

---

# Grundlagen der theoretischen Informatik

---

Kurt Sieber

Fachbereich Mathematik/Theoretische Informatik  
Universität Siegen

Vorlesung vom 19.10.2004 (Stand: 21.10.2004)

# Endliche Automaten

---

## Nächstes Ziel:

“Inhaltliches” Verständnis von endlichen Automaten

## Fragestellungen

- Wie bestimmt man die Sprache  $L(A)$  zu einem vorgegebenen Automaten  $A$ ?
- Wie beweist man, dass ein Automat tatsächlich die gewünschte Sprache akzeptiert?
- Wie findet man einen Automaten, der eine vorgegebene Sprache akzeptiert?
- Wie findet man heraus, ob es überhaupt einen solchen Automaten für eine vorgegebene Sprache gibt?

# Endliche Automaten

---

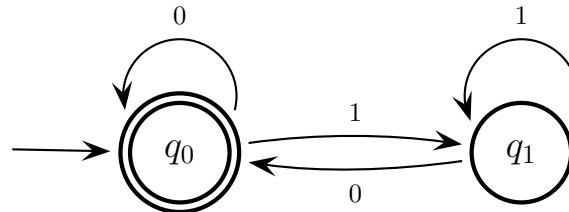
## Dazu: Grundlegende Überlegungen

- Jeder endliche Automat hat nur ein endliches “Gedächtnis”, nämlich seine endlich vielen Zustände.
- Die einzige Information über das bereits gelesene Anfangsstück eines Wortes steckt im aktuellen Zustand.
- Diese begrenzte Information muss ausreichen, um den Rest des Wortes ‘richtig’ abzuarbeiten.

# Endliche Automaten

---

**Beispiel** (wie oben):



Welche Information über das bereits gelesene Wort  $w$  steckt in den Zuständen  $q_0$  bzw.  $q_1$ ?

- $q_0$  bedeutet: Entweder  $w = \varepsilon$  oder  $w$  endet auf 0.
- $q_1$  bedeutet:  $w$  endet auf 1.

Setzt man diese Bedeutung der Zustände voraus, so ist klar, wie der Automat auf das nächste Zeichen reagieren muss:

- Mit 0 muss er stets in  $q_0$  übergehen, da  $w0$  auf 0 endet.
- Mit 1 muss er stets in  $q_1$  übergehen, da  $w1$  auf 1 endet.

## Entwurf eines Automaten für eine vorgegebene Sprache

- Man macht sich klar, welche Information über das bereits gelesene Anfangswort nötig ist, um mit dem Restwort 'richtig' weiterzuarbeiten?
- Reicht eine endliche Anzahl  $n$  von unterschiedlichen Informationen aus, so entwirft man einen Automaten mit  $n$  Zuständen.
- Die Zustandsübergänge legt man so fest, dass nach dem Lesen eines Zeichens stets wieder die richtige Information vorliegt.
- Reicht eine endliche Anzahl unterschiedlicher Informationen **nicht** aus, so gibt es keinen endlichen Automaten für die Sprache.

## Endliche Automaten

---

**Beispiel:** Automat  $A_{int}$  für die Sprache

$$L_{int} = \{\varepsilon, -\}\{0, \dots, 9\}^+ = \{\varepsilon, -\}\{0, \dots, 9\}\{0, \dots, 9\}^*$$

Sei  $\Sigma = \{-, 0, \dots, 9\}$ . Man muss folgende Fälle für das bereits gelesene Wort  $w$  unterscheiden:

- $w = \varepsilon$ : Noch nichts gelesen, die gesamte Zahl wird noch erwartet.
- $w = -$ : Vorzeichen gelesen, es wird noch eine **nichtleere** Ziffernfolge erwartet.
- $w \in L_{int} = \{\varepsilon, -\}\{0, \dots, 9\}^+$ : Es war schon mindestens eine Ziffer da, jetzt darf noch eine **beliebige** Ziffernfolge kommen.
- $w$  enthält ‘-’ an der falschen Stelle: Schon alles verdorben!

