Algorithmen II
Vorlesung am 17.01.2013

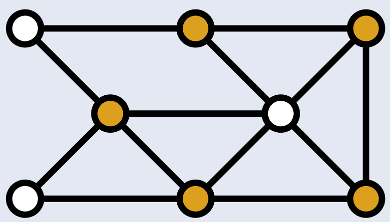
Parametrisierte Algorithmen

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Kernbildung mit Linearer Programmierung

Vertex Cover als ILP

Problem: Vertex Cover
Finde *Vertex Cover* $V' \subseteq V$ mit $|V'| \leq k$. V' heißt Vertex Cover genau dann, wenn für jede Kante $\{v, w\} \in E$ gilt $v \in V'$ oder $w \in V'$.
(ist $\mathcal{NP}$-Schwer)



Formulierung als ILP:

- Eine Variable x_v für jeden Knoten $v \in V$.
  $\rightarrow x_v = 1$ bedeutet $v \in V'$, $x_v = 0$ bedeutet $v \notin V'$.
- Minimiere also

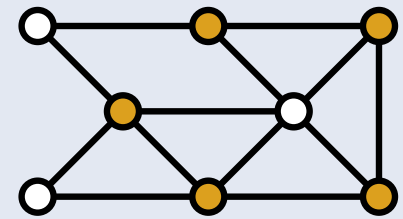
$$\sum_{x \in V} x_v$$

unter den Nebenbedingungen

$$\begin{aligned} x_v &\in \{0, 1\} && \text{für alle } v \in V \\ x_v + x_u &\geq 1 && \text{für alle } (u, v) \in E \end{aligned}$$

VERTEX COVER als ILP

Problem: VERTEX COVER
Finde *Vertex Cover* $V' \subseteq V$ mit $|V'| \leq k$. V' heißt Vertex Cover genau dann, wenn für jede Kante $\{v, w\} \in E$ gilt $v \in V'$ oder $w \in V'$. (ist $\mathcal{NP}$-Schwer)



Formulierung als ILP:

- Eine Variable x_v für jeden Knoten $v \in V$.
  $\rightarrow x_v = 1$ bedeutet $v \in V'$, $x_v = 0$ bedeutet $v \notin V'$.
- Minimiere also

$$\sum_{x \in V} x_v$$

  unter den Nebenbedingungen

$$x_v \in \{0, 1\} \qquad \text{für alle } v \in V$$
$$x_v + x_u \geq 1 \qquad \text{für alle } (u, v) \in E$$

Relaxierung des ILP's:
Ersetze Bedingung $x_v \in \{0, 1\}$ durch $0 \leq x_v \leq 1$.

Eigenschaften der Relaxierung

Lemma: Fast ganzzahlige Lösung **(Lemma 10.8)**

Das relaxierte LP hat eine optimale Lösung mit $x_v \in \{0, \frac{1}{2}, 1\}$ für alle $v \in V$. Eine solche Lösung kann in polynomieller Zeit berechnet werden.

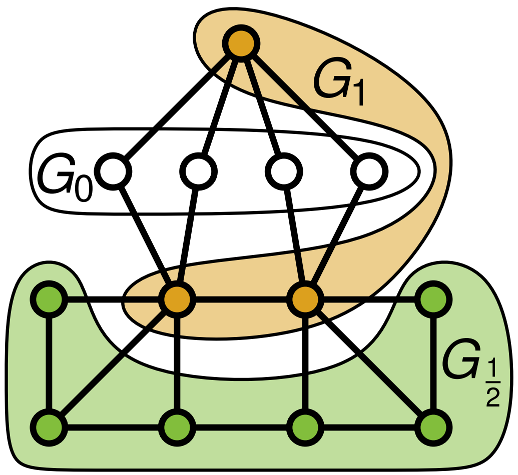
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Eigenschaften der Relaxierung

Lemma: Fast ganzzahlige Lösung **(Lemma 10.8)**
Das relaxierte LP hat eine optimale Lösung mit $x_v \in \{0, \frac{1}{2}, 1\}$ für alle $v \in V$. Eine solche Lösung kann in polynomieller Zeit berechnet werden.

Betrachte Optimallösung mit $x_v \in \{0, \frac{1}{2}, 1\}$ für alle $v \in V$.

- Sei $V_r = \{v \in V \mid x_v = r\}$ für $r \in \{0, \frac{1}{2}, 1\}$.
- Sei G_r der von V_r induzierte Graph.



