

Einführung in die Computerlinguistik

Finite State Automata (Endliche Automaten)

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2019

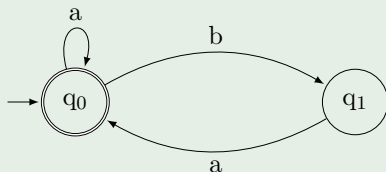


Automaten

- haben in der Regel Zustände, darunter einen Startzustand und Endzustände;
- verarbeiten eine Eingabe;
- durchlaufen bei Verarbeitung der Eingabe verschiedene Zustände;
- haben eventuell eine Ausgabe;
- akzeptieren Eingaben, die zu einem Endzustand führen.

Beispiel FSA

Automat, der Wörter über $\{a, b\}$ akzeptiert, bei denen jedes b notwendig von einem a gefolgt ist.



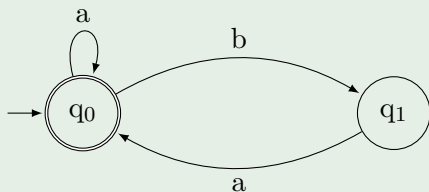
- Zustände q_0, q_1 . q_0 ist Start- und Endzustand.
- Übergänge in q_0 : falls ein a gelesen wird, bleibt man in q_0 ; falls ein b gelesen wird, wechselt man in q_1 ;
- Übergänge in q_1 : man kann als nächstes nur ein a verarbeiten und geht dabei wieder in q_0 .

DFA (1)

A **deterministic finite state automaton** (DFA) M is a quintuple $\langle Q, \Sigma, \delta, q_0, F \rangle$ such that

- Q is a finite set of states.
- Σ is a finite set, the input alphabet.
- $\delta : Q \times \Sigma \rightarrow Q$ is a partial function, the transition function.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of final states.

Beispiel: DFA als Graph und als Tupel



DFA $\langle Q, \Sigma, \delta, q_0, F \rangle$ mit

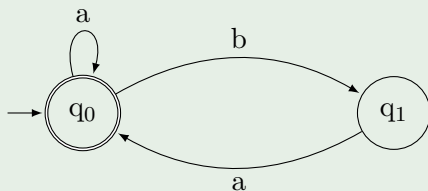
- $Q = \{q_0, q_1\}$ (q_0 Startzustand),
- $F = \{q_0\}$,
- $\Sigma = \{a, b\}$ und
- $\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_1$,
 $\delta(q_1, a) = q_0$, $\delta(q_1, b)$ nicht definiert.

Die Hülle der Übergangsfunktion, $\hat{\delta}$, beschreibt für einen Zustand q und eine Eingabe $w \in \Sigma^*$, in welchen Zustand man, ausgehend von q , nach Verarbeiten von w gelangt.

The transition closure of δ is its reflexive transitive closure:

Let $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a DFA. We define its **transition closure** $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ as follows:

- $\hat{\delta}(q, \varepsilon) := q$ for all $q \in Q$.
- $\hat{\delta}(q, wa) := \delta(\hat{\delta}(q, w), a)$ for all $a \in \Sigma, w \in \Sigma^*$.

Beispiel: $\hat{\delta}$ 

- $\hat{\delta}(q_0, aaba) = q_0$
- $\hat{\delta}(q_0, \varepsilon) = q_0$
- $\hat{\delta}(q_1, aab) = q_1$
- $\hat{\delta}(q_1, b)$ ist nicht definiert.

Die akzeptierte Sprache sind alle Wörter, die erlauben, von q_0 in einen Endzustand zu gelangen.

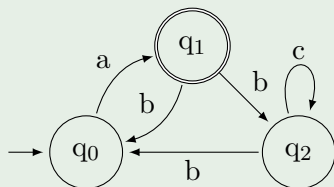
The **language accepted by a DFA** $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ is

$$L(M) := \{w \mid \hat{\delta}(q_0, w) \in F\}$$

NFA (1)

Ein endlicher Automat ist **nicht-deterministisch**, wenn es die Übergangsfunktion nicht eindeutig ist. D.h., es gibt Zustände, von denen aus bei Verarbeitung eines Eingabesymbols mehrere Zustände erreicht werden könnten.

Nicht-deterministischer Automat



Sprache: $a(ba|bc^*ba)^*$

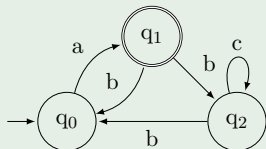
NFA (2)

A finite state automaton is non-deterministic if the transition function is not deterministic:

A **non-deterministic finite state automaton** (NFA) M is a quintuple $\langle Q, \Sigma, \delta, q_0, F \rangle$ such that

- Q, Σ, q_0 and F are like in a DFA.
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function (where $\mathcal{P}(Q)$ is the powerset of Q).