Also 4 Zustände für diese 4 Situationen:  $q_\varepsilon, q_-, q_{int}, q'$

---

# Endliche Automaten

---

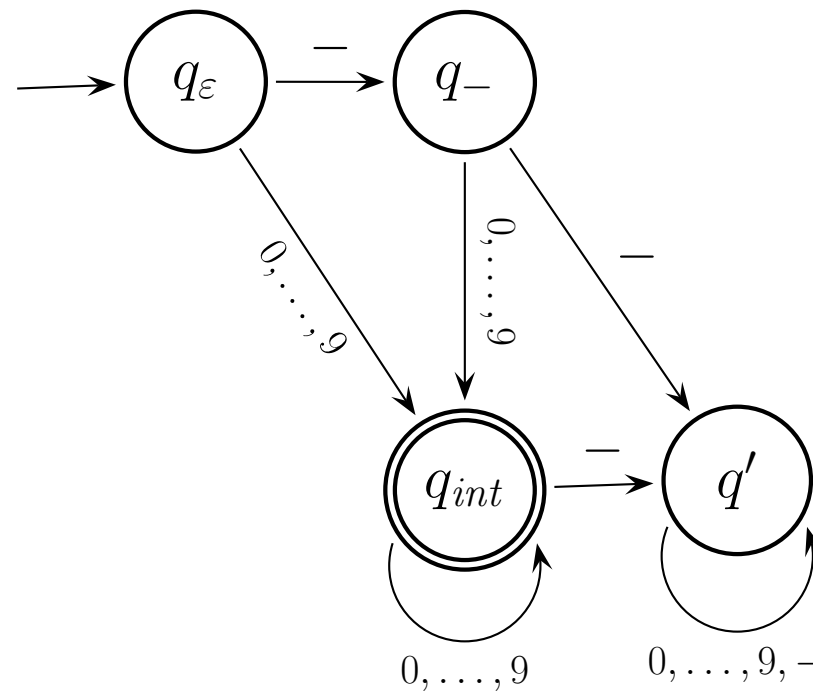
Deshalb definiert man  $A_{int} = (\Sigma, Q, s, F, \delta)$ , wobei

- $Q = \{q_\varepsilon, q_-, q_{int}, q'\}$
- $s = q_\varepsilon$
- $F = \{q_{int}\}$
- $\delta : Q \times \Sigma \rightarrow Q$ 
  - $(q_\varepsilon, -) \mapsto q_-$
  - $(q_\varepsilon, a) \mapsto q_{int}$  für alle  $a \in \{0, \dots, 9\}$
  - $(q_-, -) \mapsto q'$
  - $(q_-, a) \mapsto q_{int}$  für alle  $a \in \{0, \dots, 9\}$
  - $(q_{int}, -) \mapsto q'$
  - $(q_{int}, a) \mapsto q_{int}$  für alle  $a \in \{0, \dots, 9\}$
  - $(q', a) \mapsto q'$  für alle  $a \in \Sigma$  (Sackgasse!)

# Endliche Automaten

---

## Graphische Darstellung



Ein Pfeil mit mehreren Zeichen  $a_1, \dots, a_n$  ist Abkürzung für  $n$  Pfeile mit den einzelnen Zeichen.

## Endliche Automaten

---

Wie beweist man, dass  $L(A_{int}) = L_{int}$ ? Man zeigt, dass jedes  $w \in \Sigma^*$  zum gewünschten Zustand führt, d.h.:

1. Wenn  $w = \varepsilon$ , dann  $(q_\varepsilon, w) \vdash^* (q_\varepsilon, \varepsilon)$ .
2. Wenn  $w = -$ , dann  $(q_\varepsilon, w) \vdash^* (q_-, \varepsilon)$ .
3. Wenn  $w \in L_{int}$ , dann  $(q_\varepsilon, w) \vdash^* (q_{int}, \varepsilon)$ .
4. Wenn  $w \notin L_{int} \cup \{\varepsilon, -\}$ , dann  $(q_\varepsilon, w) \vdash^* (q', \varepsilon)$ .

