Klausurblatt AfSE WiSe 19/20 | Skript: M. Otto

Definitionen / Wissen

- **Chomsky Hierarchie**

Typ	Produktionen	Akzeptor
Typ 0 - Allgemein	• beliebige Produktionen	• (D)TM akzeptiert nur teilweise • (D)TM entscheidet nur teilweise • rekursiv aufzählbar • teilweise entscheidbar
Typ 1 - Kontextsensitiv	• nur harmlose ϵ-Produktionen $\rightarrow X_0 \rightarrow \epsilon$ $\rightarrow X_0$ nur als Startsymbol $\rightarrow X_0$ also nie auf rechter Seite • Produktionen nicht verkürzend $\rightarrow \alpha \rightarrow \beta$ und $\lvert\beta\rvert \geq \lvert\alpha\rvert$	• (D)TM akzeptiert • (D)TM entscheidet • rekursiv aufzählbar und entscheidbar
Typ 2 - Kontextfrei	• Linke Seite nur eine Variable • X $\rightarrow$ v • CNF möglich	• PDA akzeptiert • CYK entscheidet • rekursiv aufzählbar und entscheidbar
Typ 3 - Regulär	• Alle Produktionen rechtslinear • X $\rightarrow \epsilon$, X $\rightarrow$ a, X $\rightarrow$ aY • falls Variable, dann ganz rechts	• NFA akzeptiert • DFA entscheidet • rekursiv aufzählbar und entscheidbar

Typ	abgeschlossen unter $\cup$	$\cap$	$^-$	$\cdot$	$*$
3	+	+	+	+	+
2	+	−	−	+	+
1	+	+	+	+	+
0	+	+	−	+	+
bel. Σ-Sprachen	+	+	+	+	+

- **Sprachen im Niveau der Chomsky-Hierarchie**
  - Sprache ($L \subseteq \Sigma^*$) vom selben Typ wie Grammatik G, falls es Grammatik G gibt mit $L = L(G)$
  - $L_{Typ3} \subsetneq L_{Typ2} \subsetneq L_{Typ1} \subsetneq L_{Typ0} \subsetneq \Sigma - Sprachen$
  - $\subsetneq$ Echte Teilmenge

- **Grammatik-Tricks:**
  - X_0: Neuer Startpunkt | $X_{0,1}$: Startpunkt erste Grammatik
  - Vereinigung: $X_0 \rightarrow X_{0,1} | X_{0,2}$
  - Konkatenation: $X_0 \rightarrow X_{0,1} X_{0,2}$
  - Stern-Bildung: $X_0 \rightarrow \epsilon | X_{0,1} X_0$

- **Definitionen**
  - **Grammatik** $G = (\Sigma, V, P, S)$
    $\rightarrow$ Terminalalphabet Σ, Variablen V, Produktionen P, Startsymbol S
  - **DFA/NFA** $A = (\Sigma, Q, q_0, \Delta/\delta, A)$
    $\rightarrow$ Eingabealphabet Σ, Zustandsmenge Q, Startzustand q_0
    $\rightarrow$ Übergangsrelation/-funktion Δ/δ, Akzeptierende Endzustände A
    $\rightarrow$ $\Delta = Q\ x\ \Sigma\ x\ Q$ (Von q mit x nach q')
  - **PDA** $P = (\Sigma, \Gamma, Q, q_0, A, \Delta, \#)$
    $\rightarrow$ Eingabealphabet Σ, Kelleralphabet Γ, Zustandsmenge Q, Startzustand q_0
    $\rightarrow$ Akzeptierende Endzustände A, Übergangsrelation Δ, Anfangs-Kellersymbol $\#$
    $\rightarrow$ $\Delta = Q\ x\ \Gamma\ x\ (\Sigma \cup \{\epsilon\})\ x\ \Gamma^*\ x\ Q$
    $\rightarrow$ (Von q, KellerPop, Lesenzeichen, KellerPush, nach q')
  - **Turingmaschine** $M = (\Sigma, Q, q_0, \delta, q+, q-)$
    $\rightarrow$ Bandalphabet Σ, Zustandsmenge Q, Startzustand q_0
    $\rightarrow$ Übergangsrelation δ, akzeptierender Endzustand $q+$, verwerfender Endzustand $q-$
    $\rightarrow$ $\delta = Q\ x\ (\Sigma \cup \{\Box\}) \rightarrow (\Sigma \cup \{\Box\})\ x\ \{<, o, >\}\ x\ Q$
    $\rightarrow$ Von q, Lese vom Band $\rightarrow$ schreibe aufs Band, Bewege, nach q'

