

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

Definitionen / Wissen

• Chomsky Hierarchie

Typ	Produktionen	Akzeptor
Typ 0 - Allgemein	<ul style="list-style-type: none"> beliebige Produktionen 	<ul style="list-style-type: none"> (D)TM akzeptiert nur teilweise (D)TM entscheidet nur teilweise rekursiv aufzählbar teilweise entscheidbar
Typ 1 - Kontextsensitiv	<ul style="list-style-type: none"> 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 $\beta \geq \alpha$ 	<ul style="list-style-type: none"> (D)TM akzeptiert (D)TM entscheidet rekursiv aufzählbar und entscheidbar
Typ 2 - Kontextfrei	<ul style="list-style-type: none"> Linke Seite nur eine Variable $X \rightarrow v$ CNF möglich 	<ul style="list-style-type: none"> PDA akzeptiert CYK entscheidet rekursiv aufzählbar und entscheidbar
Typ 3 - Regulär	<ul style="list-style-type: none"> Alle Produktionen rechtslinear $X \rightarrow \epsilon$, $X \rightarrow a$, $X \rightarrow aY$ falls Variable, dann ganz rechts 	<ul style="list-style-type: none"> 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 $L = L(G)$
- $L_{Typ3} \subsetneq L_{Typ2} \subsetneq L_{Typ1} \subsetneq L_{Typ0} \subsetneq \Sigma - \text{Sprachen}$
- \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** $G = (\Sigma, V, P, S)$
 - Terminalalphabet Σ , Variablen V , Produktionen P , Startsymbol S
- **DFA/NFA** $A = (\Sigma, Q, q_0, \Delta/\delta, A)$
 - Eingabealphabet Σ , Zustandsmenge Q , Startzustand q_0
 - Übergangsrelation/-funktion Δ/δ , Akzeptierende Endzustände A
 - $\Delta = Q \times \Sigma \times Q$ (Von q mit x nach q')
- **PDA** $P = (\Sigma, \Gamma, Q, q_0, A, \Delta, \#)$
 - Eingabealphabet Σ , Kelleralphabet Γ , Zustandsmenge Q , Startzustand q_0
 - Akzeptierende Endzustände A , Übergangsrelation Δ , Anfangs-Kellersymbol $\#$
 - $\Delta = Q \times \Gamma \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \times Q$
 - (Von q , KellerPop, Lesenzeichen, KellerPush, nach q')
- **Turingmaschine** $M = (\Sigma, Q, q_0, \delta, q+, q-)$
 - Bandalphabet Σ , Zustandsmenge Q , Startzustand q_0
 - Übergangsrelation δ , akzeptierender Endzustand $q+$, verwerfender Endzustand $q-$
 - $\delta = Q \times (\Sigma \cup \{\square\}) \rightarrow (\Sigma \cup \{\square\}) \times \{<, o, >\} \times Q$
 - Von q , Lese vom Band → 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** $L_1 \cup L_2 = L \setminus \dots$
 - * 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
 - 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:
 - * $\exists x \in L$ mit $|x| \geq n$:
 - * \forall Zerlegung $x = uvw$ mit $v \neq \epsilon$ und $|uv| \leq n$
 - * $\exists m \in \mathbb{N}$, sodass $uv^mw \notin L$
- 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 $uv^mw = uw$ weniger a als b. Damit folgt $uv^mw \notin L$
- Da v nur aus a besteht, enthält das Wort weniger a als b, wenn $v^0 = \epsilon$ nutzt

$$x = \overbrace{a a a \dots a}^n \overbrace{b b b \dots b}^n$$

$\underbrace{\hspace{1.5cm}}_u \quad \underbrace{\hspace{1.5cm}}_v \quad \underbrace{\hspace{1.5cm}}_w$
 (da $|uv| \leq n \rightarrow v$ besteht nur aus a)

• Pumping Lemma(kontextfrei) - Anwendung

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

– Beispiel:

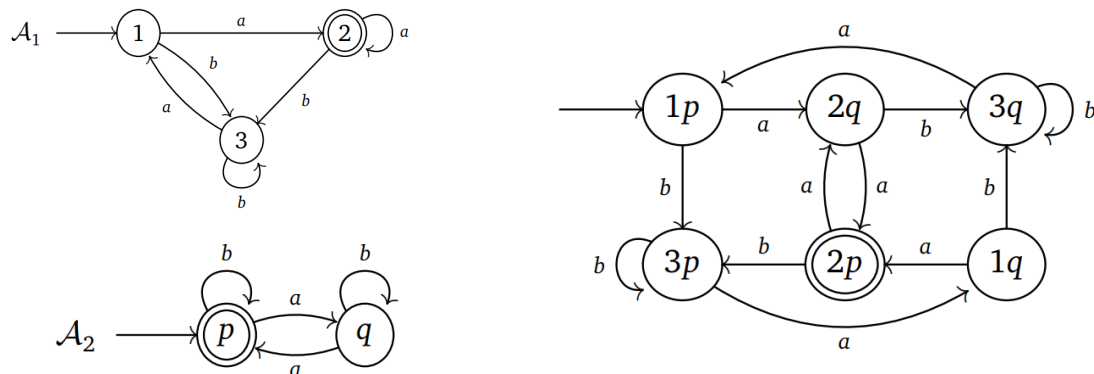
- * $L = \{a^n b^n c^n : n \in \mathbb{N}\}$
 - * Sei $n \in \mathbb{N}$ beliebig:
 - * Wähle $x = a^n b^n c^n$:
 - * Zerlegung ...
 - * Fall 1: uvw hat kein c. $m = 2$: $yu^m v w^m z$ hat mehr a/b als c $\Rightarrow yu^m v w^m z \notin L$
 - * Fall 2: uvw hat kein a. $m = 2$: $yu^m v w^m z$ hat mehr b/c als a $\Rightarrow yu^m v w^m z \notin L$
 - * (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: $L = \{w \in \{a, b\}^* \mid w \text{ hat mehr } b \text{ als } a\}$
 - * $k < n$
 - * $b^n a^k \in L$ (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

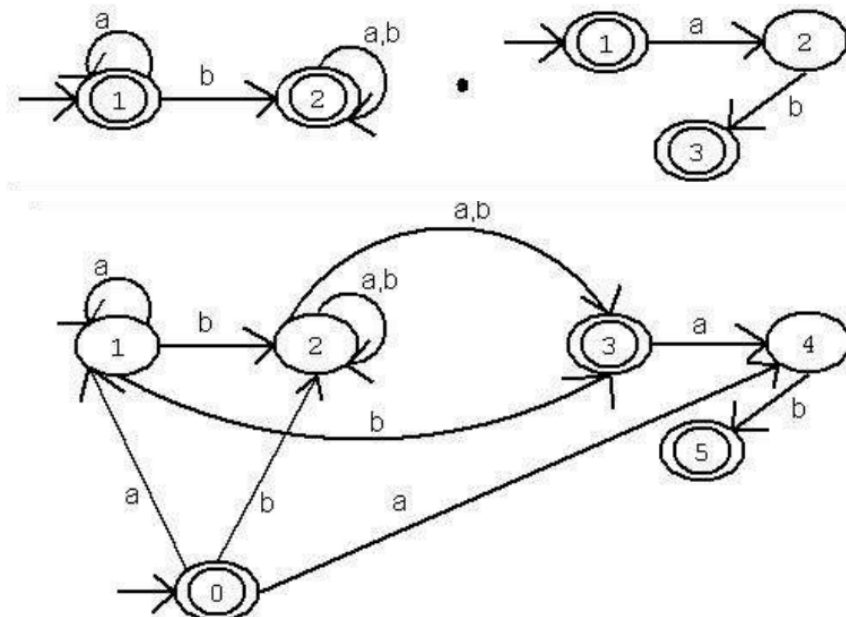
Algorithmen / Rechenmuster

• 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: $1 \xrightarrow{a} 2$ und $p \xrightarrow{a} q \Rightarrow 2q$
- 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

