CFG → CNF   ! $\varepsilon \in L(G)$ geht verloren !

1. ε-Regeln entfernen
2. Kettenregeln entfernen
3. Nichtterminale hinzufügen
4. Verkürzen

$S \to aSTbb \mid b$
$T \to aSa$

1. ✓
2. ✓

3. füge jedem Terminalzeichen ein neues Nichtterminal hinzu und modifiziere Produktionen entsprechend

$S \to ASTBB \mid b$ ← hier Ersetzung nicht nötig
$T \to ASA$
$A \to a$
$B \to b$

4. spalte Produktionen mit rechte Seite länger als 2 auf und füge neue Nichtterminale ein

$\underbrace{\begin{array}{l} S \to AX_1 \mid b \\ X_1 \to SX_2 \\ X_2 \to TX_3 \\ X_3 \to BB \end{array}}_{S \to ASTBB}$ $\underbrace{\begin{array}{l} T \to AX_4 \\ X_4 \to SA \end{array}}_{T \to ASA}$ $\begin{array}{l} A \to a \\ B \to b \end{array}$

$S \to aUb \mid TT$
$R \to bbS \mid RSUa$
$T \to bR \mid \varepsilon$
$U \to STS \mid ab$

1. bestimme Menge der Nichtterminale, aus denen man ε ableiten kann

$M = \{\underbrace{T}_{\text{direkt}}, \underbrace{S}_{\substack{S \to TT \\ T \to \varepsilon \\ \Rightarrow S \to^* \varepsilon}}, \underbrace{U}_{U \to STS}\}$

1.2 entferne ε-Regeln direkt

$S \to aUb \mid TT$

$R \to bbS \mid RSUa$
$T \to bR$
$U \to STS \mid ab$

(1.3) baue Abkürzungsregeln für alle Produktionen, wo rechts ein Symbol aus M steht

→ Kopie von aUb ohne U

$S \to aUb \mid TT \mid ab \mid T$
$R \to bbS \mid RSUa \mid bb \mid RUa \mid RSa \mid Ra$
$T \to bR$
$U \to STS \mid ab \mid TS \mid SS \mid ST \mid S \mid T$

jede Kombination von weg- und nicht weglassen

exponentiell!

$S \to Ta \mid T$
$T \to bTb \mid a \mid R \mid U$
$R \to aSb \mid b$
$U \to ab \mid ST$

(2) bestimme Menge von Paaren von Kettenregeln

$M = \{ \underbrace{(S,T), (T,R), (T,U)}_{\text{direkt}}, \underbrace{(S,R), (S,U)}_{\text{transitiv}} \}$

(2.1) entferne Kettenregeln

$S \to Ta$
$T \to bTb \mid a$
$R \to aSb \mid b$
$U \to ab \mid ST$

(2.2) baue Abkürzungsregeln für alle Paare aus M → kopiere rechte Seiten ohne Kettenregeln

$S \to Ta \mid \underbrace{bTb \mid a}_{S \to T} \mid \underbrace{aSb \mid b}_{S \to R} \mid \underbrace{ab \mid ST}_{S \to U}$
$T \to bTb \mid a \mid \underbrace{aSb \mid b}_{T \to R} \mid \underbrace{ab \mid ST}_{T \to U}$
$R \to aSb \mid b$
$U \to ab \mid ST$

CYK-Algorithmus → Wortproblem für CNF

$S \to AB \mid CD \mid AD$
$C \to AB \mid SC$
$D \to BC \mid AD \mid BB$
$A \to a$
$B \to b$

$w = abbab$

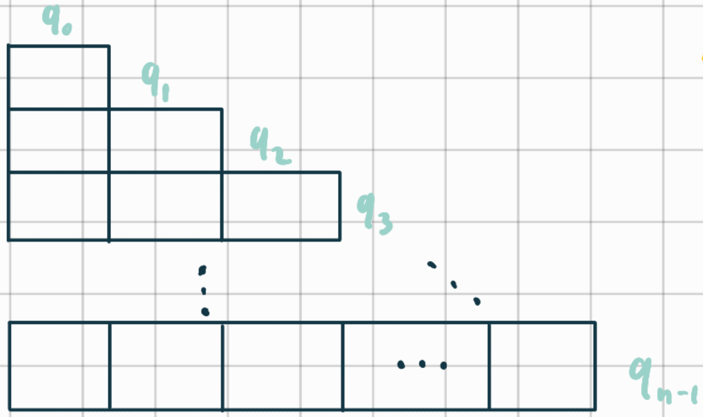
$abbab \in L(G)$?