- **Beweis-Tipps**
  - **Mengenverhältnisse** ($A \subseteq B$)
    * Am Besten über einzelne Elemente der Menge zeigen ($x \in A$)
    * Zeigen, von welcher Menge x Element ist
    * Danach Schluss darauf, dass es auch von anderer Seite Element ist
  - **L(G) = L**
    * $\supseteq$: Jedes w aus L ist in G $\rightarrow$ Induktion über die Wortlänge
    * $\subseteq$: Jedes ableitbare Wort von G ist in L $\rightarrow$ Induktion über die Anzahl an Ableitungsschritten
  - **Beweise der Art** $L_1 \cup L_2 = L \setminus ...$
    * Untersuchen, ob ϵ nur in einer Menge vorkommt (Widerlegbarkeit mit Gegenbeispiel)
  - **Gegenbeweis der Allaussage** $\forall n \in \mathbb{N} A(n)$
    * Zeige, durch Gegenbeispiel $\exists n \in \mathbb{N} \neg A(n)$
  - **Induktion über Wortlänge**
    * Induktionshypothese IH | Induktionsanfang IA | Induktionsschritt IS
    * IH: Aussage gilt für Wörter der Länge n
    * IA: Zeige für Länge n = 0, also ϵ
    * IS: Wort der Länge n + 1 auf Länge n zurück führen
  - **Induktion über die Anzahl der Ableitungsschritte**
    * IH: Aussage über von einer Grammatik erzeugten Worte (A(w))
    * IA: A(w) gilt für alle Worte, die in einem Schritt ableitbar sind ($\rightarrow_G$)
    * IS: A(w) gilt für alle Worte über $\rightarrow_G^*$ mit $n + 1$ Ableitungsschritten
    $\rightarrow$ Zurückführen auf ein Wort mit n Ableitungsschritten und Zeigen des letzten Schrittes
  - **Induktion über Erzeugungsprozess/Konkatenation**
    * IH: A(w): Aussage über alle Worte der Sprache
    * IA: A(w) gilt für $w = \epsilon$
    * IS: A(wa) nachweisen über A(w) und Konkatenation mit a ($\forall a \in \Sigma$)

- **Pumping Lemma(regulär) - Anwendung**
  - Schema zur Widerlegung:
  - WENN:
    - $\forall n \in \mathbb{N}$ gilt:
    - $\exists x \in Lmit|x| \geq n$:
    - $\forall$ Zerlegung $x = uvw$ mit $v \neq \epsilon$ und $|uv| \leq n$
    - $\exists m \in \mathbb{N}$, sodass $uv^m w \notin L$
  - DANN:
    - L nicht kontextfrei

- **Beispiel:**
  - $L = \{a^p b^p : p \geq 0\}$
  - Sei $n \in \mathbb{N}$ gegeben (All-Aussage)
  - Setze $x = a^n b^n$ (Wort länger als vorgegebene Länge n)
  - Sei Zerlegung mit $x = uvw$ mit $v \neq \epsilon$ und $|uv| \leq n$ gegeben (All-Aussage)
  - Setze $m = 0$. Dann hat $uv^m w = uw$ weniger a als b. Damit folgt $uv^m w \notin L$
  - Da v nur aus a besteht, enthält das Wort weniger a als b, wenn $v^0 = \epsilon$ nutzt

$$x = \overbrace{\underbrace{a\,a}_{u}\,\underbrace{a\ldots a}_{v}}^{n}\;\underbrace{\overbrace{bbb\ldots b}^{n}}_{w}$$

(da $|uv| \leq n \rightarrow v$ besteht nur aus a)

- **Pumping Lemma(kontextfrei) - Anwendung**
  - Schema zur Widerlegung:
  - WENN:
    - $\forall n \in \mathbb{N}$ gilt:
    - $\exists x \in Lmit|x| \geq n$:
    - $\forall$ Zerlegung $x = yuvwz$ mit $uw \neq \epsilon$ und $|uvw| \leq n$
    - $\exists m \in \mathbb{N}$, sodass $yu^m vw^m z \notin L$
  - DANN:
    - L nicht erkennbar | L ist keine reguläre Sprache

  - **Beispiel:**
    - $L = \{a^n b^n c^n : n \in \mathbb{N}\}$
    - Sei $n \in \mathbb{N}$ beliebig:
    - Wähle $x = a^n b^n c^n$:
    - Zerlegung ...
    - Fall 1: uvw hat kein c. $m = 2$: $yu^m vw^m z$ hat mehr a/b als c $\Rightarrow yu^m vw^m z \notin L$
    - Fall 2: uvw hat kein a. $m = 2$: $yu^m vw^m z$ hat mehr b/c als a $\Rightarrow yu^m vw^m z \notin L$
    - (Aufgrund von $|uvw| \leq n$ kann es, sobald es ein c hat, kein a mehr haben)

  $\Rightarrow$ L ist nicht kontextfrei

- **Myhill-Nerode - Anwendung**
  - Satz von Myhill-Nerode:
    - $L \in \Sigma^*$ ist erkennbar $\Leftrightarrow \sim_L$ hat endlichen Index
  - Wortäquivalenz $\sim_L$
    - $w \sim_L w' \;\Leftrightarrow\; \forall x \in \Sigma^* : \; (wx \in L \Leftrightarrow w'x \in L)$
    - Für $\sim_L$ gelten folgende Eigenschaften:
      1. $\sim_L$ ist rechts invariant: $w \sim_L w' \Rightarrow \forall u \in \Sigma^*(wu \sim_L w'u)$
      2. L ist abgeschlossen unter $\sim_L$: $(w \in L \wedge w \sim_L w') \Rightarrow w' \in L$
      3. L ist die Vereinigung aller Äquivalenzklassen von $\sim_L$
  - Zustandsäquivalenz $\sim_A$
    - $w \sim_A w' \Leftrightarrow \hat{\delta}(q_0, w) = \hat{\delta}(q_0, w')$
    - Für $\sim_A$ gelten folgende Eigenschaften:
      1. $\sim_A$ hat endlichen Index, nämlich $index(\sim_A) \geq |Q|$
      2. $\sim_A$ ist rechts-invariant: $w \sim_A w' \Rightarrow \forall u \in \Sigma^* wu \sim_A w'u$
      3. $\sim_A$ verfeinert $\sim_L$: $w \sim_A w' \Rightarrow w \sim_L w'$

  - **Anwendung**
    - Aufzeigen unendlich vieler Äquivalenzklassen
    - Beispiel: $L = \{w \in \{a, b\}^* | \; w \; hat \; mehr \; b \; als \; a\}$
    - $k < n$
    - $b^n a^k \in L$ (Aufstellen von b^k und b^n und Anhängen des selben Wortes)
    - $b^n a^k \notin L$ (Eins liegt in L, das andere nicht)
    - Dementsprechend gilt $\sim_L$ nicht

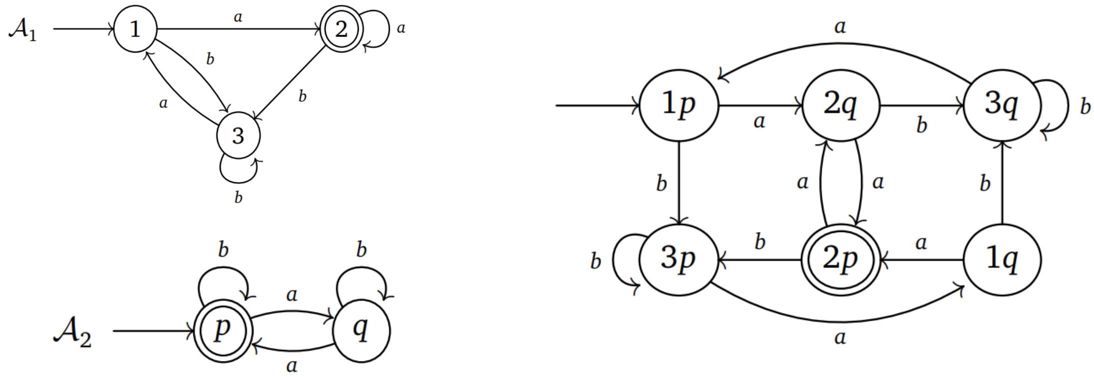
    $\Rightarrow$ k und n hier beliebig gewählt, dementsprechend beliebig viele Äquivalenzklassen

    $\Rightarrow$ $| \sim_L | = \infty$

    $\Rightarrow$ Sprache nicht regulär

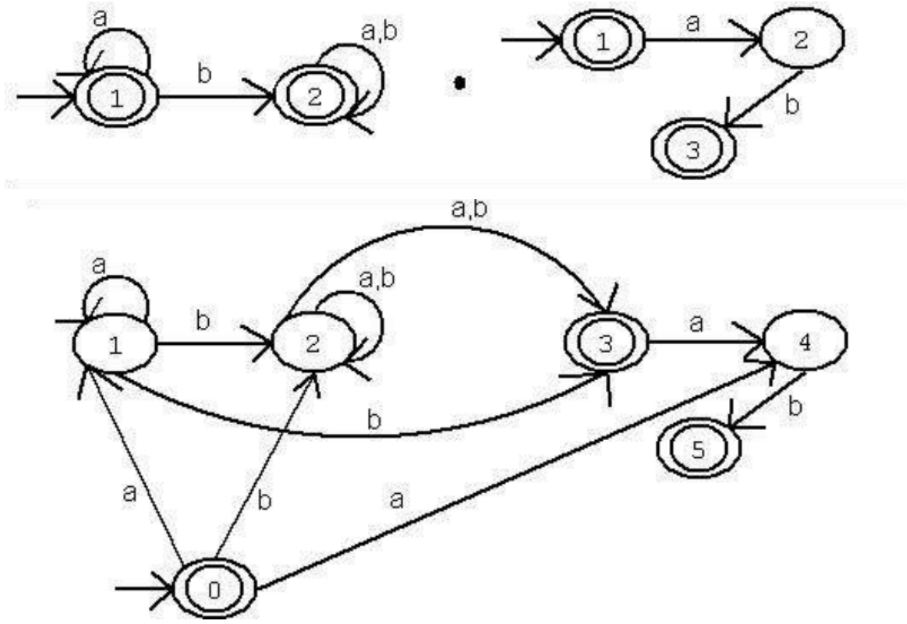
Algorithmen / Rechenmuster

- **Produktautomat Durchschnitt $\cap$ und Schnitt $\cup$**



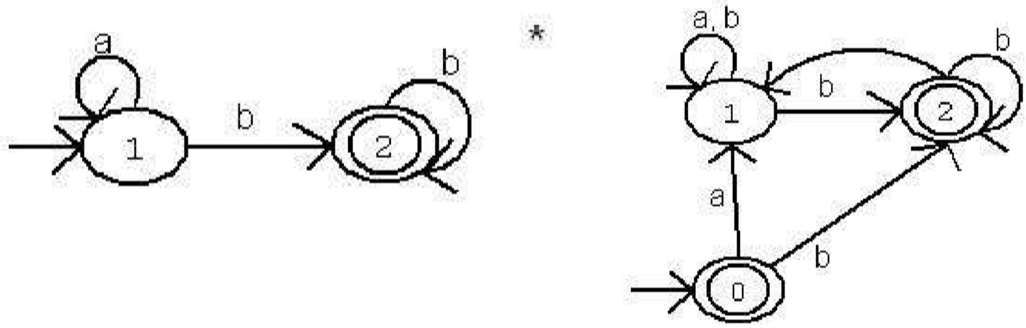
  - 1. Beide Startzustände zu 1p zusammenfassen
  - 2. Von dort aus neue Zustände bilden, die man z.B. mit a erreicht
  - Bsp: $1 \rightarrow^a 2$ und $p \rightarrow^a q \Rightarrow$ 2q
  - 3. Fortfahren bis alle Zustände abgedeckt sind
  - **Akzeptierende Zustände**
    * Bei $\cup$: Alle Zustände, die zu mindestens einem Teil aus altem akzeptierenden Zustand bestehen
    * Bei $\cap$: Alle Zustände, die zu beiden Teilen aus alten akzeptierenden Zuständen bestehen

- **Konkatenations-Automat $\cdot$**

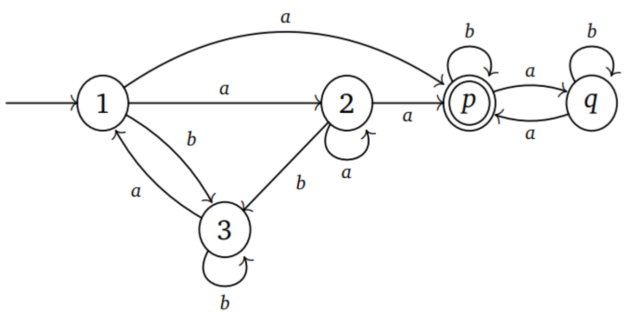


  - Konkatenation durch Aneinanderhängen zweier Automaten
  - Für jede Transition des ersten Automaten in dessen Endzustand:
    $\Rightarrow$ Einbauen einer Transition in den Anfangszustand des anderen Automaten
    $\Rightarrow$ Dies gilt insbesondere auch für Schleifen, die die vom akzep. Zustand auf sich selbst zeigen
  - **Akzeptierende Zustände:** nur die des zweiten Automaten
  - **Achtung:** Falls erster Automat ϵ akzeptiert:
    $\Rightarrow$ Einführen eines extra Startzustands, der alle Transitionen beider Startzustände besitzt
    $\Rightarrow$ Falls beide Automaten ϵ akzeptieren $\rightarrow$ neuer Startzustand akzeptierend

- **Sternautomat * (NFA)**
  - Stern-Sprache der aktuellen Sprache



  - 1. Einbauen eines neuen akzeptierenden Startzustands
  - 2. Neuer Startzustand enthält alle Transitionen des alten Starts
  - 3. Lenkung aller Transitionen, die auf akzep. Zustand zeigen auf den alten Start (oder neuer Start)
  - (auch Selbstschleifen)
- **Komplementbildung eines Automaten $\bar{L}$**
  - Wechsel der akzeptierenden und nicht akzeptierenden Zustände
- **Potenzmengenautomat (NFA → DFA)**
  - Jede Zustandsmenge simuliert in welchen Zuständen sich der NFA befinden könnte
  - 1. Erstellen einer Tabelle und Start bei Anfangszustand $\{0\}$
  - 2. Notieren aller Zustände, die der Startzustand mithilfe einer Transition erreicht
  - 3. Erstellen eines neuen Zeileneintrags mit dieser Menge an Zuständen
  - 4. Durchführung bis alle Zustände abgedeckt sind
  - **Akzeptierende Zustände:** Akzeptiert, falls einer der Zustände in der Menge akzeptierend ist



δ	a	b
$\{1\}$	$\{2,p\}$	$\{3\}$
$\{2,p\}$	$\{2,p,q\}$	$\{3,p\}$
$\{3\}$	$\{1\}$	$\{3\}$
$\{2,p,q\}$	$\{2,p,q\}$	$\{3,p,q\}$
$\{3,p\}$	$\{1,q\}$	$\{3,p\}$
$\{3,p,q\}$	$\{1,p,q\}$	$\{3,p,q\}$
$\{1,q\}$	$\{2,p\}$	$\{3,q\}$
$\{1,p,q\}$	$\{2,p,q\}$	$\{3,p,q\}$
$\{3,q\}$	$\{1,p\}$	$\{3,q\}$
$\{1,p\}$	$\{2,p,q\}$	$\{3,p\}$

  - **Hier:** Links: NFA Rechts: Tabelle zu DFA
  - Beispiel hier: Startzustand führt mit a zu 2 und p deswegen neuer Zustand mit $\{2,p\}$
  - Tipp: Vielleicht hilfreich Transitionstabelle für Quellautomaten zu machen (Ablesefehler)

- **Satz von Kleene (Automat → regulärer Ausdruck)**
  - Durchführen von k Iterationschritten zum Erhalten des regulären Ausdrucks
  - k = Anzahl der Zustände
  - Rekursionsformel: $a^{k+1}_{l,m} = a^k_{l,m} + a^k_{l,k+1}(a^k_{k+1,k+1})^* a^k_{k+1,m}$

  - **Durchführung**
    - 1. Aufstellen von $a^0_{l,m}$ mithilfe der Formel:

$$\alpha^0_{\ell,m} = \begin{cases} a_1 + \cdots + a_r & \text{falls } \ell \neq m \text{ und } \delta(\ell, a_i) = m \text{ für } i = 1, \ldots, r \\ \epsilon + a_1 + \cdots + a_r & \text{falls } \ell = m \text{ und } \delta(\ell, a_i) = m \text{ für } i = 1, \ldots, r \end{cases}$$

    - 2. Anwendung der Rekursionsformel zur Erstellung von $a^k_{l,m}$
    - 3. Durchführung bis $a^{k-1}_{l,m}$
    - 4. Danach Durchführung für die "akzeptierende Zelle" um regulären Ausdruck zu erhalten

