftarrow})$	$(\Box, \circ, q^-)$
$q^{\leftarrow 1}$	$(\Box, \circ, q^+)$	$(\Box, \circ, q^-)$	$(\Box, <, q^{\leftarrow})$
$q^{\leftarrow}$	$(\Box, >, q^?)$	$(0, <, q^{\leftarrow})$	$(1, <, q^{\leftarrow})$

intendierte Rolle der Zustände:

q_0	:	Startzustand	$q^?$	:	Anfang abfragen
$q^{\to 0}$	:	zum Ende, merke 0	$q^{\leftarrow 0}$	:	vergleiche Ende mit 0
$q^{\to 1}$	:	zum Ende, merke 1	$q^{\leftarrow 1}$	:	vergleiche Ende mit 1
$q^{\leftarrow}$	:	zum Anfang	q^+/q^-	:	akzeptiere/verwerfe

Church-Turing These

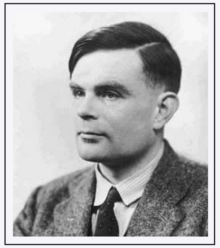
algorithmische Entscheidbarkeit = Turing-Entscheidbarkeit
algorithmische Erzeugbarkeit = Turing-Aufzählbarkeit
Berechenbarkeit = Turing-Berechenbarkeit

Belege:
- Erfahrung: alle akzeptierten Algorithmen lassen sich im Prinzip mit DTM simulieren
- Robustheit des TM Modells
- bewiesene Äquivalenz mit ganz unterschiedlichen alternativen Charakterisierungen

wichtig: idealisiertes Konzept von *prinzipieller* Machbarkeit
im Ggs. zu *praktischer* Machbarkeit

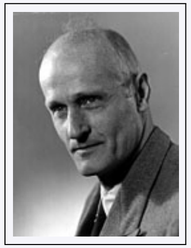
einige Väter der Berechenbarkeitstheorie

Church (1903–1995)



Turing (1912–1954)

Gödel (1906–1978)



Kleene (1909–1994)