Startsymbol oben
$\Rightarrow w \in L(G)$

Zelle: Nichtterminale, die ein Teilwort ableiten

DFA-Minimierung

- entferne alle aus q_0 unerreichbaren Zustände
- berechne äquivalenten Zustände
- fasse äquivalente Zustände zusammen



Äquivalente Zustände

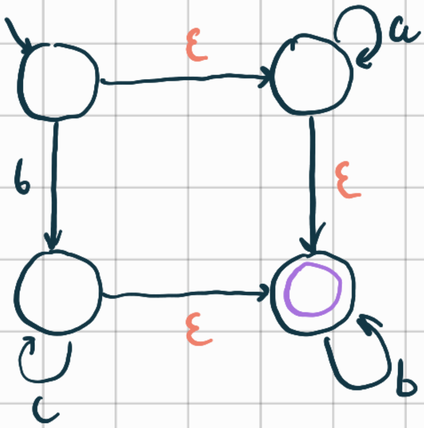
- markiere alle Zellen mit einem Endzustand und einem Nichtendzustand mit ε
- für alle anderen:
  gibt es ein Wort, so dass ich aus einem Zustand in einen Endzustand komme und mit dem anderen nicht? → ja: markiere
  sonst: $\equiv$

ε-NFA → NFA

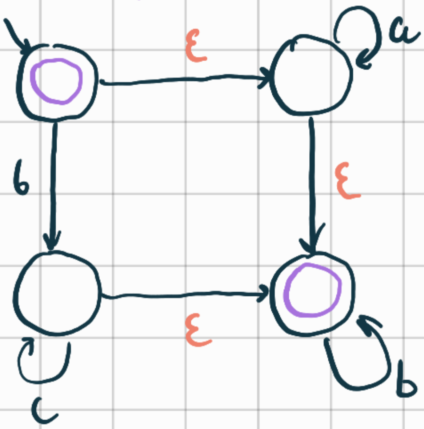
Schritt 1 Kann man nur durch ε-Übergänge den Endzustand erreichen? $\xrightarrow{y}$ Startz. wird Endz.

Schritt 2 Eliminiere ε-Übergänge schrittweise

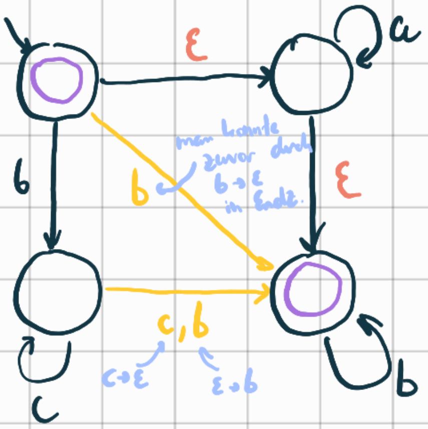
wozu war die ε-Kante nützlich?

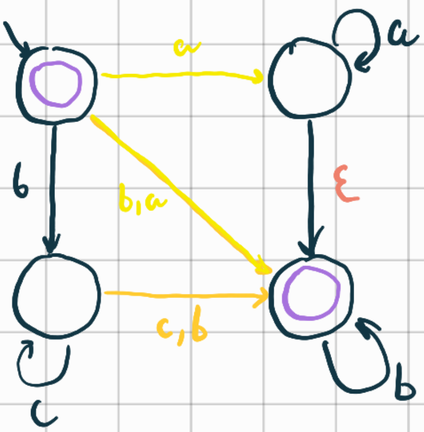


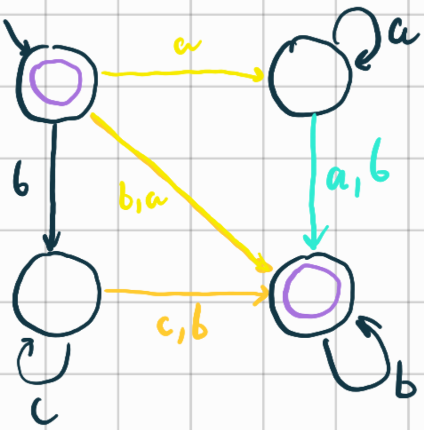
Schritt 1: ja!



Schritt 2:







nützlich, erzeugend, erreichbar   G CFG

X ist...