  - Tipps: (in k-ter Tabelle)
    - $a^k_{l,m}$: entspricht Werten in vorheriger Tabelle an selber Stelle
    - $a^k_{l,k+1}$: entspricht jeweils den Werten der k-ten Spalte in vorheriger Tabelle
    - $(a^k_{k+1,k+1})^*$: ist für die ganze Tabelle der selbe Wert (k-te Zeile, k-te Spalte)
    - $a^k_{k+1,m}$: entspricht jeweils den Werten der k-ten Zeile in vorheriger Tabelle
  - **Beispiel im Anhang**

- **Minimierung eines Automaten**
  - 1. Start bei $\sim_0$: Einteilung der Zustände in Akzeptierend und Nicht-Akzeptierend
  - 2. Eintragen in Tabelle mit Klassen → Transitionen enden in Klassen
  - 3. Aufteilung einer Klasse in Unterklasse, falls Elemente unterschiedliche Transitionen haben
  - 4. Durchführung für $\sim_{i+1}$, bis jedes Element in jeder Klasse die selbe Transition hat
  - 5. Aufzeichnen des neuen Automaten mit Klassen als Zustände
  - **Akzeptierende Klasse/Zustände**: Klasse, die akzeptierende Zustände enthält

	$\sim_0$	a	b
$p_1^{(0)}$	1	p_1	p_1
	2	p_2	p_1
	4	p_1	p_1
	5	p_1	p_1
	6	p_2	p_1
$p_2^{(0)}$	3	p_2	p_2
	7	p_2	p_2
	8	p_2	p_2

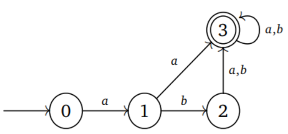
	$\sim_1$	a	b
$p_1^{(1)}$	1	p_1	p_1
	4	p_2	p_1
	5	p_2	p_1
$p_2^{(1)}$	2	p_3	p_1
	6	p_3	p_1
$p_3^{(1)}$	3	p_3	p_3
	7	p_3	p_3
	8	p_3	p_3

	$\sim_2$	a	b
$p_1^{(2)}$	1	p_1	p_2
$p_2^{(2)}$	4	p_3	p_2
	5	p_3	p_2
$p_3^{(2)}$	2	p_4	p_2
	6	p_4	p_2
$p_4^{(2)}$	3	p_4	p_4
	7	p_4	p_4
	8	p_4	p_4

- **Umformung kontextfreier Grammatik in Chomsky-Normalform**
  - **Bedingung:** Alle Produktionen in Form: $X \to YZ$ oder $A \to a$
  - **Durchführung**
    - 1. Eliminiere ϵ-Produktionen
      - 1.1 Aufstellen einer Nicht-Terminal-Menge, die zu ϵ-Produktionen führt
      - 1.2 Ersetzen durch alle möglichen Kombinationen ohne ϵ an Stellen aus Menge

$$P: \begin{array}{lcl} X_0 & \to & aXY \mid bXb \mid a \\ X & \to & aXa \mid bY \mid \varepsilon \\ Y & \to & bX_0a \mid aX_0 \end{array} \qquad \begin{array}{lcl} X_0 & \to & aXY \mid bXb \mid a \mid aY \mid bb \\ X & \to & aXa \mid bY \mid aa \\ Y & \to & bX_0a \mid aX_0 \end{array}$$

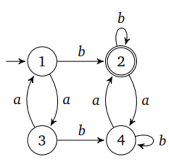
    - 2. Variablen für Terminale einfügen ($A \to a, B \to B$)
    - 3. Kettenproduktionen eliminieren ($X \to Y$)
    - 4. Eliminiere $X \to X_0, .., X_k$ mit $k \geq 3$ (Mehr als 2 Variablen aufteilen)
    - Aus $A \to ABC$ wird $A \to AD$ und $D \to BC$
  - **Beispiel im Anhang**
- **CYK Algorithmus**
  - Bestimmung des Wortproblems für kontextfreie Grammatiken in CNF
  - **Durchführung**
    - 1. Aufstellen einer Tabelle (Größe nxn (n = Wortlänge)
    - 2. Eintragen der Integranden für einzelne Buchstaben in oberste Zeile
    - 3. Ausfüllen der Tabelle (guckt euch 'n Video an)
    - 4. Falls das Startsymbol in der letzten Zeile entsteht $\to$ Wort wird durch Grammatik erkannt
  - Ableitungsbaum ausgehend von S erstellen, falls benötigt
- **Reguläre Grammatiken $\Leftrightarrow$ Automaten**
  - Ziemlich selbstverständlich anhand Beispielen



$$\begin{array}{lcl} X_0 & \to & aX_1 \\ X_1 & \to & aX_3 \mid bX_2 \\ X_2 & \to & aX_3 \mid bX_3 \\ X_3 & \to & aX_3 \mid bX_3 \mid \varepsilon \end{array}$$

$G = (\{a,b\}, \{X_1, X_2, X_3, X_4\}, P, X_1)$ mit

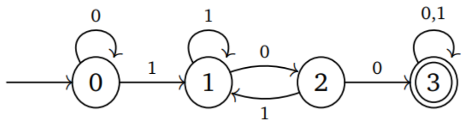
$$P: \begin{array}{lcl} X_1 & \to & aX_3 \mid bX_2 \\ X_2 & \to & aX_4 \mid bX_2 \mid \varepsilon \\ X_3 & \to & aX_1 \mid bX_4 \\ X_4 & \to & aX_2 \mid bX_4. \end{array}$$