Lemma: Zusammensetzbarkeit von Lösungen **(Lemma 10.9)**
Für den Wert vc$(\cdot)$ eines optimalen Vertex Covers gilt:

(i) $\text{vc}(G_{\frac{1}{2}}) \geq \frac{1}{2}|V_{\frac{1}{2}}|$

(ii) $\text{vc}(G_{\frac{1}{2}}) = \text{vc}(G) - |V_1|$

Der Kern des Problems

Lemma: Zusammensetzbarkeit von Lösungen **(Lemma 10.9)**

Für den Wert vc(·) eines optimalen Vertex Covers gilt:

(i) $\mathrm{vc}(G_{\frac{1}{2}}) \geq \frac{1}{2}|V_{\frac{1}{2}}|$

(ii) $\mathrm{vc}(G_{\frac{1}{2}}) = \mathrm{vc}(G) - |V_1|$

Folgerung: Kernbildung zu Instanz (G, k) von Vertex Cover:

(1) Löse LP mit Bedingung $x_v \in \{0, \frac{1}{2}, 1\}$ in Polynomialzeit.

(2) Setzte $k' = k - |V_1|$ (falls $k' < 0$, so gibt es kein Vertex Cover der Größe k).

(3) Falls $|V_{\frac{1}{2}}| > 2k'$, dann gibt es in $G_{\frac{1}{2}}$ kein Vertex Cover der Größe k' $\Rightarrow$ es gibt in G keines der Größe k.

(4) Andernfalls gib $(G_{\frac{1}{2}}, k')$ als Kern aus.

Beachte: $G_{\frac{1}{2}}$ enthält nur $O(k')$ $(k' \leq k)$ Knoten (wegen (3)).

$\Rightarrow$ Laufzeit zur Berechnung eines Vertex Covers der Größe k: $O(p(n) + f(k'))$

Dabei bezeichnet $f(k')$ die Laufzeit zur Berechnung eines Vertex Covers der Größe k' in $G_{\frac{1}{2}}$.

Erinnerung: Vertex Cover der Größe k in einem Graph mit n Knoten kann in $O(k \cdot n + 1.342^k \cdot k^2)$ Zeit berechnet werden.

Reguläre Sprachen

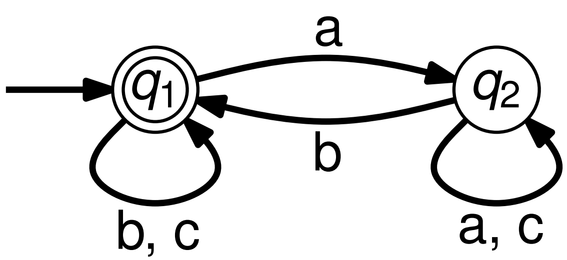
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erkennen regulärer Sprachen

Problem:
Gegeben seien ein Text T, sowie ein regulärer Ausdruck R. Ist T ein Wort der Sprache $L(R)$?

Beispiel: Regulärer Ausdruck
$R = (a(a \cup c)^\star b)^\star (b \cup c)^\star$
$\Rightarrow$ R beschreibt die Sprache $L(R)$ aller Wörter über dem Alphabet $\Sigma = \{a, b, c\}$ in denen nach jedem Auftreten eines a's auch noch ein b auftritt.

Repräsentation als endlicher Automat:
Test ob $T \in L(R)$: Simulation des Automaten.

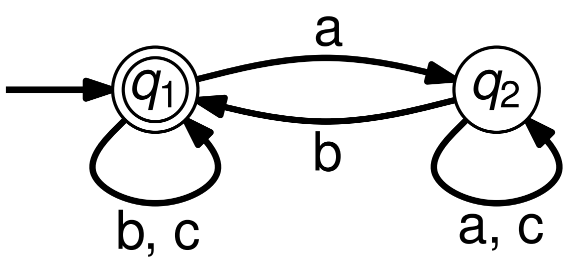


Erkennen regulärer Sprachen

Problem:
Gegeben seien ein Text T, sowie ein regulärer Ausdruck R. Ist T ein Wort der Sprache $L(R)$?

Beispiel: Regulärer Ausdruck
$R = (a(a \cup c)^\star b)^\star (b \cup c)^\star$
$\Rightarrow$ R beschreibt die Sprache $L(R)$ aller Wörter über dem Alphabet $\Sigma = \{a, b, c\}$ in denen nach jedem Auftreten eines a's auch noch ein b auftritt.

Repräsentation als endlicher Automat:
Test ob $T \in L(R)$: Simulation des Automaten.



Zu beantwortende Fragen:

- Wie konstruiert man (effizient) einen endlichen Automaten zu einem gegebenen regulären Ausdruck?
- Wie simuliert man ein Wort effizient?
- NEA oder DEA?