nützlich $\leftrightarrow$ $S \xrightarrow{*}_G w \in \Sigma^*$

X kommt in der Ableitung vor

erzeugend $\leftrightarrow$ $X \xrightarrow{*}_G w \in \Sigma^*$

erreichbar $\leftrightarrow$ $S \xrightarrow{*}_G \alpha X \beta$

$S \to AB \mid a$
$A \to b$

① Elimination nicht erzeugender Symbole + Produktionen

Idee: markiere Nichtterminale, die in genau n Schritten ein Wort erzeugen

$P_0 = \{S, A\}$
$P_1 = \{S\} \cup \{S, A\} = P_0$ → B nicht erzeugend → ~~AB~~

→ $S \to a$
$\cancel{A \to b}$

② Elimination unerreichbarer Symbole + Produktionen

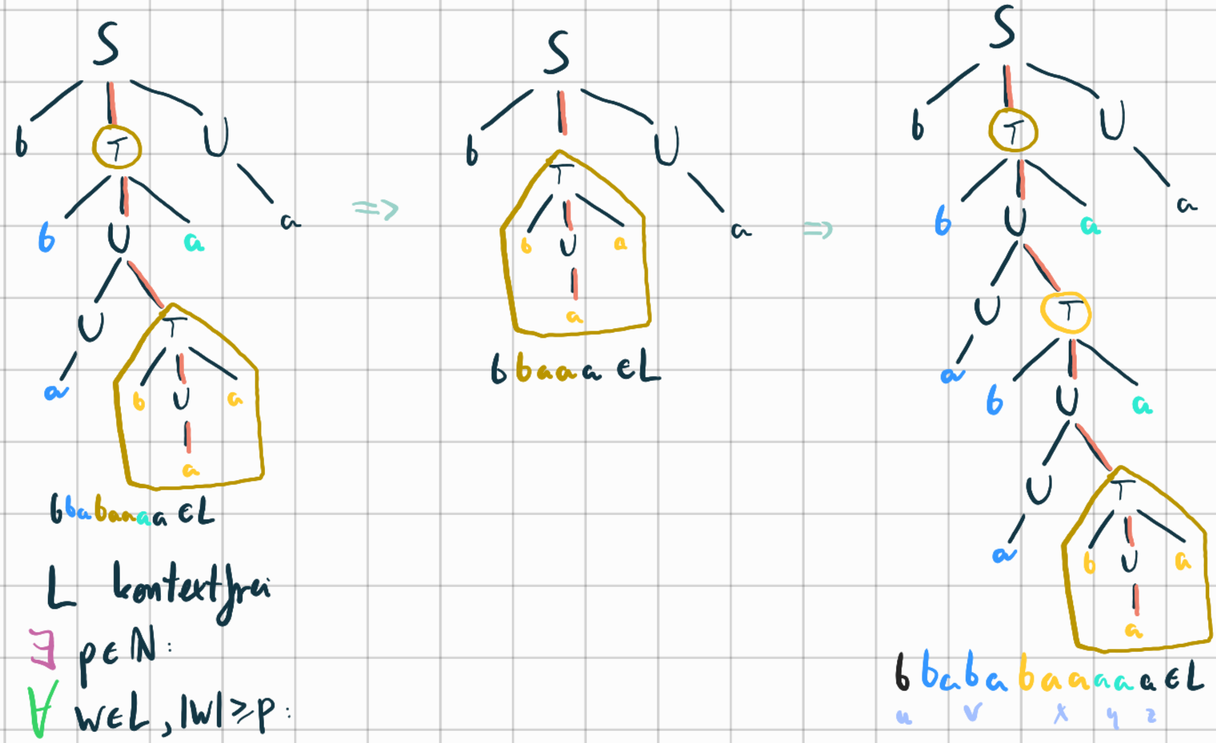
$S \to a$

Pumping-Lemma (CFL)

$G = (V, \Sigma, P, S)$
$V = \{S, T, U\}$
$\Sigma = \{a, b\}$

$$P = \begin{cases} S \to bTU \mid b \\ T \to bUa \\ U \to UT \mid a \end{cases}$$

Idee: 5 Schritte für größte Tiefe
3 Nichtterminale
$\Rightarrow$ mind. 1 Nichtterminal doppelt!



Wenn L kontextfrei
Dann $\exists\, p \in \mathbb{N}$:
$\forall\, w \in L, |w| \geq p$:
$\exists$ Zerlegung $w = uvxyz$
mit $v \neq \varepsilon$ oder $y \neq \varepsilon$
und $|vxy| \leq p$
$\forall\, i \in \mathbb{N}$: $uv^ixy^iz \in L$

Beweisschema (Pumping-Lemma → L nicht kontextfrei)

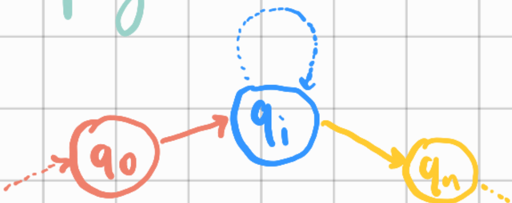
Wenn $\forall\, p \in \mathbb{N}$:
$\exists\, w \in L, |w| \geq p$:
$\forall$ Zerlegung $w = uvxyz$
mit $v \neq \varepsilon$ oder $y \neq \varepsilon$
und $|vxy| \leq p$
$\exists\, i \in \mathbb{N}$: $uv^ixy^iz \notin L$
Dann L nicht kontextfrei

$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$
Sei $p \in \mathbb{N}$ beliebig.
Wähle $a^p b^p c^p \in L$.