Automatenbeispiele

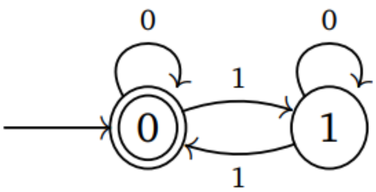
- L: Wörter, in denen 100 als Teilwort vorkommt
  - $a = (0+1)^*100(0+1)^*$

$\mathcal{A}_1$



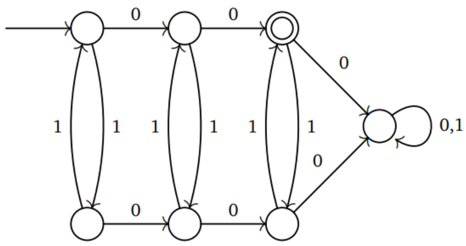
- L: Wörter mit gerader Anzahl von 1
  - $a = (0 + 10^*1)^*$

$\mathcal{A}_2$

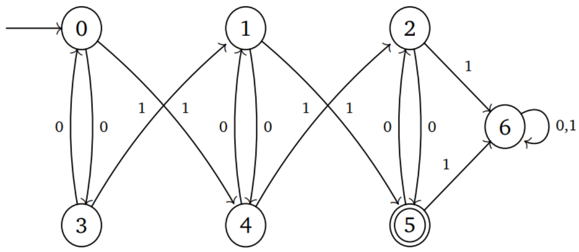


- L: Wörter mit gerader Anzahl von 1 und genau zweimal 0
  - Mit $\alpha = 1(11)^* und \beta = (11)^*$
  - $a = \alpha 0 \alpha 0 \beta + \alpha 0 \beta 0 \alpha + \beta 0 \alpha 0 \alpha + \beta 0 \beta 0 \beta$

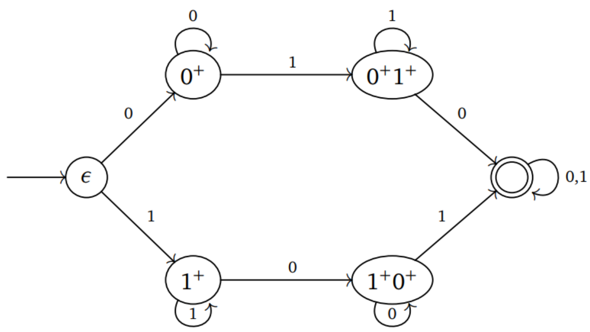
$\mathcal{A}_3$



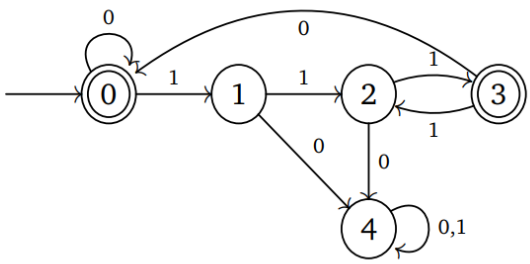
- L: Wörter von ungerader Länge mit genau zwei 1



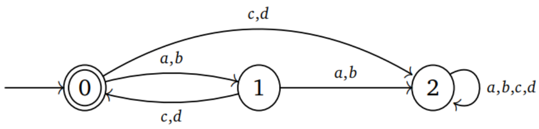
- L: Wörter, die 01 und 10 als (nicht notwendigerweise disjunkte) Teilwörter enthalten



- L: Wörter, in denen alle 1-Blöcke eine Länge der Form $2n + 3$ haben



- L $= L(((a+b)(c+d))^*)$



Sprachen $\Rightarrow$ Grammatik

- $L((bc+a)^*aba(ac+b)^*)$

Eine Grammatik, die diese Sprache erzeugt, ist z.B. $G = \big(\{a,b,c\},\{S,X,Y\},P,S\big)$ mit

$$\begin{array}{rrcl} P: & S & \to & aba \,|\, XabaY \,|\, Xaba \,|\, abaY \\ & X & \to & bc \,|\, bcX \,|\, a \,|\, aX \\ & Y & \to & ac \,|\, acY \,|\, b \,|\, bY\,. \end{array}$$

- $L = \{ucw \in \{a,b,c\}^* | u,w \in a,b,c^*, |u_b| = |w_b|\}$

Eine Grammatik, die diese Sprache erzeugt, ist z.B. $G = \big(\{a,b,c\},\{S\},P,S\big)$ mit

$$P: \quad S \to bSb \,|\, aS \,|\, Sa \,|\, cS \,|\, Sc \,|\, c\,.$$

Aufgabe H6.1 (Kontextfreie Grammatiken)
Geben Sie kontextfreie Grammatiken an, die folgende Sprachen über dem Alphabet $\Sigma = \{a,b,c\}$ erzeugen:

(a) $L_1 := \{a^n b^m \mid m \le n \le 2m\}$.

(b) $L_2 := \{ucv \mid u,v \in \{a,b\}^*,\ |u| = |v|\}$.