Dies geschieht durch Induktion über  $|w|$ :

$|w| = 0$ , d.h.  $w = \varepsilon$ : Es gilt  $(q_\varepsilon, w) \vdash^0 (q_\varepsilon, \varepsilon)$ .

$|w| > 0$ , d.h.  $w = va$ : Fallunterscheidung für  $v$ . Wenn  $v \in L_{int}$ , dann gilt nach Induktionsannahme  $(q_\varepsilon, v) \vdash^* (q_{int}, \varepsilon)$ . Ist  $a \in \{0, \dots, 9\}$ , dann folgt  $(q_\varepsilon, w) = (q_\varepsilon, va) \vdash^* (q_{int}, a) \vdash (q_{int}, \varepsilon)$  und das ist der gewünschte Zustand, weil  $w = va \in L_{int}$ . Ist  $a = -$ , dann folgt  $(q_\varepsilon, w) = (q_\varepsilon, va) \vdash^* (q_{int}, a) \vdash (q', \varepsilon)$  und das ist wieder der richtige Zustand, weil  $w \notin L_{int} \cup \{\varepsilon, -\}$ . Ähnlich für die übrigen Fälle!  $\square$

---

# Endliche Automaten

---

**Beispiel:** Ein Automat  $A$  für die Sprache  $L = \{a^n b^n \mid n \geq 0\}$ ?

**Intuition:**

- Um  $w \in L$  zu testen, müsste ein endlicher Automat (insbesondere) die Anzahl der  $a$ 's und  $b$ 's in  $w$  vergleichen.
- Also muss er stets die Anzahl der bisher gelesenen  $a$ 's kennen.
- Da diese Anzahl beliebig groß werden kann, reichen die endlich vielen Zustände zum Speichern dieser Information nicht aus.
- Also gibt es keinen Automaten  $A$  mit  $L = L(A)$ .

**Schlagwort:** “Ein endlicher Automat kann nicht zählen.”  
(Soll heißen: Er kann nicht beliebig hoch zählen).

# Endliche Automaten

---

## Exakter Beweis:

Angenommen, es existiert ein DEA  $A = (\Sigma, Q, s, F, \delta)$  mit  $L(A) = L$ .

Wir betrachten alle Wörter der Form  $a^n$  mit  $n \geq 0$ .

Für jedes  $n \geq 0$  existiert ein Zustand  $q \in Q$  mit  $(s, a^n) \vdash_A^* (q, \varepsilon)$  (der Zustand, der durch Lesen von  $a^n$  erreicht wird). Da es unendlich viele Wörter  $a^n$  gibt, aber nur endlich viele  $q \in Q$ , muss es (mindestens) zwei Wörter  $a^i, a^j$  ( $i \neq j$ ) geben, mit denen man den **gleichen** Zustand  $q$  erreicht, d.h.  $(s, a^i) \vdash_A^* (q, \varepsilon)$  und  $(s, a^j) \vdash_A^* (q, \varepsilon)$ .

Sei  $q' \in Q$  mit  $(q, b^i) \vdash_A^* (q', \varepsilon)$ . Dann gilt  $(s, a^i b^i) \vdash_A^* (q, b^i) \vdash_A^* (q', \varepsilon)$  und  $(s, a^j b^i) \vdash_A^* (q, b^i) \vdash_A^* (q', \varepsilon)$ . Aber  $a^i b^i \in L$  und  $a^j b^i \notin L$ , d.h. sie dürfen nicht beide zum gleichen Zustand  $q'$  führen.