überschneiden sich

Fall 1: $|vxy|_c = 0$
Wähle $i = 2$, dann hat uv^2xy^2z mehr as oder bs als cs.
$\Rightarrow uv^2xy^2z \notin L$.
Fall 2: $|vxy|_a = 0$
Wähle $i = 2$, dann hat uv^2xy^2z mehr bs oder cs als as.
$\Rightarrow uv^2xy^2z \notin L$

Pumping-Lemma (reg. Sprachen)



aaa aaa aaa bbbbbbbbb $\in L$, akzeptiert
aaa aaa aaa aaa bbbbbbbbb $\notin L$, akzeptiert
aaa aaa bbbbbbbbb $\notin L$, akzeptiert

Idee: $\{a^n b^n\}$ & Automat mit n Zuständen, irgendein Zustand muss doppelt erreicht werden → Schleife, die man beliebig oft wiederholen darf → akzeptiert, aber $\notin L$

Bsp.

Wenn L regulär
Dann $\exists p \in \mathbb{N}$: → PL-Zahl
$\forall w \in L, |w| \geq p$:
$\exists$ Zerlegung $w = xyz$
mit $y \neq \varepsilon$
und $|xy| \leq p$:
$\forall i \in \mathbb{N}$: $xy^iz \in L$

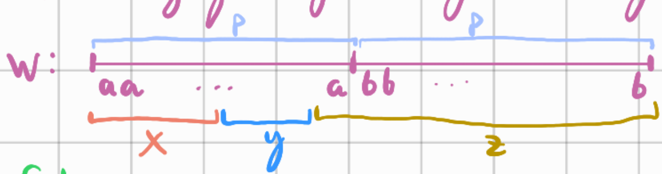
→ wenn L die Eigenschaft nicht hat, ist L <u>nicht</u> regulär!

Beweisschema (Pumping-Lemma → L nicht regulär)

Wenn $\forall p \in \mathbb{N}$:
$\exists w \in L, |w| \geq p$:
$\forall$ Zerlegung $w = xyz$
mit $y \neq \varepsilon$
und $|xy| \leq p$:
$\exists i \in \mathbb{N}$: $xy^iz \notin L$
Dann L nicht regulär

$L = \{a^n b^n \mid n \geq 0\}$

gegeben: Sei $p \in \mathbb{N}$ gegeben.
gewählt: Wähle $w = a^p b^p$
gegeben: Sei Zerlegung $w = xyz$ mit $y \neq \varepsilon$ und $|xy| \leq |p|$ gegeben.

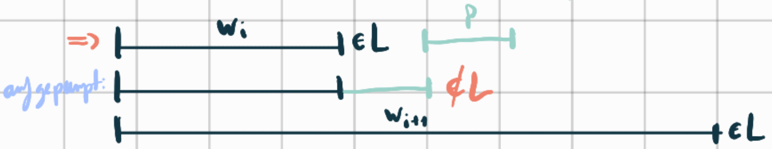


gewählt: Setze $i = 0$.
Dann hat $|xy^0z|_a = |xz|_a < |xz|_b$.
Somit $xy^iz \notin L$.

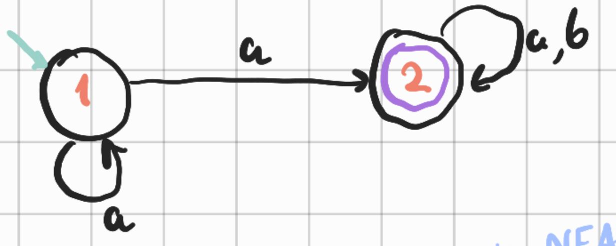
TIPP: Wenn eine Sprache beliebig große Lücken bzgl. Wortlänge hat, ist sie <u>nicht</u> regulär!

Beispiel $L = \{a^{n^2} \mid n \in \mathbb{N}\}$ $L = \{a^{n!} \mid n \in \mathbb{N}\}$ $L = \{a^{2^n} \mid n \in \mathbb{N}\}$

$a \in L$
$aaaa \in L$ > 2
$aaaaaaaaa \in L$ > 4



Potenzmengenkonstruktion NFA → DFA



$Q = \{1, 2\}$

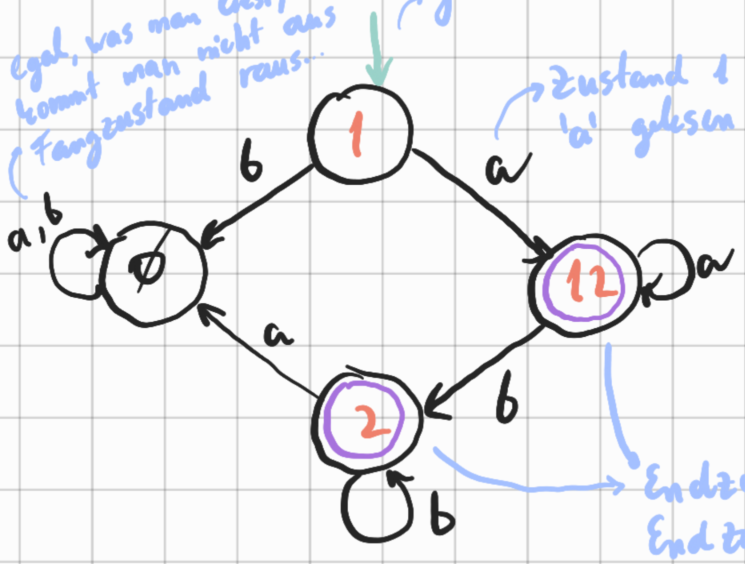
$\mathcal{P}(Q) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$