(c) $L_3 := \{a^i b^j c^k \mid i+j=k\}$

(d) $L_4 := \{a^i b^j c^k \mid j=i+k\}$

(e) $L_5 := \{a^i b^j c^k \mid i=j \text{ oder } i=k\}$

(f) $L_6 := \emptyset$

Lösung: Je $\boxed{2\text{ P.}}$:

(a) $G_1 = (\Sigma,\{S\},P,S)$ mit

$$P: \quad S \to aSb \,|\, aaSb \,|\, \varepsilon$$

(b) $G_2 = (\Sigma,\{S\},P,S)$ mit

$$P: \quad S \to aSa \,|\, aSb \,|\, bSa \,|\, bSb \,|\, c$$

(c) $G_3 = (\Sigma,\{S,X\},P,S)$ mit

$$\begin{array}{rrcl} P: & S & \to & aSc \,|\, X \\ & X & \to & bXc \,|\, \varepsilon \end{array}$$

(d) $G_4 = (\Sigma,\{S,X,Y\},P,S)$ mit

$$\begin{array}{rrcl} P: & S & \to & aXbY \,|\, XbYc \,|\, \varepsilon \\ & X & \to & aXb \,|\, \varepsilon \\ & Y & \to & bYc \,|\, \varepsilon \end{array}$$

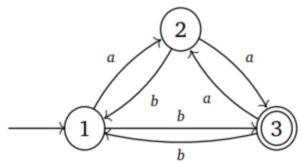
(e) $G_5 = (\Sigma,\{S,X,Y,U,V\},P,S)$ mit

$$\begin{array}{rrcl} P: & S & \to & XY \,|\, U \\ & X & \to & aXb \,|\, \varepsilon \\ & Y & \to & cY \,|\, \varepsilon \\ & U & \to & aUc \,|\, V \\ & V & \to & bV \,|\, \varepsilon \end{array}$$

(f) $G_6 = (\Sigma,\{S\},P,S)$ mit

$$P: \quad S \to S$$

Satz von Kleene - Beispiel



Lösung: Für $k = 0$ bekommen wir folgende Ausdrücke $\alpha^0_{\ell,m}$: (2 Punkte)

	$m = 1$	2	3
$\ell = 1$	ϵ	a	b
2	b	ϵ	a
3	b	a	ϵ

Mit der Rekursionsformel

$$\alpha^{k+1}_{\ell,m} = \alpha^k_{\ell,m} + \alpha^k_{\ell,k+1}(\alpha^k_{k+1,k+1})^*\alpha^k_{k+1,m}$$

ergibt sich im nächsten Schritt folgende Tabelle mit Ausdrücken $\alpha^1_{\ell,m}$, die wir zu der Tabelle auf der rechten Seite vereinfachen können: (4 Punkte)

	1	2	3
1	$\epsilon + \epsilon\epsilon^*\epsilon$	$a + \epsilon\epsilon^* a$	$b + \epsilon\epsilon^* b$
2	$b + b\epsilon^*\epsilon$	$\epsilon + b\epsilon^* a$	$a + b\epsilon^* b$
3	$b + b\epsilon^*\epsilon$	$a + b\epsilon^* a$	$\epsilon + b\epsilon^* b$

	1	2	3
1	ϵ	a	b
2	b	$\epsilon + ba$	$a + bb$
3	b	$a + ba$	$\epsilon + bb$

Man beachte den systematischen Aufbau der Terme!
Für $k = 2$ erhalten wir (4 Punkte)

	1	2	3
1	$\epsilon + a(\epsilon + ba)^* b$	$a + a(\epsilon + ba)^*(\epsilon + ba)$	$b + a(\epsilon + ba)^*(a + bb)$
2	$b + (\epsilon + ba)(\epsilon + ba)^* b$	$\epsilon + ba + (\epsilon + ba)(\epsilon + ba)^*(\epsilon + ba)$	$a + bb + (\epsilon + ba)(\epsilon + ba)^*(a + bb)$
3	$b + (a + ba)(\epsilon + ba)^* b$	$a + ba + (a + ba)(\epsilon + ba)^*(\epsilon + ba)$	$\epsilon + bb + (a + ba)(\epsilon + ba)^*(a + bb)$

,

was wie folgt vereinfacht werden kann:

	1	2	3
1	$(ab)^*$	$a(ba)^*$	$b + a(ba)^*(a + bb)$
2	$(ba)^* b$	$(ba)^*$	$(ba)^*(a + bb)$
3	$b + (a + ba)(ba)^* b$	$(a + ba)(ba)^*$	$\epsilon + bb + (a + ba)(ba)^*(a + bb)$

,

Schließlich können wir den Ausdruck $\alpha^3_{1,3}$ wie folgt bestimmen: (2 Punkte)

$$\begin{aligned}
\alpha^3_{1,3} &= \alpha^2_{1,3} + \alpha^2_{1,3}(\alpha^2_{3,3})^*\alpha^2_{3,3} \\
&= b + a(ba)^*(a + bb) + \\
&\quad (b + a(ba)^*(a + bb))\,(\epsilon + bb + (a + ba)(ba)^*(a + bb))^*\,(\epsilon + bb + (a + ba)(ba)^*(a + bb)),
\end{aligned}$$

was wir wiederum zu

$$(b + a(ba)^*(a + bb))\,(bb + (a + ba)(ba^*)(a + bb))^*$$

vereinfachen können.

Chomsky-Normalform - Beispiel

Betrachten Sie die kontextfreie Grammatik $G = (\{a,b\},\{X_0,X,Y\},P,X_0)$ mit

$$P: \begin{array}{lcl} X_0 & \to & aXY \mid bXb \mid a \\ X & \to & aXa \mid bY \mid \varepsilon \\ Y & \to & bX_0a \mid aX_0 \end{array}$$

Konstruieren Sie eine zu G äquivalente Grammatik in Chomsky-Normalform.