Nicht-deterministischer Automat



Übergangsfunktion:

$$\delta(q_0, a) = \{q_1\} \quad \delta(q_2, c) = \{q_2\}$$

$$\delta(q_1, b) = \{q_0, q_2\} \quad \delta(q_2, b) = \{q_0\}$$

In allen anderen Fällen ist der Wert \emptyset .

NFA (3)

$\hat{\delta}$, beschreibt für einen Zustand q und eine Eingabe $w \in \Sigma^*$, in welchen Zustände man, ausgehend von q , nach Verarbeiten von w gelangen kann.

Let $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a NFA. We define its **transition closure** $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$ as follows:

- $\hat{\delta}(q, \varepsilon) := \{q\}$ for all $q \in Q$.
- For all $a \in \Sigma, w \in \Sigma^*$:
 $\hat{\delta}(q, wa) := \{p \mid \text{there is a } r \in \hat{\delta}(q, w) \text{ such that } p \in \delta(r, a)\}$.

Beispiel $\hat{\delta}$ bei NFA

(Automat s. vorherige Folie):

- $\hat{\delta}(q_0, ab) = \{q_0, q_2\}$,
- $\hat{\delta}(q_0, abca) = \emptyset$

Die Sprache des NFA ist die Menge aller Wörter, die erlauben, ausgehend von q_0 einen Endzustand zu erreichen.

The **language accepted by a NFA** $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ is

$$L(M) := \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Equivalence of DFA and NFA (1)

The following holds:

Let L be a language. There is a DFA M such that $L = L(M)$ iff there is a NFA M' such that $L = L(M')$.

The “ \Rightarrow ” direction of the iff is obvious: Each DFA can be written as a NFA by replacing $\delta(q_1, a) = q_2$ with $\delta(q_1, a) = \{q_2\}$.

Equivalence of DFA and NFA (2)

For the “ \Leftarrow ” direction, we need to show how to construct an equivalent DFA for a given NFA.

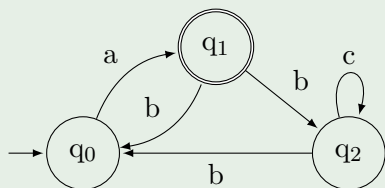
For this, we need to do two things:

- 1 Give the construction algorithm.
- 2 Prove that the resulting DFA accepts the string language of the original NFA.

Proofs of this type are called **construction proofs**.

Equivalence of DFA and NFA (3)

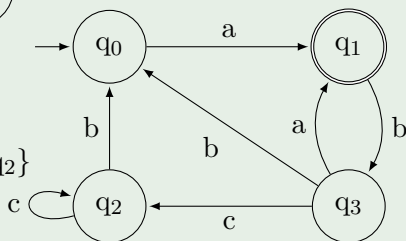
Idea of the construction: In the new DFA, the states are subsets of the state set Q of the NFA. A state contains all $q \in Q$ that are reachable with the same input word.



neue Zustände:

$q_0 = \{q_0\}$, $q_1 = \{q_1\}$,

$q_2 = \{q_2\}$, $q_3 = \{q_0, q_2\}$



Equivalence of DFA and NFA (4)

Let $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ be an NFA. We construct an equivalent DFA $M' = \langle Q', \Sigma, \delta', q'_0, F' \rangle$ as follows:

- $Q' := \mathcal{P}(Q)$.
- $q'_0 := \{q_0\}$.
- $\delta'(Q_1, a) := \bigcup_{q \in Q_1} \delta(q, a)$.
- $F' := \{Q_f \subseteq Q \mid Q_f \cap F \neq \emptyset\}$.

After having read the input w , the automaton is in a state $Q_w \subseteq Q$ such that, in the original NFA, $\hat{\delta}(q_0, w) = Q_w$.

Minimizing DFAs (1)

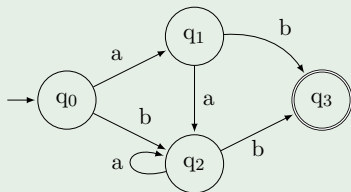
Ein DFA kann eindeutig durch folgende Äquivalenzrelation minimiert werden:

$p \sim q$ genau dann wenn für alle w gilt: $\hat{\delta}(p, w) \in F$
gdw. $\hat{\delta}(q, w) \in F$.

Die Äquivalenzklassen $[q]$ bilden dann die Zustände, und für die neue Übergangsfunktion δ' gilt $\delta'([q], a) = [\delta(q, a)]$.

Mit anderen Worten: Zustände, die die gleiche Sprache definieren, werden als äquivalent zusammengefasst.