(NEA/DEA: nichtdeterministischer/deterministischer endlicher Automat)

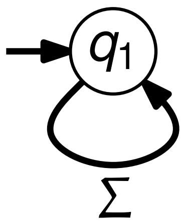
Regulärer Ausdruck $\rightarrow$ NEA

Rekursive Definition eines regulären Ausdrucks:

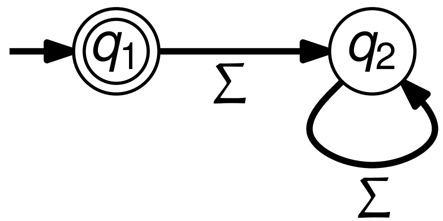
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

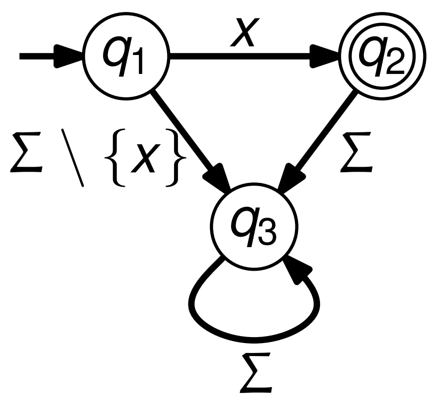
$\emptyset$:



ε:



x:



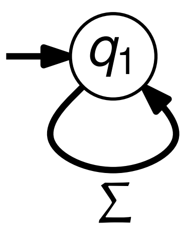
Regulärer Ausdruck $\rightarrow$ NEA

Rekursive Definition eines regulären Ausdrucks:

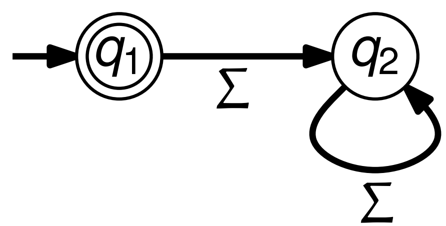
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

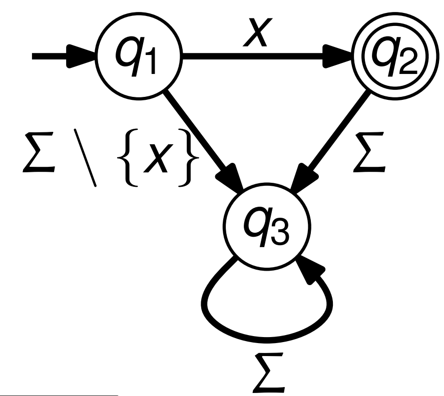
$\emptyset$:



ε:

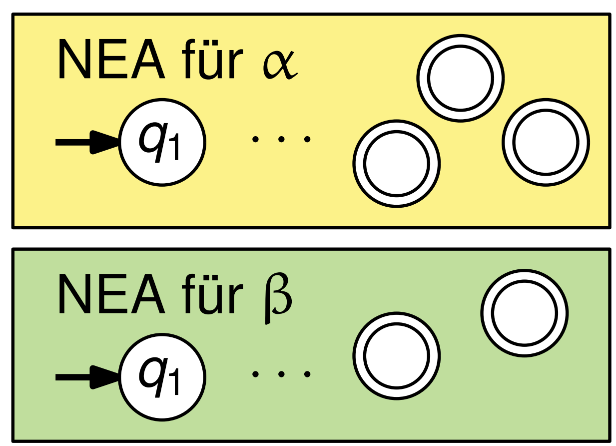


x:



Zusammengesetzte Ausdrücke:

$(\alpha) \cup (\beta)$:

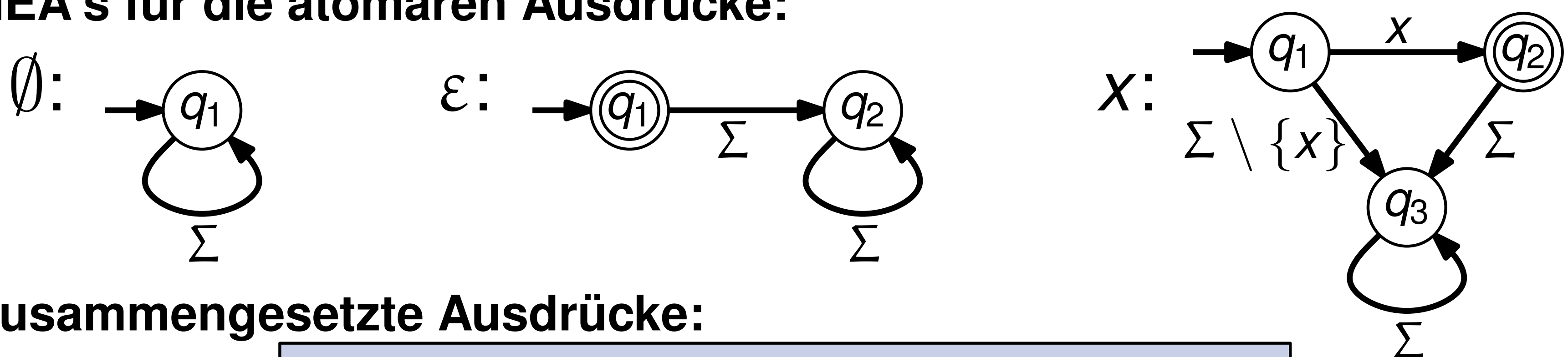


Regulärer Ausdruck → NEA

Rekursive Definition eines regulären Ausdrucks:

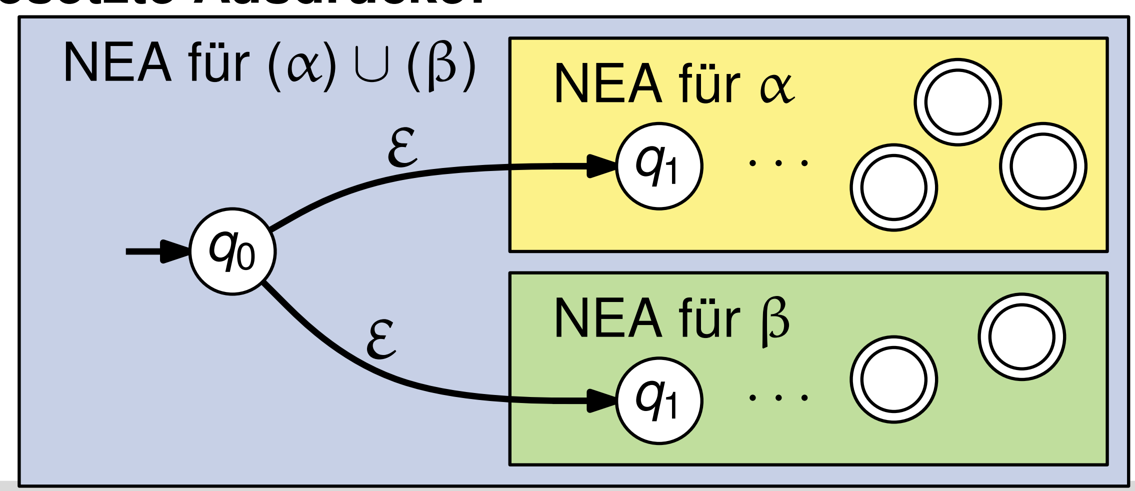
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:



Zusammengesetzte Ausdrücke:

$(\alpha) \cup (\beta)$:



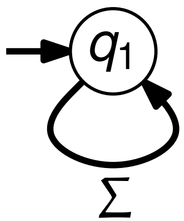
Regulärer Ausdruck → NEA

Rekursive Definition eines regulären Ausdrucks:

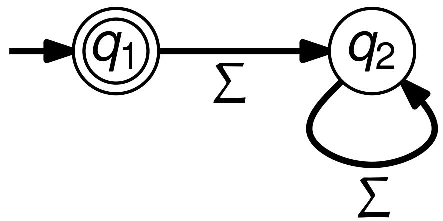
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

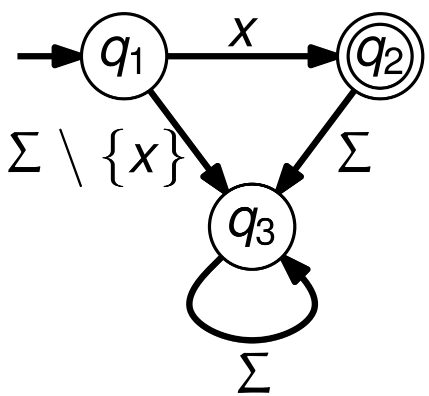
$\emptyset$:



ε:

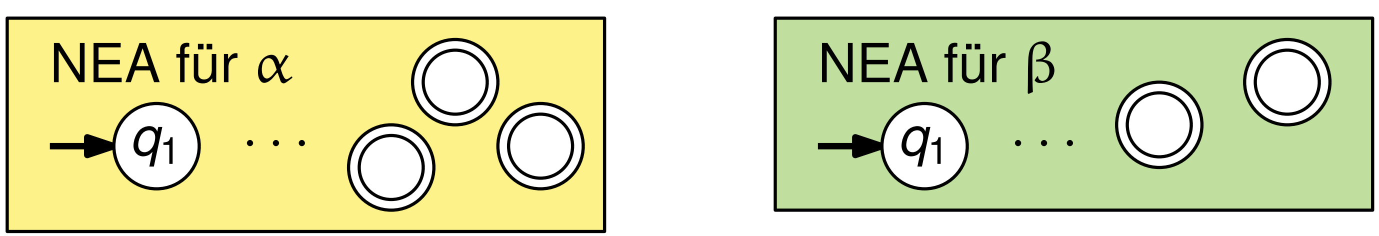


x:



Zusammengesetzte Ausdrücke:

$(\alpha) \cdot (\beta)$:



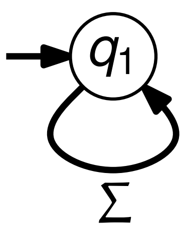
Regulärer Ausdruck → NEA

Rekursive Definition eines regulären Ausdrucks:

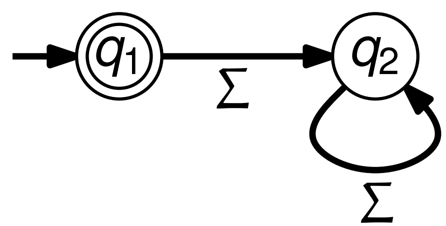
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

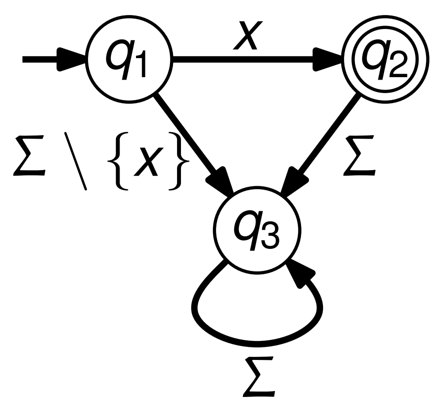
$\emptyset$:



ε:

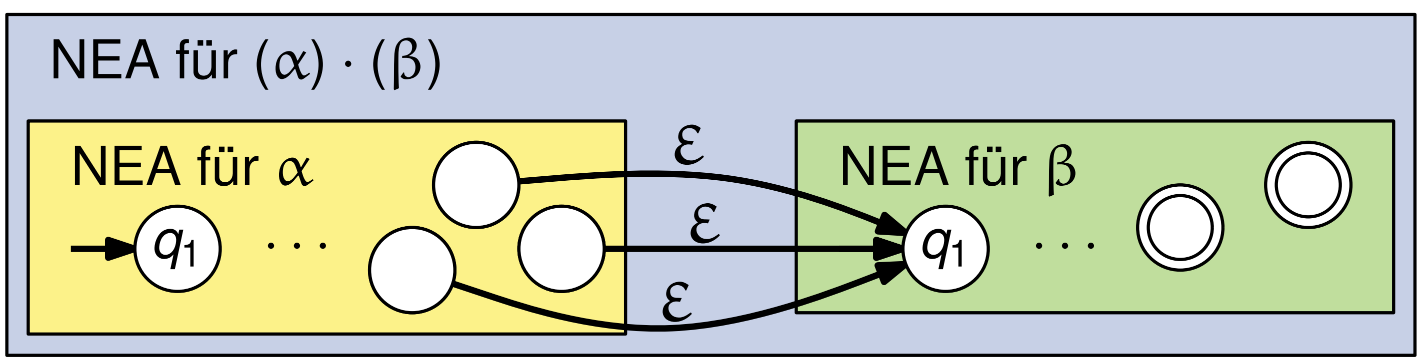


x:



Zusammengesetzte Ausdrücke:

$(\alpha) \cdot (\beta)$:



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

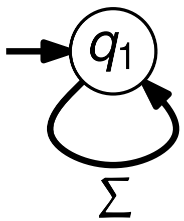
Regulärer Ausdruck $\rightarrow$ NEA

Rekursive Definition eines regulären Ausdrucks:

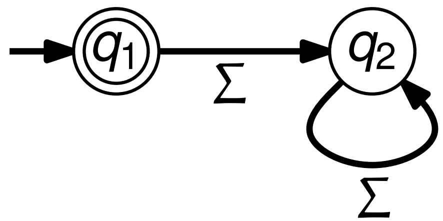
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