Lösung: 1. Schritt (eliminiere ε-Produktionen) $\boxed{4\text{ P.}}$:

$$\begin{array}{lcl} X_0 & \to & aXY \mid bXb \mid a \mid aY \mid bb \\ X & \to & aXa \mid bY \mid aa \\ Y & \to & bX_0a \mid aX_0 \end{array}$$

2. Schritt (Variablen vor Buchstaben) $\boxed{4\text{ P.}}$:

$$\begin{array}{lcl} X_0 & \to & Z_aXY \mid Z_bXZ_b \mid Z_a \mid Z_aY \mid Z_bZ_b \\ X & \to & Z_aXZ_a \mid Z_bY \mid Z_aZ_a \\ Y & \to & Z_bX_0Z_a \mid Z_aX_0 \\ Z_a & \to & a \\ Z_b & \to & b \end{array}$$

3. Schritt (eliminiere $X \to Y$ und $X \to X_0 \dots X_k$ mit $k \geq 3$) $\boxed{4\text{ P.}}$:

$$\begin{array}{lcl} X_0 & \to & Z_aU \mid Z_bV \mid a \mid Z_aY \mid Z_bZ_b \\ X & \to & Z_aW \mid Z_bY \mid Z_aZ_a \\ Y & \to & Z_bT \mid Z_aX_0 \\ U & \to & XY \\ V & \to & XZ_b \\ W & \to & XZ_a \\ T & \to & X_0Z_a \\ Z_a & \to & a \\ Z_b & \to & b \end{array}$$

Quizfragen

	$\Rightarrow$	$\Leftarrow$		
L ist regulär	☐	☒	L ist endlich	
	☒	☒	L wird von einem DFA akzeptiert	Kleene
	☒	☒	L wird von einem NFA akzeptiert	Kleene
	☒	☒	$\Sigma^* \setminus L$ ist regulär	
	☒	☐	L^* ist regulär	
	☒	☐	L enthält eine reguläre Sprache, d.h. es gibt eine reguläre Sprache $L_1 \subseteq \Sigma^*$ mit $L_1 \subseteq L$	$(\emptyset)$
	☒	☐	L ist Teilmenge einer regulären Sprache, d.h. es gibt eine reguläre Sprache $L_2 \subseteq \Sigma^*$ mit $L \subseteq L_2$	(Σ^*)

(a) Es gibt kontextfreie Sprachen, die regulär sind.

Lösung: *Richtig.* Jede reguläre Sprache ist auch kontextfrei.

(b) Jede kontextfreie Sprache ist kontextsenstiv.

Lösung: *Richtig.* Zwar ist nicht jede kontextfreie Grammatik auch kontextsensitiv, aber es gibt zu jeder kontextfreien Grammatik eine kontextsensitive, die die gleiche Sprache beschreibt.

(c) Für zwei kontextfreie Sprachen L_1, L_2 ist auch $L_1 \cup L_2$ kontextfrei.

Lösung: *Richtig.*

(d) Der Schnitt einer kontextfreien mit einer regulären Sprache ist regulär.

Lösung: *Falsch.* $\{a^n b^n : n \geq 0\}$ ist kontextfrei aber nicht regulär. Für $\{a^n b^n : n \geq 0\} \cap \Sigma^*$ gilt das auch.

(e) Es gibt eine kontextfreie Sprache L, für die $\sim_L$ unendlichen Index hat.

Lösung: *Richtig*, denn sonst wären alle kontextfreien Sprachen regulär. Ein Beispiel für eine solche Sprache ist $\{a^n b^n : n \geq 0\}$.

(f) Ist L_1 kontextfrei und $L_2 \subseteq L_1$, so ist auch L_2 kontextfrei.

Lösung: *Falsch.* Sei $L_1 = \Sigma^*$ und L_2 eine nicht-kontextfreie Sprache.

Erstellung Kellerautomat aus Grammatik

Aufgabe G8.1 (Kellerautomaten)
Konstruieren Sie einen Kellerautomaten, der die folgende kontextfreie Sprache erkennt:

$$L = \{a^i b^j c^k \,:\, i = j + k\}.$$

Lösung: Sei $\mathcal{P} = (\Sigma, Q, q_a, \Delta, A, \Gamma, \#)$ der Kellerautomat mit Eingabealphabet $\Sigma = \{a, b, c\}$, Zustandsmenge $Q = \{q_a, q_b, q_c\}$, q_a als Anfangszustand, $A = \{q_a, q_b, q_c\}$ als Menge der akzeptierenden Zustände, Kelleralphabet $\Gamma = \{\#, |\}$ und Übergangsrelation Δ gegeben durch

$$\begin{array}{ll} \{ & (q_a, \#, \varepsilon, \varepsilon, q_a) \\ & (q_a, \#, a, |, q_a) \\ & (q_a, |, a, ||, q_a) \\ & (q_a, |, b, \varepsilon, q_b) \\ & (q_a, |, c, \varepsilon, q_c) \\ & (q_b, |, b, \varepsilon, q_b) \\ & (q_b, |, c, \varepsilon, q_c) \\ & (q_c, |, c, \varepsilon, q_c) \quad \}. \end{array}$$

Dann erkennt $\mathcal{P}$ die Sprache L.
Idee: pro Buchstabe wird ein Zustand benötigt, im Stack wird der a-Zähler mit | erhöht beim Lesen von a und beim Lesen von b und c jeweils um eins verringert. Wenn der Stack leer ist, gilt $i = j + k$ und damit ist der jeweilige Zustand akzeptierend. Die Zustandsübergänge sind derart, dass nach dem Einlesen von b nur b, c und nach Lesen von c nur weitere c akzeptiert werden.