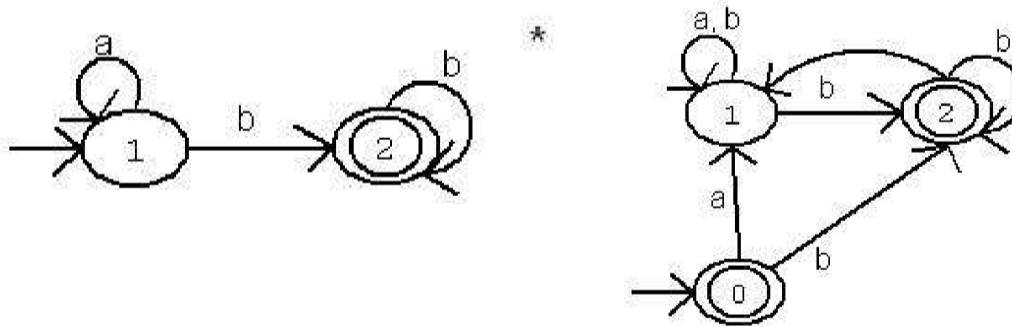
• Konkatenations-Automat



- 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



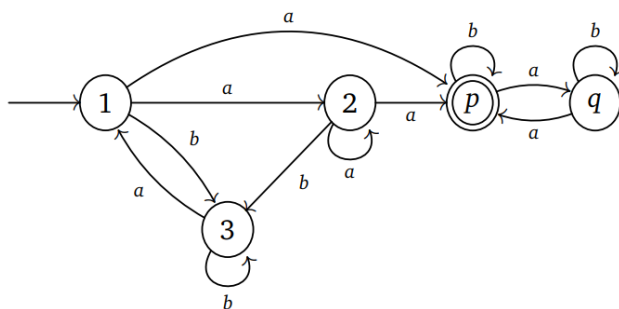
- 1. Einbauen eines neuen akzeptierenden Startzustands
- 2. Neuer Startzustand enthält alle Transitionen des alten Starts
- 3. Lenkung aller Transitionen, die auf akzept. Zustand zeigen auf den alten Start (oder neuer Start)
- (auch Selbstschleifen)

- Komplementbildung eines Automaten \bar{L}

- Wechsel der akzeptierenden und nicht akzeptierenden Zustände

- Potenzmengenautomat (NFA \rightarrow 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 \rightarrow regulärer Ausdruck)**

- Durchführen von k Iterationschritten zum Erhalten des regulären Ausdrucks
- k = Anzahl der Zustände
- Rekursionsformel: $a_{l,m}^{k+1} = a_{l,m}^k + a_{l,k+1}^k (a_{k+1,k+1}^k)^* a_{k+1,m}^k$

– **Durchführung**

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

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

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

– Tipps: (in k -ter Tabelle)

- * $a_{l,m}^k$: entspricht Werten in vorheriger Tabelle an selber Stelle
- * $a_{l,k+1}^k$: entspricht jeweils den Werten der k -ten Spalte in vorheriger Tabelle
- * $(a_{k+1,k+1}^k)^*$: ist für die ganze Tabelle der selbe Wert (k -te Zeile, k -te Spalte)
- * $a_{k+1,m}^k$: 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 \rightarrow 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
	4	p_3	p_2
$p_2^{(2)}$	5	p_3	p_2
	2	p_4	p_2
$p_3^{(2)}$	6	p_4	p_2
	3	p_4	p_4
$p_4^{(2)}$	7	p_4	p_4
	8	p_4	p_4

• Umformung kontextfreier Grammatik in Chomsky-Normalform

– **Bedingung:** Alle Produktionen in Form: $X \rightarrow YZ$ oder $A \rightarrow a$

– **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

$$\begin{array}{ll}
 P: X_0 \rightarrow aXY | bXb | a & X_0 \rightarrow aXY | bXb | a | aY | bb \\
 X \rightarrow aXa | bY | \epsilon & X \rightarrow aXa | bY | aa \\
 Y \rightarrow bX_0a | aX_0 & Y \rightarrow bX_0a | aX_0
 \end{array}$$

- * 2. Variablen für Terminale einfügen ($A \rightarrow a, B \rightarrow B$)
- * 3. Kettenproduktionen eliminieren ($X \rightarrow Y$)
- * 4. Eliminiere $X \rightarrow X_0, \dots, X_k$ mit $k \geq 3$ (Mehr als 2 Variablen aufteilen)
- * Aus $A \rightarrow ABC$ wird $A \rightarrow AD$ und $D \rightarrow BC$

– **Beispiel im Anhang**

• CYK Algorithmus

– Bestimmung des Wortproblems für kontextfreie Grammatiken in CNF

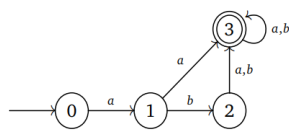
– **Durchführung**

- * 1. Aufstellen einer Tabelle (Größe $n \times n$ (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 \rightarrow Wort wird durch Grammatik erkannt

– Ableitungsbaum ausgehend von S erstellen, falls benötigt

• Reguläre Grammatiken \Leftrightarrow Automaten

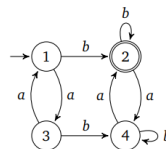
– Ziemlich selbstverständlich anhand Beispielen



$$\begin{array}{ll}
 X_0 \rightarrow aX_1 \\
 X_1 \rightarrow aX_3 | bX_2 \\
 X_2 \rightarrow aX_3 | bX_3 \\
 X_3 \rightarrow aX_3 | bX_3 | \epsilon
 \end{array}$$