$|\mathcal{P}(Q)| = 2^{|Q|}$ exponentiell!

egal, was man liest, kommt man nicht aus Fangzustand raus…

gleich wie NFA

Zustand 1 im NFA
'a' gelesen } => 1 oder 2



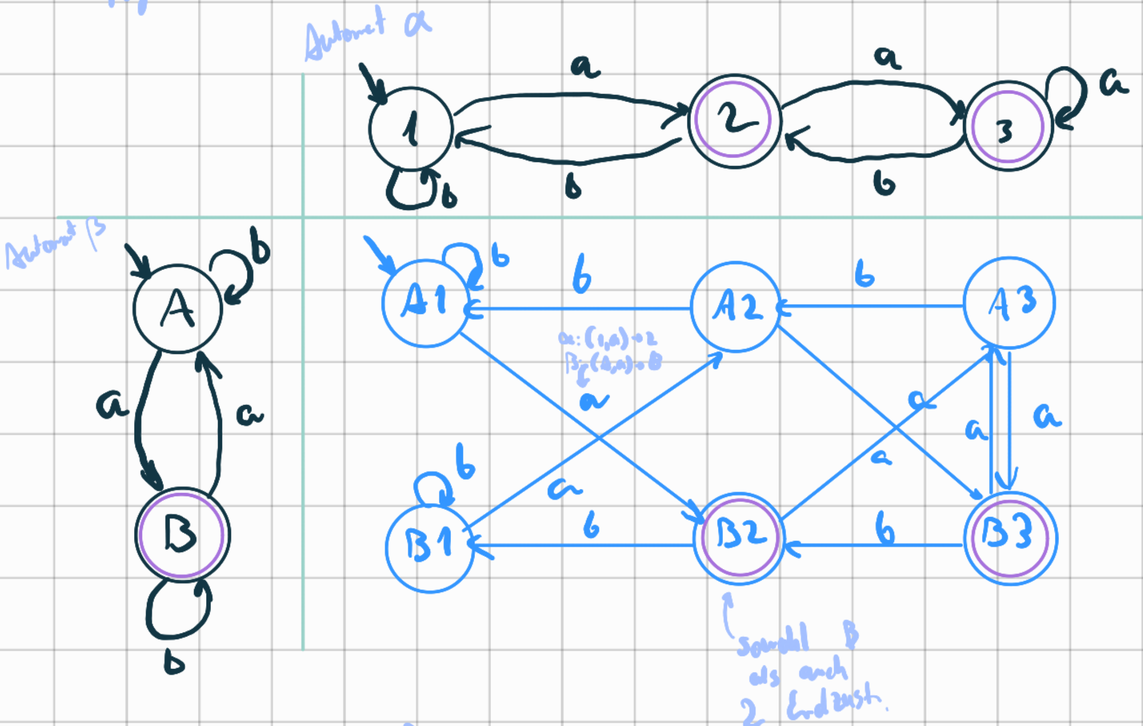
$\delta(1,a) \cup \delta(2,a)$

Endzustände sind die, die Endzustände des NFAs enthalten

Produktautomat

polynomiell.

$L(G_1)$ erkennbar, $L(G_2)$ erkennbar $\Big\} \Rightarrow L(G_1) \cap L(G_2)$ erkennbar



· für $L(G_1) \cup L(G_2)$: B1, A2, A3 auch Endzustände

RegEx → ε-NFA

$(a^* | bc)^*$

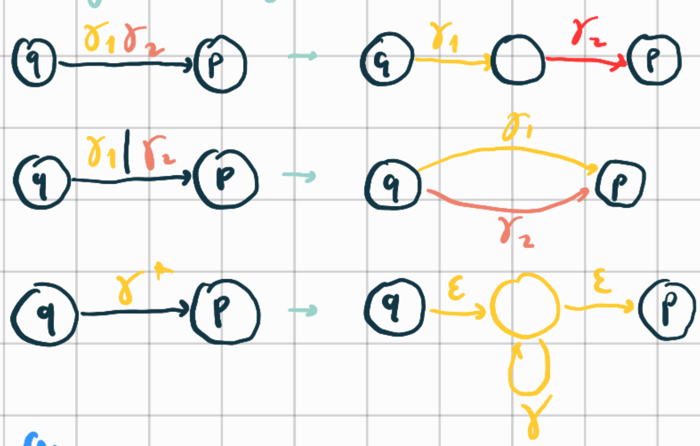
bsp nicht unbedingt vorlesungstreu...