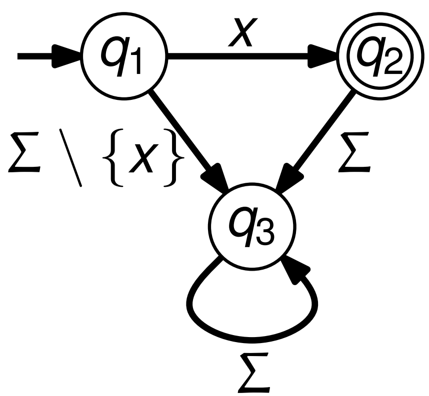
$\emptyset$:



ε:

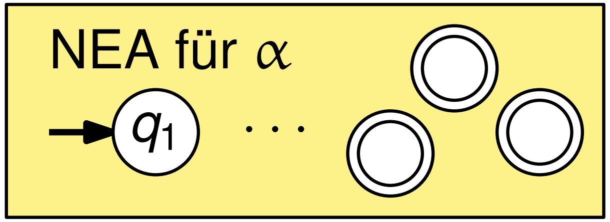


x:



Zusammengesetzte Ausdrücke:

$(\alpha)^\star$:



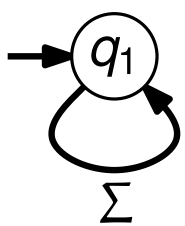
Regulärer Ausdruck → NEA

Rekursive Definition eines regulären Ausdrucks:

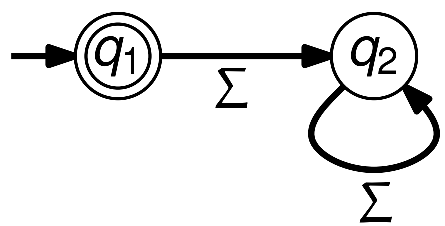
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

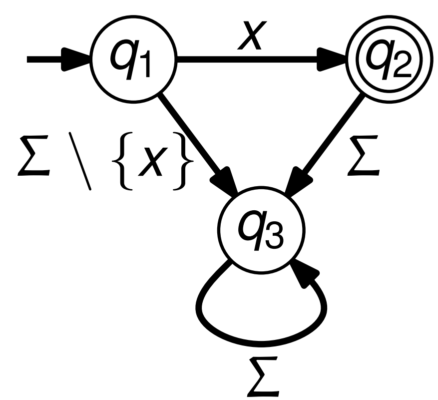
$\emptyset$:



ε:

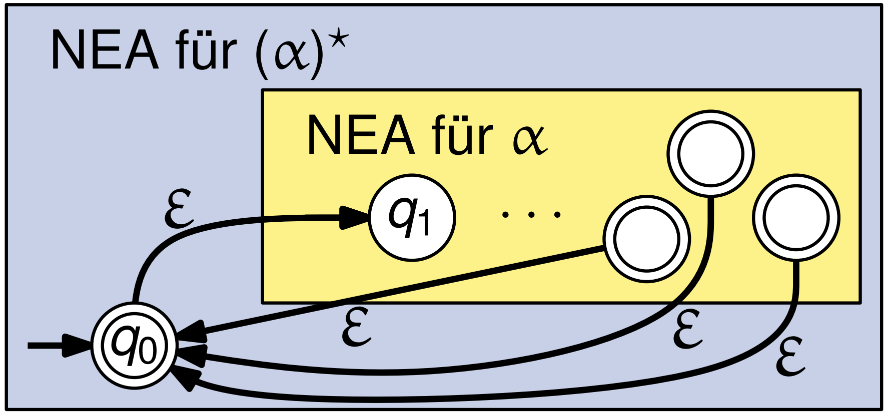


x:



Zusammengesetzte Ausdrücke:

$(\alpha)^\star$:



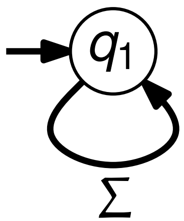
Regulärer Ausdruck → NEA

Rekursive Definition eines regulären Ausdrucks:

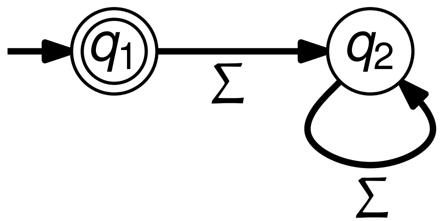
- Die leere Menge $\emptyset$, das leere Wort ε und einzelne Zeichen $x \in \Sigma$ sind reguläre Ausdrücke.
- Für zwei reguläre Ausdrücke α, β sind auch $(\alpha) \cup (\beta)$, $(\alpha) \cdot (\beta)$, $(\alpha)^+$ und $(\alpha)^\star$ reguläre Ausdrücke.