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

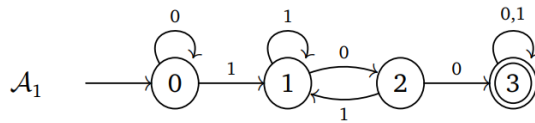
$$\begin{array}{ll}
 P: X_1 \rightarrow aX_3 | bX_2 \\
 X_2 \rightarrow aX_4 | bX_2 | \epsilon \\
 X_3 \rightarrow aX_1 | bX_4 \\
 X_4 \rightarrow aX_2 | bX_4
 \end{array}$$



Automatenbeispiele

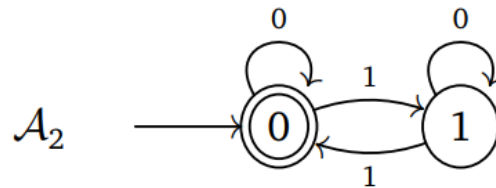
- L: Wörter, in denen 100 als Teilwort vorkommt

– $a = (0 + 1)^*100(0 + 1)^*$



- L: Wörter mit gerader Anzahl von 1

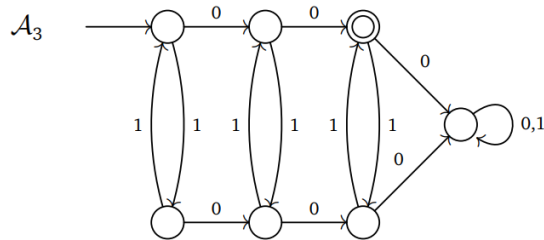
– $a = (0 + 10^*1)^*$



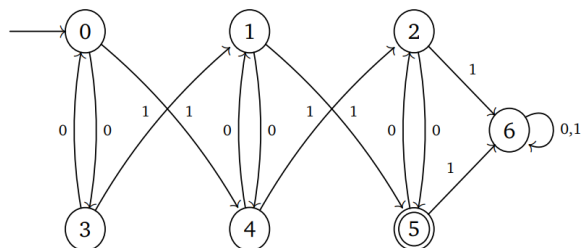
- L: Wörter mit gerader Anzahl von 1 und genau zweimal 0

– Mit $\alpha = 1(11)^*$ und $\beta = (11)^*$

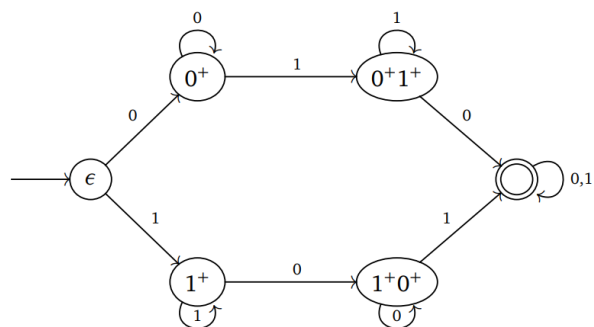
– $a = \alpha 0 \alpha 0 \beta + \alpha 0 \beta 0 \alpha + \beta 0 \alpha 0 \alpha + \beta 0 \beta 0 \beta$



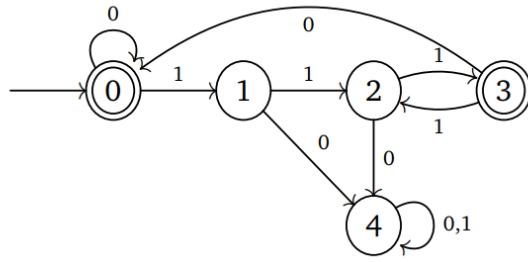
- L: Wörter von ungerader Länge mit genau zwei 1



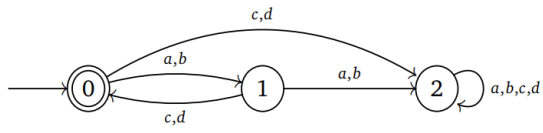
- L: Wörter, die 01 und 10 als (nicht notwendigerweise disjunkte) Teilwörter enthalten



- 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. $G = (\{a, b, c\}, \{S, X, Y\}, P, S)$ mit

$$\begin{aligned} P: \quad S &\rightarrow aba|XabaY|Xaba|abaY \\ X &\rightarrow bc|bcX|a|aX \\ Y &\rightarrow ac|acY|b|bY. \end{aligned}$$

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

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

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

Aufgabe H6.1 (Kontextfreie Grammatiken)

Geben Sie kontextfreie Grammatiken an, die folgende Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$ erzeugen:

- $L_1 := \{a^n b^m \mid m \leq n \leq 2m\}$.
- $L_2 := \{ucv \mid u, v \in \{a, b\}^*, |u| = |v|\}$.
- $L_3 := \{a^i b^j c^k \mid i + j = k\}$
- $L_4 := \{a^i b^j c^k \mid j = i + k\}$
- $L_5 := \{a^i b^j c^k \mid i = j \text{ oder } i = k\}$
- $L_6 := \emptyset$