Preprocessing γ regex

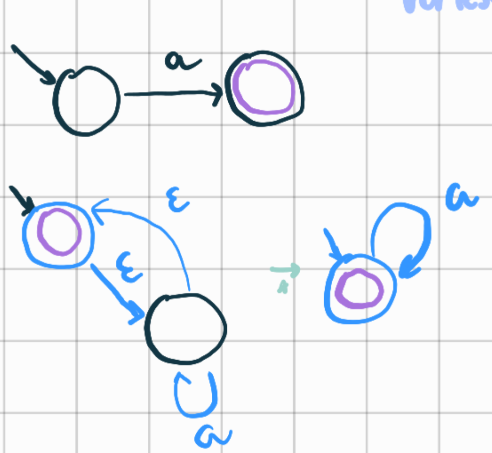
$\gamma\emptyset \to \emptyset$ $\quad$ $\gamma | \emptyset \to \gamma$ $\quad$ $\emptyset^* \to \varepsilon$

$\emptyset\gamma \to \emptyset$ $\quad$ $\emptyset | \gamma \to \gamma$

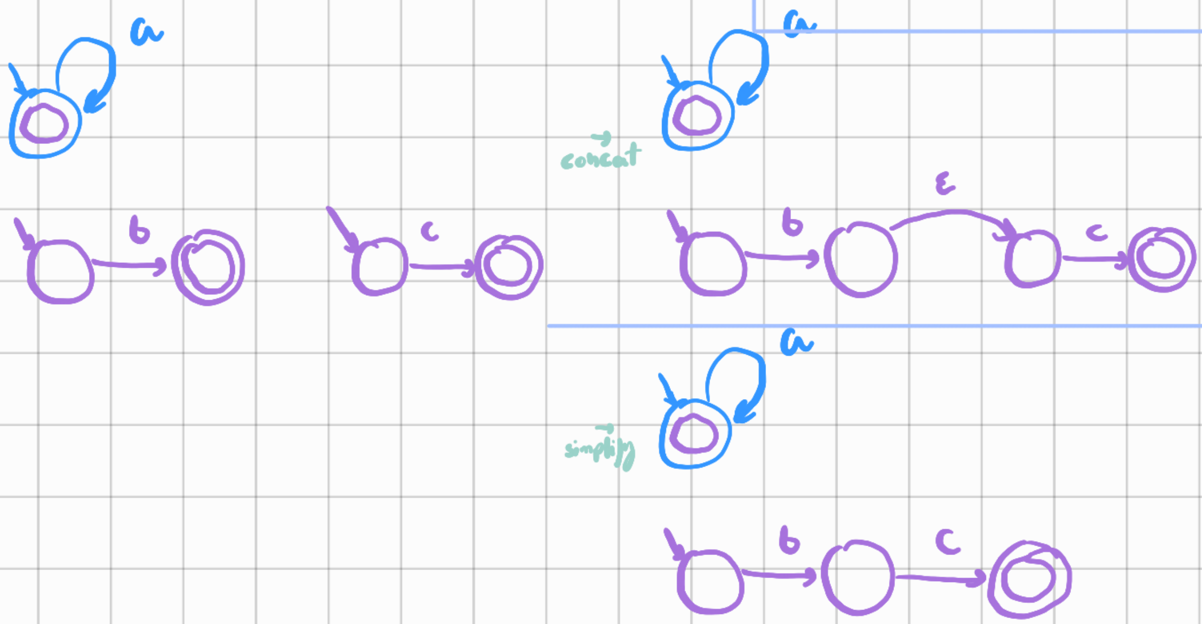
Transformationsregeln



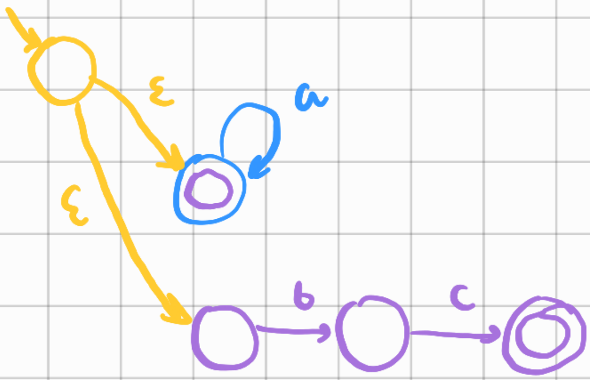
a^*



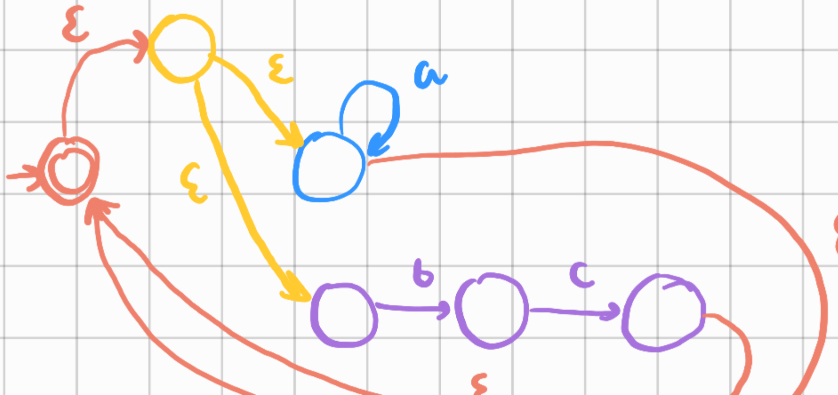
a^* bc



$a^* | bc$



$(a^* | bc)^*$

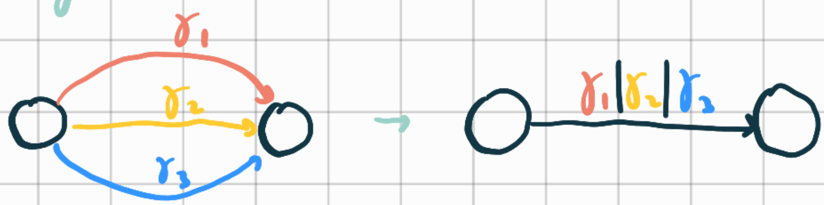


alternativ

ε-NFA → RegEx

Preprocessing

Transformation (wdh. so lange wie möglich)

- 

- wähle Zustand q weder Startzustand noch Endzustand
- IF q hat Schleife σ: $((p,\gamma,q),(q,\beta,p')) \rightarrow (p,\gamma\sigma^*\beta,p')$
  $p \neq q \neq p'$

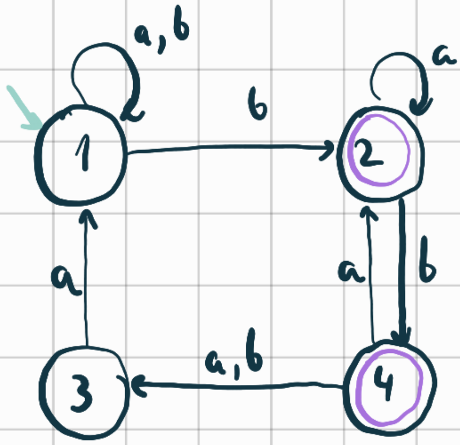
ELSE q hat keine Schleife: $((p,\gamma,q),(q,\beta,p')) \rightarrow (p,\gamma\beta,p')$
$p \neq q \neq p'$

NFA → RegEx

$X \equiv \alpha X | \beta \implies X \equiv \alpha^* \beta$

$\varepsilon \notin L(\alpha)$ regex

NICHT FÜR ε-NFAs!



Gleichungssystem

$L_1 = \underbrace{(a|b)}_{\text{Kante}}\underbrace{L_1}_{q} | bL_2$

$L_2 = aL_2 | bL_4 | \varepsilon$ Endzustand

$L_3 = aL_1$

$L_4 = aL_2 | (a|b)L_3 | \varepsilon$

Lösen des Gleichungssystems standardmäßig

$L_1 = (a|b)L_1 | bL_2$

$L_2 = aL_2 | bL_4 | \varepsilon$

$L_3 = aL_1$

$L_4 = aL_2 | (a|b)L_3 | \varepsilon$

→ $L_1 = (a|b)L_1 | bL_2$

$L_2 = aL_2 | bL_4 | \varepsilon$

$L_4 = aL_2 | (a|b)aL_1 | \varepsilon$

→ $L_1 = (a|b)L_1 | bL_2$

$L_2 = aL_2 | b(aL_2 | (a|b)aL_1 | \varepsilon) | \varepsilon$

AL → $L_1 = (a|b)^* bL_2$

$L_2 = aL_2 | b(aL_2 | (a|b)aL_1 | \varepsilon) | \varepsilon$

von hier aus improv...

→ $L_2 = aL_2 | b(aL_2 | (a|b)a((a|b)^* bL_2) | \varepsilon) | \varepsilon$

$= aL_2 | baL_2 | b(a|b)a((a|b)^* bL_2) | b | \varepsilon$

$= aL_2 | baL_2 | b(a|b)a(a|b)^* bL_2 | b | \varepsilon$

→ $L_2 = a^*(baL_2 | b(a|b)a(a|b)^* bL_2 | b | \varepsilon)$

$= a^*baL_2 | a^*b(a|b)a(a|b)^* bL_2 | a^*b | a^*$

→ $L_2 = (a^*ba)^*(a^*b(a|b)a(a|b)^* bL_2 | a^*b | a^*)$

$= (a^*ba)^*a^*b(a|b)a(a|b)^* bL_2 | (a^*ba)^*a^*b | (a^*ba)^*a^*$

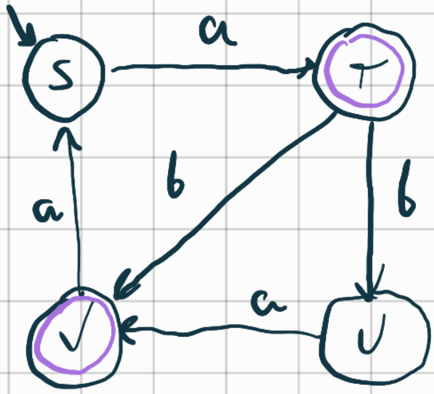
→ $L_2 = ((a^*ba)^*a^*b(a|b)a(a|b)^*b)^*((a^*ba)^*a^*b | (a^*ba)^*a^*)$