NEA's für die atomaren Ausdrücke:

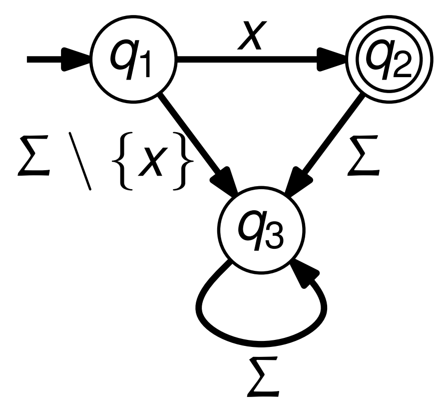
$\emptyset$:



ε:

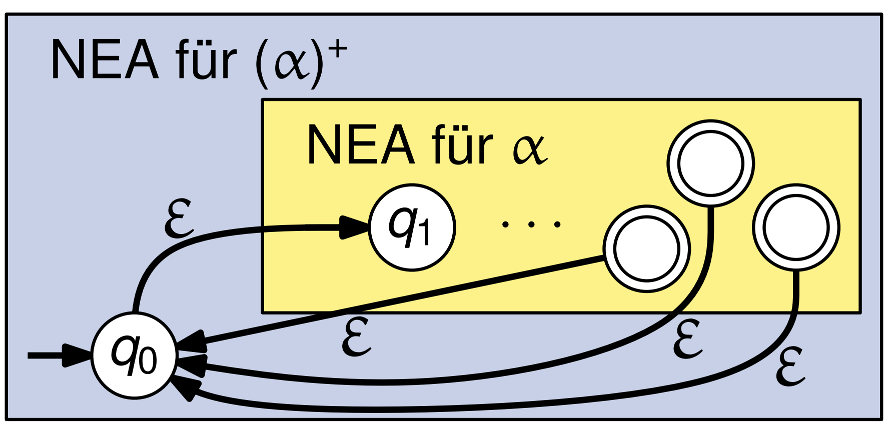


x:



Zusammengesetzte Ausdrücke:

$(\alpha)^+$:



Simulation des NEA

Größe des NEA's: Der so konstruierte NEA hat $O(k)$ viele Zustände, wobei $k = |R|$ die Länge des regulären Ausdrucks ist.

Simulation eines Wortes T:

- Erinnerung: T wird akzeptiert, wenn es eine Ausführung gibt, die in einem Endzustand landet.
- Alle möglichen Ausführungen auflisten $\rightarrow$ zu aufwendig.
- Markiere stattdessen zu jedem Zeitpunkt alle Zustände, in denen man sich aktuell befinden könnte.
- Berechne daraus, abhängig vom nächsten Zeichen, die neue Markierung.
- Akzeptiere, wenn am Ende ein Endzustand markiert ist.

Simulation des NEA

Größe des NEA's: Der so konstruierte NEA hat $O(k)$ viele Zustände, wobei $k = |R|$ die Länge des regulären Ausdrucks ist.

Simulation eines Wortes T:

- Erinnerung: T wird akzeptiert, wenn es eine Ausführung gibt, die in einem Endzustand landet.
- Alle möglichen Ausführungen auflisten $\rightarrow$ zu aufwendig.
- Markiere stattdessen zu jedem Zeitpunkt alle Zustände, in denen man sich aktuell befinden könnte.
- Berechne daraus, abhängig vom nächsten Zeichen, die neue Markierung.
- Akzeptiere, wenn am Ende ein Endzustand markiert ist.

Laufzeit:
$O(k)$ Aufwand für jedes Gelesene Zeichen $\Rightarrow$ insgesamt $O(k \cdot n)$.

Geht das Schneller?
Idee: In einem DEA braucht die Ausführung nur $O(1)$ Zeit pro Zeichen.

NEA $\rightarrow$ DEA

Potenzmengenkonstruktion (Idee): (bekannt aus TGI)

- Ein Knoten in dem DEA entspricht einer Teilmenge von Knoten in dem NEA.
- Eine solche Teilmenge kommt als Knoten vor, wenn sie als Markierung in einer Abarbeitung vorkommen kann.
- Die Übergänge wandeln entsprechend Markierungen ineinander um.
- Ein Zustand des DEA ist akzeptierend, falls er einen akzeptierenden Zustand des NEA enthält.