Lösung: Je 2 P.:

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

$$P: \quad S \rightarrow aSb|aaSb|\varepsilon$$

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

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

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

$$\begin{aligned} P: \quad S &\rightarrow aSc|X \\ X &\rightarrow bXc|\varepsilon \end{aligned}$$

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

$$\begin{aligned} P: \quad S &\rightarrow aXbY|XbYc|\varepsilon \\ X &\rightarrow aXb|\varepsilon \\ Y &\rightarrow bYc|\varepsilon \end{aligned}$$

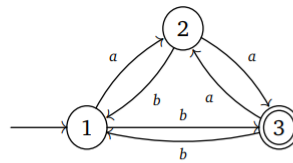
- $G_5 = (\Sigma, \{S, X, Y, U, V\}, P, S)$ mit

$$\begin{aligned} P: \quad S &\rightarrow XY|U \\ X &\rightarrow aXb|\varepsilon \\ Y &\rightarrow cY|\varepsilon \\ U &\rightarrow aUc|V \\ V &\rightarrow bV|\varepsilon \end{aligned}$$

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

$$P: \quad S \rightarrow S$$

Satz von Kleene - Beispiel



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

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

Mit der Rekursionsformel

$$\alpha_{\ell,m}^{k+1} = \alpha_{\ell,m}^k + \alpha_{\ell,k+1}^k (\alpha_{k+1,k+1}^k)^* \alpha_{k+1,m}^k$$

ergibt sich im nächsten Schritt folgende Tabelle mit Ausdrücken $\alpha_{\ell,m}^1$, 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 $\alpha_{1,3}^3$ wie folgt bestimmen: (2 Punkte)

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

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

$$\begin{aligned} P: \quad X_0 &\rightarrow aXY \mid bXb \mid a \\ X &\rightarrow aXa \mid bY \mid \varepsilon \\ Y &\rightarrow bX_0a \mid aX_0 \end{aligned}$$

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

Lösung: 1. Schritt (eliminiere ε -Produktionen) 4 P:

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

2. Schritt (Variablen vor Buchstaben) 4 P:

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

3. Schritt (eliminiere $X \rightarrow Y$ und $X \rightarrow X_0 \dots X_k$ mit $k \geq 3$) 4 P:

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

Quizfragen

L ist regulär	<input type="checkbox"/> \Rightarrow	L ist endlich	
	<input checked="" type="checkbox"/> \Leftarrow		
	<input checked="" type="checkbox"/> \Rightarrow	L wird von einem DFA akzeptiert	Kleene
	<input checked="" type="checkbox"/> \Leftarrow		
	<input checked="" type="checkbox"/> \Rightarrow	L wird von einem NFA akzeptiert	Kleene
	<input checked="" type="checkbox"/> \Leftarrow		
	<input checked="" type="checkbox"/> \Rightarrow	$\Sigma^* \setminus L$ ist regulär	
	<input checked="" type="checkbox"/> \Leftarrow		
	<input type="checkbox"/> \Rightarrow	L^* ist regulär	
	<input type="checkbox"/> \Leftarrow		
	<input checked="" type="checkbox"/> \Rightarrow	L enthält eine reguläre Sprache, d.h. es gibt eine reguläre Sprache $L_1 \subseteq \Sigma^*$ mit $L_1 \subseteq L$	(\emptyset)
	<input type="checkbox"/> \Leftarrow		
	<input checked="" type="checkbox"/> \Rightarrow	L ist Teilmenge einer regulären Sprache, d.h. es gibt eine reguläre Sprache $L_2 \subseteq \Sigma^*$ mit $L \subseteq L_2$	(Σ^*)
	<input type="checkbox"/> \Leftarrow		

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

Lösung: Richtig. Jede reguläre Sprache ist auch kontextfrei.

- (b) Jede kontextfreie Sprache ist kontextsensitiv.

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

$$\begin{aligned} \{ & (q_a, \#, \varepsilon, \varepsilon, q_a) \\ & (q_a, \#, a, |, q_a) \\ & (q_a, |, a, ||, q_a) \\ & (q_a, |, b, \varepsilon, q_b) \\ & (q_a, |, c, \varepsilon, q_c) \\ & (q_b, |, b, \varepsilon, q_b) \\ & (q_b, |, c, \varepsilon, q_c) \\ & (q_c, |, c, \varepsilon, q_c) \} \end{aligned}$$

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.