$= ((a^*ba)^*a^*b(a|b)a(a|b)^*b)^*(a^*ba)^*a^*b$

$| ((a^*ba)^*a^*b(a|b)a(a|b)^*b)^*(a^*ba)^*a^*$

i have no idea if this is correct

FA → CFG



→

$G = \{V, \Sigma, P, S\}$

$V = \{S, T, U, V\}$

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

$P = \begin{cases} S \to aT \\ T \to bU \mid bV \mid \varepsilon \\ U \to aV \\ V \to aS \mid \varepsilon \end{cases}$

Kante · nächste Zustand

Endzustand

$S = S$

CFG → PDA leerer Keller!

$S \to aSbb \mid bT$
$T \to Tba \mid Sb \mid \varepsilon$

Idee: $S \to aSbb \to abTbb \to abbb$

CFG →

← PDA

a
b
b
$\underline{S} \rightsquigarrow \underline{b} \rightsquigarrow \underline{\varepsilon}$

erzeuge Wort — prüfe Zeichen

Intuition...

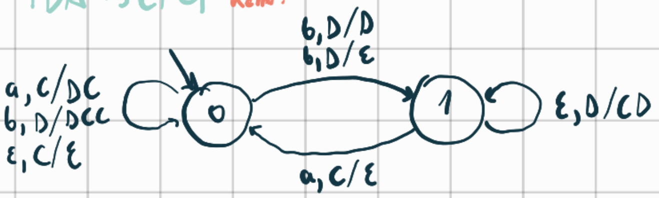
$\varepsilon, S/aSbb$
$\varepsilon, S/bT$
$\varepsilon, T/Tba$
$\varepsilon, T/Sb$
$\varepsilon, T/\varepsilon$
$a, a/\varepsilon$
$b, b/\varepsilon$

Schritt 1: w auf Stack schreiben
Schritt 2: arbeite Wort Zeichen für Zeichen ab } simultan!

w = abbb

$S \to \cancel{a}Sbb \to Sbb \to \cancel{b}Tbb \to \cancel{T}bb \to \cancel{b}b \to \cancel{b} \to \varepsilon$ ✓

PDA → CFG leerer Keller!



unterstes Kellerzeichen C

Idee: PDA hat Aufgabe, oberstes Stackzeichen irgendwann loszuwerden
formal: ∀ Höhe des Stacks n ∃ Moment in der Zukunft mit Stackhöhe n-1

Format: [i, X, j] = Automat will von Zustand i in Zustand j und X vom Stack löschen
Aufgabe
zwischendurch kann egal was passieren...

Nichtterminal

S → [0,C,0] | [0,C,1]   fange im Startzustand 0 an mit Stackzeichen C zu löschen

[0,C,0] → ε   ε, C/ε   von 0 nach 0 und lösche C, direkt möglich!
[1,C,0] → a   a, C/ε
[0,D,1] → b   b, D/ε
[0,D,0] → b[1,D,0]
[0,D,1] → b[1,D,1]   } b, D/D   neue Aufgabe: wir sind in Zustand 1 gekommen und müssen ein D loswerden!
[0,C,0] → a[0,D,0][0,C,0]
[0,C,0] → a[0,D,1][1,C,0]
[0,C,1] → a[0,D,0][0,C,1]
[0,C,1] → a[0,D,1][1,C,1]   } a, C/DC
[1,D,0] → [1,C,0][0,D,0]
[1,D,0] → [1,C,1][1,D,0]
[1,D,1] → [1,C,0][0,D,1]
[1,D,1] → [1,C,1][1,D,1]   } ε, D/CD

[0, D, 0] → 6 [0, D, 0] [0, C, 0] [0, C, 0]
[0, D, 0] → 6 [0, D, 0] [0, C, 1] [1, C, 0]
[0, D, 0] → 6 [0, D, 1] [1, C, 0] [0, C, 0]
[0, D, 0] → 6 [0, D, 1] [1, C, 1] [1, C, 0]
[0, D, 1] → 6 [0, D, 0] [0, C, 0] [0, C, 1]
[0, D, 1] → 6 [0, D, 0] [0, C, 1] [1, C, 1]
[0, D, 1] → 6 [0, D, 1] [1, C, 0] [0, C, 1]
[0, D, 1] → 6 [0, D, 1] [1, C, 1] [1, C, 1]

6, D/DCC