NEA $\rightarrow$ DEA

Potenzmengenkonstruktion (Idee): (bekannt aus TGI)

- Ein Knoten in dem DEA entspricht einer Teilmenge von Knoten in dem NEA.
- Eine solche Teilmenge kommt als Knoten vor, wenn sie als Markierung in einer Abarbeitung vorkommen kann.
- Die Übergänge wandeln entsprechend Markierungen ineinander um.
- Ein Zustand des DEA ist akzeptierend, falls er einen akzeptierenden Zustand des NEA enthält.

Laufzeit:

- Der resultierende DEA kann $O(2^k)$ (also exponentiell) groß sein.
- Die Simulation von n Zeichen dauert dennoch nur $O(n)$ Zeit.
- Laufzeit: $O(2^k + n)$. $\Rightarrow$ FPT bezüglich Parameter k.

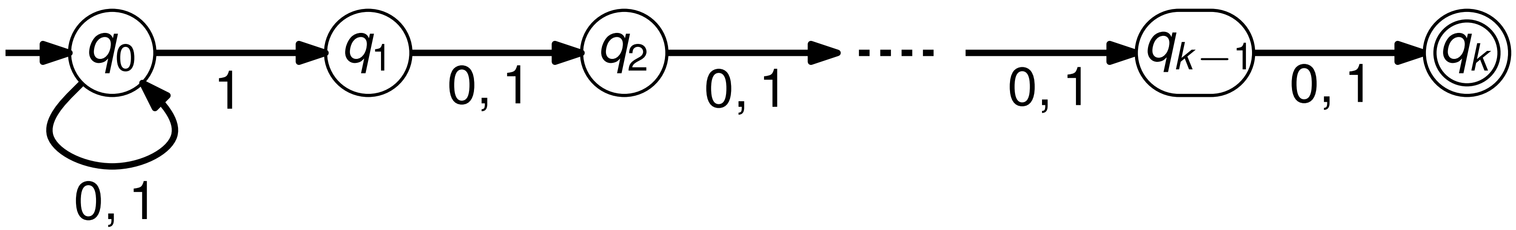
Beachte:

Auch wenn es einen polynomiellen Algorithmus mit Laufzeit $O(k \cdot n)$ gibt, kann es schneller sein den FPT-Algorithmus zu verwenden (für $k << n$).

Ein kleinerer DEA?

Kann man einen kleineren DEA finden?

Behauptung:
Der folgende NEA beschreibt eine Sprache, für die jeder DEA 2^k Zustände braucht.



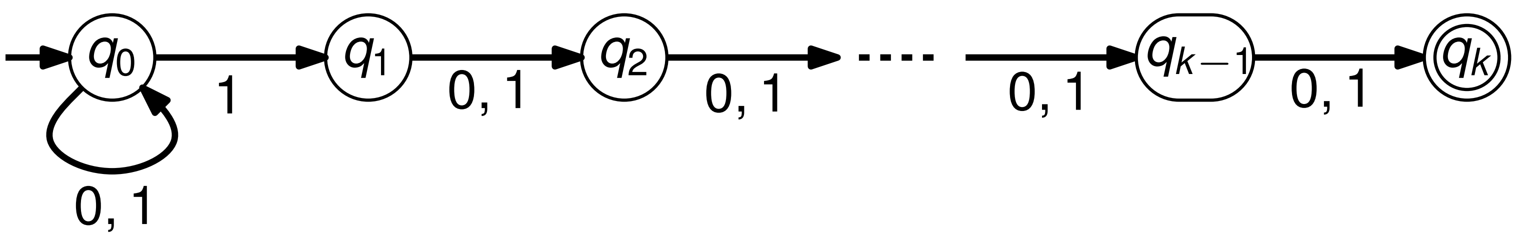
(akzeptiert werden alle Wörter, die an der k-letzten Stelle eine 1 hatten)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ein kleinerer DEA?

Kann man einen kleineren DEA finden?

Behauptung:
Der folgende NEA beschreibt eine Sprache, für die jeder DEA 2^k Zustände braucht.



(akzeptiert werden alle Wörter, die an der k-letzten Stelle eine 1 hatten)

Beweis Idee:

- Jedes der k zuletzt gelesenen Zeichen kann für die Akzeptanz entscheidend sein. (der DEA weiß nicht wie lang das Wort ist)
- Unterscheiden sich die letzten k Zeichen zweier Eingaben, so muss der DEA in unterschiedlichen, nichtäquivalenten Zuständen sein.
- Es gibt 2^k verschiedene Zeichenketten der Länge $k \Rightarrow 2^k$ Zustände nötig.