Damit ist die Annahme zum Widerspruch geführt, d.h. es existiert kein DEA  $A$  mit  $L(A) = L$ . □

# Endliche Automaten

---

Im Beweis wurde benutzt:

- Das **Schubfachprinzip** (engl. **pigeonhole principle**):  
Wenn man eine Menge von Dingen auf eine gewisse Anzahl von Schubfächern verteilt, und die Anzahl der Dinge ist größer als die Anzahl der Schubfächer, dann landen mindestens zwei Dinge im gleichen Schubfach.  
Hier: Unendlich viele Wörter  $a^n$  wurden auf endlich viele Zustände verteilt.
- Folgende Eigenschaft von  $\vdash_A^*$ :  
Wenn  $(q, u) \vdash_A^* (q', \varepsilon)$ , dann gilt auch  $(q, uv) \vdash_A^* (q', v)$ .

## Endliche Automaten

---

**Lemma 1.6** Sei  $A = (\Sigma, Q, s, F, \delta)$  ein DEA, seien  $u, v \in \Sigma^*$  und  $q, q' \in Q$ . Dann gilt:

1.  $(q, u) \vdash_A^* (q', \varepsilon) \Leftrightarrow (q, uv) \vdash_A^* (q', v)$
2.  $(q, uv) \vdash_A^* (q', \varepsilon) \Leftrightarrow$  es existiert ein  $q'' \in Q$   
mit  $(q, uv) \vdash_A^* (q'', v) \vdash_A^* (q', \varepsilon)$

### Begründung:

1. Die Übergangsschritte, die  $A$  bei Abarbeitung eines Wortes  $u$  macht, werden von einem *hinter*  $u$  stehenden Wort  $v$  nicht beeinflusst. Denn  $A$  arbeitet ja von links nach rechts, und kann deshalb das Wort  $v$  während der Abarbeitung von  $u$  gar nicht sehen.
2. Jede Folge von Übergangsschritten für ein Wort  $uv$  lässt sich aufteilen in die Schritte für  $u$  und die Schritte für  $v$ . Das liegt daran, dass der Automat *zeichenweise* von links nach rechts liest.

# Endliche Automaten

---

## Frage:

Für welche Sprachen  $L$  existiert ein DEA  $A$  mit  $L(A) = L$ ?

## Wir werden beweisen:

Ein DEA  $A$  mit  $L = L(A)$  existiert genau dann, wenn  $L$  eine reguläre Sprache ist. (vgl. Übungsblatt 3, Aufgabe 2)

# Endliche Automaten

---

## Die Funktionsschreibweise $\delta^*$

Sei  $A = (\Sigma, Q, s, F, \delta)$  ein DEA.

Dann existiert für jede Konfiguration  $(q, w) \in Q \times \Sigma^*$  **genau ein** Zustand  $q' \in Q$  mit  $(q, w) \vdash_A^* (q', \varepsilon)$  (weil für jedes einzelne Zeichen von  $w$  stets ein eindeutiger Übergangsschritt existiert).

Diesen eindeutigen Zustand  $q'$  bezeichnen wir mit  $\delta^*(q, w)$ .

Die so definierte Funktion  $\delta^* : Q \times \Sigma^* \rightarrow Q$  lässt sich auch ohne den Umweg über  $\vdash_A^*$  definieren, nämlich durch Induktion über die Länge von  $|w|$ :

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

oder:

- $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$

# Endliche Automaten

---

## Vor- und Nachteile beider Schreibweisen:

In der Funktionsschreibweise haben wir eine eigenständige Bezeichnung für den Ergebniszustand  $q'$  (in der die Existenz und Eindeutigkeit implizit enthalten ist). In der Relationsschreibweise müssen wir immer erst die Existenz (und evtl. Eindeutigkeit) von  $q'$  erwähnen (und damit einen neuen Namen für einen Zustand einführen).

In der Relationsschreibweise kann man Übergangsschritte besser aneinanderhängen und man hat die Schreibweisen  $\vdash^n, \vdash^+, \vdash^*$  mit schönen Gesetzmäßigkeiten (Monoid!). Mit der Funktionsschreibweise geht das nicht so einfach (denn  $\delta : Q \times \Sigma \rightarrow Q$  kann man nicht mit sich selbst verknüpfen).