Minimierung: einfaches Beispiel



$$q_1 \sim q_2$$

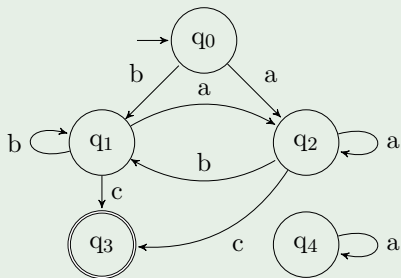
Minimizing DFAs (2)

Algorithmus für die Berechnung der Äquivalenzklassen:

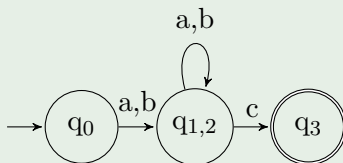
- 1 Entferne alle nutzlosen Zustände (nicht erreichbar oder zu keinem Endzustand führend).
- 2 Partitioniere Q in $\mathcal{Q} = \{F, Q \setminus F\}$.
- 3 Für jedes Eingabesymbol a zerlege die Mengen in \mathcal{Q} sukzessive weiter, so dass es anschließend für alle $Q' \in \mathcal{Q}$ ein $Q'' \in \mathcal{Q}$ gibt mit $\hat{\delta}(q', a) \in Q''$ für alle $q' \in Q'$.
Anders gesagt: aus einem $Q' \in \mathcal{Q}$ bleiben nur die Zustände in einer Menge, die nach Verarbeitung von a in ein und dieselbe Menge $Q'' \in \mathcal{Q}$ führen.
- 4 Wiederhole 3. so lange, bis sich \mathcal{Q} nicht mehr verändert.

Der neue Startzustand ist $[q_0]$, Endzustände sind alle $[q_f]$ mit $q_f \in F$.

Minimizing DFAs (3)



entferne q_4 (nutzlos)
Start: $\{q_3\}$ $\{q_0, q_1, q_2\}$
a, b: trennen nicht
c: $\{q_3\}$ $\{q_0\}$ $\{q_1, q_2\}$
keine weitere Trennung



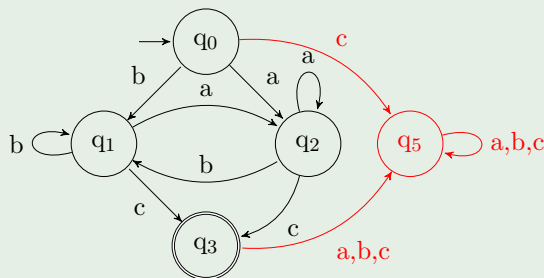
Ergebnis:

Minimizing DFAs (4)

Einfacher Algorithmus, der berechnet, welche Zustände in einer Äquivalenzklasse sind:

1. Entferne nutzlose Zustände.
2. Ergänze den DFA zu einem vollständigen DFA durch Hinzufügen eines Fehlerzustands (trap state).

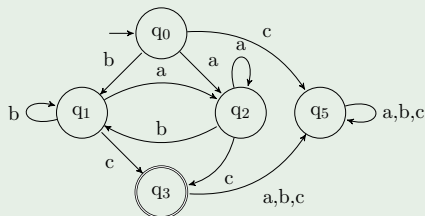
Example: trap state



Minimizing DFAs (5)

3. Bilde eine $(|Q| - 1) \times (|Q| - 1)$ Matrix, deren Spaltenindizes alle Zustände außer dem letzten und deren Zeilenindizes alle Zustände außer dem ersten, in umgekehrter Reihenfolge, durchlaufen.
4. Ein X in einem Feld der Matrix bedeutet, dass die beiden Zustände, die die Indizes dieses Feldes bilden, nicht in einer Äquivalenzklasse sein können. Beginne damit, alle Felder mit genau einem Endzustand mit X zu füllen. (Sie können wegen ε nicht in einer Klasse sein.)

Minimizing DFAs (6)

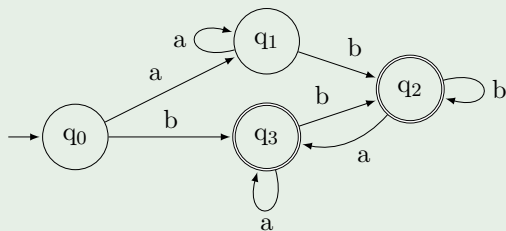


	0	1	2	3
5	X	X	X	X
3	X	X	X	
2	X			
1	X			

- Durchlaufe die noch nicht markierten Felder wiederholt, bis sich die Matrix nicht mehr ändert und führe für jedes Feld mit Indizes $\langle p, q \rangle$ folgendes durch:
Gibt es ein Eingabesymbol a , so dass das Paar $\langle \delta(p, a), \delta(q, a) \rangle$ (oder umgekehrt) markiert ist, so markiere auch das Feld $\langle p, q \rangle$.
- Freie Felder sagen uns anschließend, welche Zustände äquivalent sind.

Minimizing DFAs (7)

Weiteres Bsp.: Minimierung



FSA vollständig,
kein Trap-State nötig

Matrix für
Minimierung:

	0	1	2
3	x	x	
2	x	x	
1			

Neuer Automat:

