

Theoretische Grundlagen der Informatik

Vorlesung am 06. November 2018

INSTITUT FÜR THEORETISCHE INFORMATIK



letzte Vorlesung – alternative Sicht

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein beliebiger DEA. Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn **für alle Wörter $w \in \Sigma^*$** gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein beliebiger DEA. Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn **für alle Wörter** $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Alternative Sicht: Definiere zu jedem Zustand $q \in Q$ die Menge

$$M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}.$$

(Also \mathcal{A} **würde** w akzeptieren wenn q der Startzustand **wäre**.)

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein beliebiger DEA. Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn **für alle Wörter** $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Alternative Sicht: Definiere zu jedem Zustand $q \in Q$ die Menge

$$M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}.$$

(Also \mathcal{A} **würde** w akzeptieren wenn q der Startzustand **wäre**.)

\rightsquigarrow Beobachtung: $\delta(q, w) \in F$ gleichbedeutend mit $w \in M(q)$.

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein beliebiger DEA. Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Alternative Sicht: Definiere zu jedem Zustand $q \in Q$ die Menge

$$M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}.$$

(Also \mathcal{A} würde w akzeptieren wenn q der Startzustand wäre.)

\rightsquigarrow Beobachtung: $\delta(q, w) \in F$ gleichbedeutend mit $w \in M(q)$.

Sei $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ ein beliebiger DEA. Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

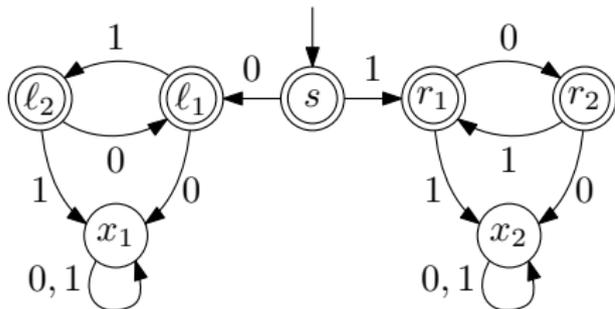
Alternative Sicht – Beispiel

Definiere $M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}$.

Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

Beispielautomat \mathcal{A} :



Der Automat \mathcal{A} erkennt alternierende Wörter, also:

$$L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$$

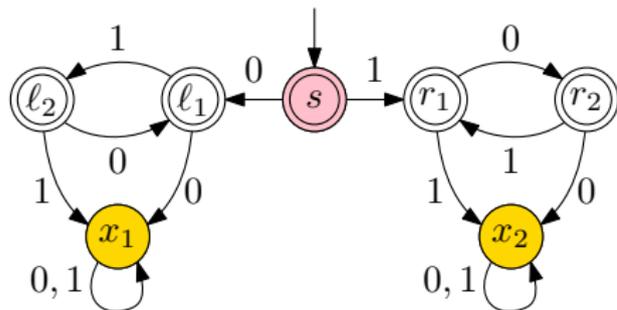
Alternative Sicht – Beispiel

Definiere $M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}$.

Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

Beispielautomat \mathcal{A} :



Der Automat \mathcal{A} erkennt alternierende Wörter, also:

$$L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$$

$$M(s) = L$$

$$M(x_1) = M(x_2) = \emptyset$$

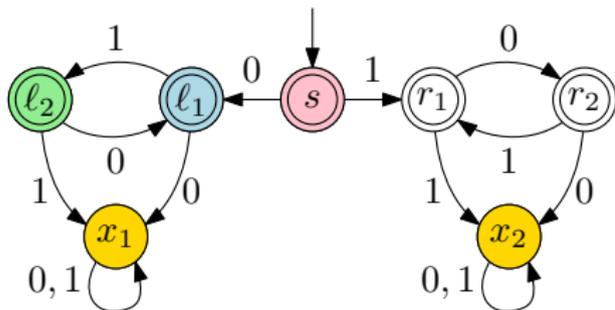
Alternative Sicht – Beispiel

Definiere $M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}$.

Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

Beispielautomat \mathcal{A} :



Der Automat \mathcal{A} erkennt
alternierende Wörter, also:

$$L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$$

$$M(s) = L$$

$$M(l_1) = (10)^* \cdot (1 \cup \varepsilon)$$

$$M(x_1) = M(x_2) = \emptyset$$

$$M(l_2) = (01)^* \cdot (0 \cup \varepsilon)$$

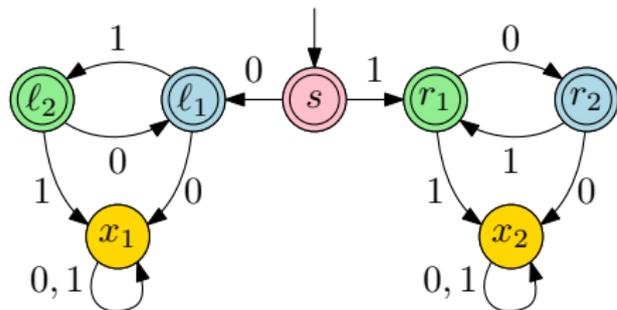
Alternative Sicht – Beispiel

Definiere $M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}$.

Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

Beispielautomat \mathcal{A} :



Der Automat \mathcal{A} erkennt alternierende Wörter, also:

$$L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$$

$$M(s) = L$$

$$M(l_1) = (10)^* \cdot (1 \cup \varepsilon)$$

$$M(r_1) = M(l_2)$$

$$M(x_1) = M(x_2) = \emptyset$$

$$M(l_2) = (01)^* \cdot (0 \cup \varepsilon)$$

$$M(r_2) = M(l_1)$$

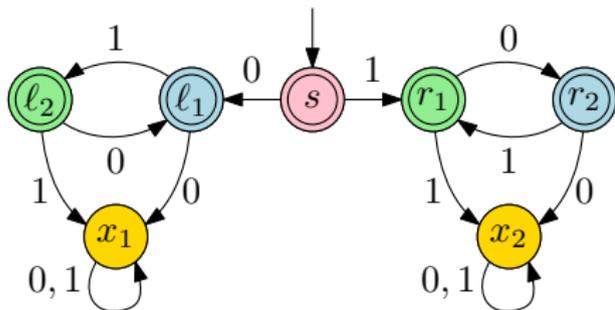
Alternative Sicht – Beispiel

Definiere $M(q) := \{w \in \Sigma^* \mid \text{Automat } (Q, \Sigma, \delta, q, F) \text{ akzeptiert } w\}$.

Zwei Zustände $p, q \in Q$ heißen **äquivalent** ($p \equiv q$), wenn gilt:

$$M(p) = M(q).$$

Beispielautomat \mathcal{A} :



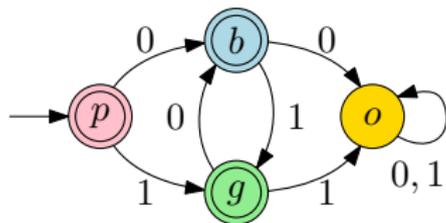
$$M(s) = L$$

$$M(x_1) = M(x_2) = \emptyset$$

$$M(l_1) = (10)^* \cdot (1 \cup \varepsilon)$$

$$M(l_2) = (01)^* \cdot (0 \cup \varepsilon)$$

Äquivalenzklassenautomat \mathcal{A}^{\equiv}



$$M(r_1) = M(l_2)$$

$$M(r_2) = M(l_1)$$

Frage:

Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Frage:

Ist der Äquivalenzklassenautomat zu einem deterministischen endlichen Automaten schon der äquivalente Automat mit der minimalen Anzahl von Zuständen?

Antwort:

Ja, wir zeigen dies wie folgt:

- Zuerst konstruieren wir den minimalen Automaten zur Sprache L (Automat der Nerode-Relation).
- Anschließend zeigen wir, dass \mathcal{A}^{\equiv} höchstens so viele Zustände hat wie der Automat der Nerode-Relation.

- Zwei Zustände p und q eines DEA heißen **äquivalent** ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

$[p]$ bezeichnet die Äquivalenzklasse der zu p äquivalenten Zustände

- Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den **Äquivalenzklassenautomaten** $\mathcal{A}^\equiv = (Q^\equiv, \Sigma^\equiv, \delta^\equiv, s^\equiv, F^\equiv)$ durch:

- $Q^\equiv := \{[q] \mid q \in Q\}$

- $\Sigma^\equiv := \Sigma$

- $\delta^\equiv([q], a) := [\delta(q, a)]$

- $s^\equiv := [s]$

- $F^\equiv := \{[f] \mid f \in F\}$

- **Beachte:** Hier betrachten wir **Äquivalenz von Zuständen eines DEA**.

Definition (Rechtsinvarianz und Index):

Eine Äquivalenzrelation R über Σ^* heißt **rechtsinvariant**, wenn für alle Wörter $x, y \in \Sigma^*$ gilt:

falls $x R y$ so gilt auch $xz R yz$ für alle $z \in \Sigma^*$.

Den **Index** von R bezeichnen wir mit **ind(R)**; er ist die Anzahl der Äquivalenzklassen von Σ^* bezüglich R .

Beispiele:

- Sei $x R_1 y$ wenn $x, y \in \Sigma^*$ dasselbe erste Zeichen haben.
- Sei $x R_2 y$ wenn $x, y \in \Sigma^*$ dasselbe letzte Zeichen haben.
- Sei $x R_3 y$ wenn $x, y \in \Sigma^*$ dasselbe vorletzte Zeichen haben.

R_1, R_2 sind rechtsinvariant, R_3 ist nicht rechtsinvariant.

Definition (Nerode-Relation):

Für eine Sprache $L \subseteq \Sigma^*$ ist die **Nerode-Relation** R_L definiert durch:
für $x, y \in \Sigma^*$ ist $x R_L y$ genau dann wenn $(xz \in L \Leftrightarrow yz \in L)$ für alle $z \in \Sigma^*$ gilt.

Die Nerode-Relation R_L zu einer Sprache $L \subseteq \Sigma^*$ ist eine rechtsinvariante Äquivalenzrelation. Es gilt:

$$\begin{aligned}x R_L y &\Rightarrow (xw \in L \Leftrightarrow yw \in L) \text{ für alle } w \in \Sigma^* \\&\Rightarrow (xzw \in L \Leftrightarrow yzw \in L) \text{ für alle } w, z \in \Sigma^* \\&\Rightarrow (xz R_L yz) \text{ für alle } z \in \Sigma^* .\end{aligned}$$

Satz (von Nerode):

Die folgenden Aussagen sind äquivalent:

- 1 $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.
- 2 L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.
- 3 Die Nerode-Relation hat endlichen Index.

Beweis zu Satz von Nerode: (1) \rightarrow (2)

- (1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.
- (2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation mit endlichem Index.

Beweis: Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ der deterministische endliche Automat, der L akzeptiert, und $R_{\mathcal{A}}$ wie folgt definiert:

$$\forall x, y \in \Sigma^* : x R_{\mathcal{A}} y \iff \delta(s, x) = \delta(s, y).$$

- $R_{\mathcal{A}}$ ist eine rechtsinvariante Äquivalenzrelation.
- Der Index von $R_{\mathcal{A}}$ ist die Anzahl der nicht überflüssigen Zustände von \mathcal{A} , also endlich.
- Also ist L die Vereinigung der Äquivalenzklassen von $R_{\mathcal{A}}$, die zu den Endzuständen von \mathcal{A} gehören.

Beweis zu Satz von Nerode: (2) \rightarrow (3)

- (2) L ist die Vereinigung von (einigen) Äquivalenzklassen einer rechtsinvarianten Äquivalenzrelation R mit endlichem Index.
- (3) Die Nerode-Relation hat endlichen Index.

Beweis:

- Wir zeigen $x R y$ impliziert $x R_L y$ (R_L ist eine Vergrößerung von R)
- Dann gilt $\text{ind}(R_L) \leq \text{ind}(R) < \infty$.

Sei also $x R y$.

- Da R rechtsinvariant ist, gilt für alle $z \in \Sigma^*$: $xz R yz$.
- Voraussetzung: Jede Äquivalenzklasse von R gehört entweder ganz oder gar nicht zu L
- Also: $xz, yz \in L$ oder $xz, yz \notin L$.
- Damit folgt $x R_L y$.

Beweis zu Satz von Nerode: (3) \rightarrow (1)

- (3) Die Nerode-Relation hat endlichen Index.
- (1) $L \subseteq \Sigma^*$ wird von einem deterministischen endlichen Automaten erkannt bzw. akzeptiert.

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L .
Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Beweis zu Satz von Nerode: (3) \rightarrow (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L .
Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

δ ist wohldefiniert:

- Falls $[w]_{R_L} = [w']_{R_L}$ dann gilt $w R_L w'$ und wegen Rechtsinvarianz von R_L auch $wa R_L w'a$.
- Also ist $[wa]_{R_L} = [w'a]_{R_L}$.

Beweis zu Satz von Nerode: (3) \rightarrow (1)

Beweis: Wir konstruieren zu R_L einen deterministischen endlichen Automaten, der L akzeptiert. Sei $\mathcal{A} := (Q, \Sigma, \delta, s, F)$ mit:

- $Q := \{[x]_{R_L} \mid x \in \Sigma^*\}$, Menge aller Äquivalenzklassen bezüglich R_L .
Es ist also $|Q| = \text{ind}(R_L) < \infty$.
- $s := [\varepsilon]_{R_L}$,
- $F := \{[w]_{R_L} \mid w \in L\}$ (wohldefiniert)
- $\delta([x]_{R_L}, a) := [xa]_{R_L}$

Es bleibt zu zeigen, dass \mathcal{A} genau L akzeptiert.

- Nach Konstruktion ist $\delta(s, w) = \delta([\varepsilon], w) = [\varepsilon w]_{R_L} = [w]_{R_L}$.
- Also wird w von \mathcal{A} akzeptiert genau dann, wenn $[w] \in F$ gilt, d.h. wenn $w \in L$.

Korollar

Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat \mathcal{A} zu R_L — der **Automat der Nerode-Relation** — ist minimal.

Korollar

Der im dritten Beweisteil zum Satz von Nerode konstruierte Automat \mathcal{A} zu R_L — der **Automat der Nerode-Relation** — ist minimal.

Beweis: Sei $\mathcal{A}' := (Q', \Sigma, \delta', s', F')$ ein deterministischer endlicher Automat, der L akzeptiert.

- Aus (1) \Rightarrow (2) folgt, dass eine rechtsinvariante Äquivalenzrelation $R_{\mathcal{A}'}$ mit $\text{ind}(R_{\mathcal{A}'}) \leq |Q'|$ existiert.
- Wegen (2) \Rightarrow (3) gilt: $\text{ind}(R_L) \leq \text{ind}(R_{\mathcal{A}'})$.
- Mit (3) \Rightarrow (1) folgt

$$|Q| = \text{ind}(R_L) \leq \text{ind}(R_{\mathcal{A}'}) \leq |Q'|,$$

für den Nerode-Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$.

Satz (Minimalität des Äquivalenzklassenautomats):

Der Äquivalenzklassenautomat \mathcal{A}^{\equiv} zu einem deterministischen endlichen Automaten \mathcal{A} ohne überflüssige Zustände ist minimal.

Satz (Minimalität des Äquivalenzklassenautomats):

Der Äquivalenzklassenautomat \mathcal{A}^{\equiv} zu einem deterministischen endlichen Automaten \mathcal{A} ohne überflüssige Zustände ist minimal.

Beweis: Sei L die vom Automaten \mathcal{A} bzw. \mathcal{A}^{\equiv} akzeptierte Sprache.

- \mathcal{A}^{\equiv} hat keine überflüssigen Zustände.
- Letztes Korollar: Es genügt zu zeigen, dass $|Q^{\equiv}| = \text{ind}(R_L)$.
- Es bleibt zu zeigen, dass für alle $x, y \in \Sigma^*$ gilt:
 $x R_L y \Rightarrow \delta(s, x) \equiv \delta(s, y)$. (hier: Äquivalenz von Zuständen!)

$$\begin{aligned}x R_L y &\Rightarrow \forall z \in \Sigma^*: (xz \in L \Leftrightarrow yz \in L) \\&\Rightarrow \forall z \in \Sigma^*: (\delta(s, xz) \in F \Leftrightarrow \delta(s, yz) \in F) \\&\Rightarrow \forall z \in \Sigma^*: (\delta(\delta(s, x), z) \in F \Leftrightarrow \delta(\delta(s, y), z) \in F) \\&\Rightarrow \delta(s, x) \equiv \delta(s, y)\end{aligned}$$

- Ein DEA ist ein Modell für einen sehr einfachen Computer
- Folgende Mengen sind gleich
 - Die Menge der regulären Sprachen
 - Die Menge aller Sprachen, die von einem DEA erkannt werden.
 - Die Menge aller Sprachen, die von einem NEA erkannt werden.
- Mit Potenzmengenkonstruktion kann ein zu einem NEA äquivalenter DEA konstruiert werden.
- Das Pumping-Lemma für reguläre Sprachen und das Verallgemeinerte Pumping-Lemma für reguläre Sprachen sind Hilfsmittel, mit denen für manche Sprachen gezeigt werden kann, dass sie nicht regulär sind.
- Der Äquivalenzklassenautomat zu einem DEA ohne überflüssige Zustände akzeptiert die gleiche Sprache und ist zustandsminimal.
- Der Automat der Nerode-Relation zu einem DEA akzeptiert die gleiche Sprache und ist zustandsminimal

- **Turing-Maschinen**
- **Berechenbarkeit**



Anil Nerode 1958

*1932

Ph.D. 1956 (Chicago)



John R. Myhill 1958

*1923, †1987

Ph.D. 1949 (Harvard)



Nerode-Automat 1958



Alan Turing 1936

*1912, †1954

Ph.D. 1938 (Princeton)



Turing-Maschine 1936

Beobachtung:

Endliche Automaten sind als Berechnungsmodell nicht mächtig genug.

Frage:

Gibt es ein mächtigeres, realistisches Rechnermodell, das als Grundlage für allgemeine theoretische Aussagen über Berechenbarkeit, Entscheidbarkeit und Komplexität geeignet ist?

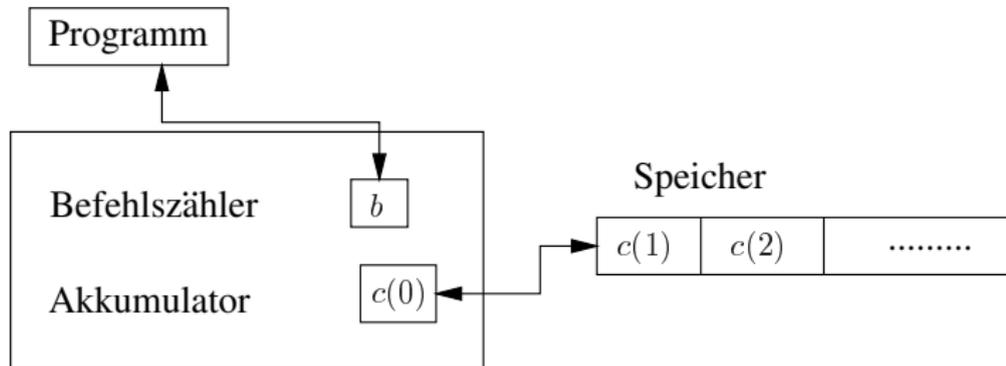
Hauptfrage in diesem Kapitel:

Welche Probleme sind berechenbar?

Die Registermaschine (RAM)

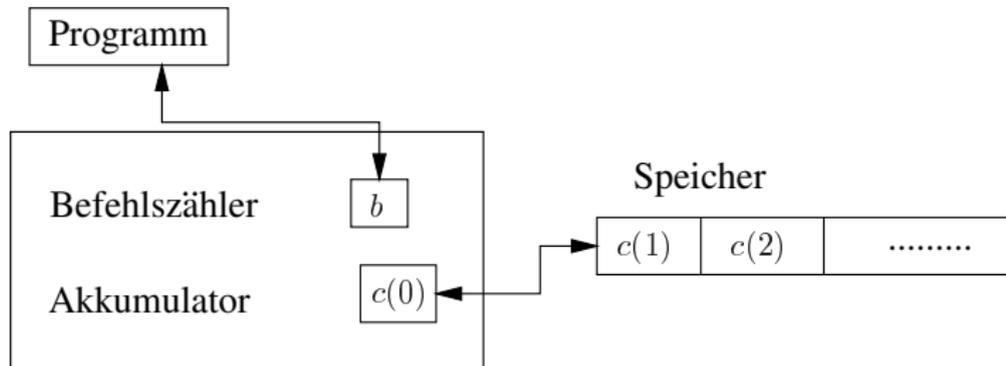
Die RAM besteht aus

- Befehlszähler (zeigt auf den nächsten Befehl im Programm),
- Akkumulatoren (endlicher Speicher zum Ausführen der Befehle),
- Registern (unendlicher Speicher), und
- Programm.



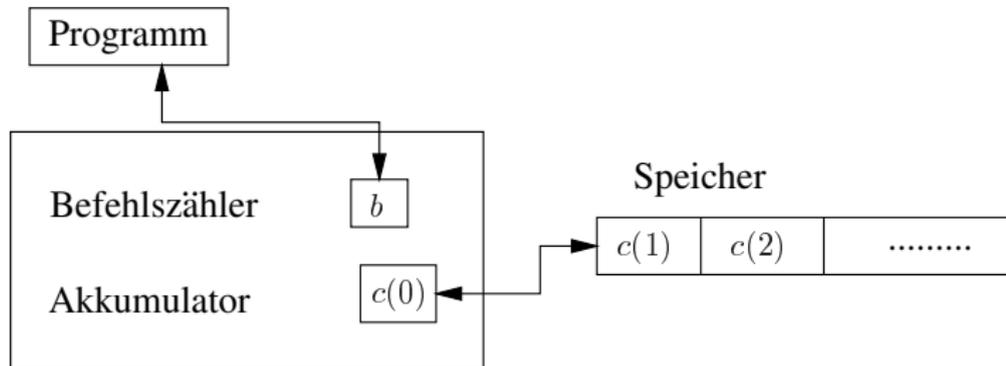
Die Registermaschine (RAM)

- Ein Programm besteht aus einer Folge von Befehlen.
- Programmzeilen sind durchnummeriert.
- Der Befehlszähler b startet bei 1 und enthält jeweils die Nummer des nächsten auszuführenden Befehls.



Die Registermaschine (RAM)

- In den ersten Registern steht zu Beginn der Berechnung die Eingabe.
- In den übrigen Registern steht 0.
- Am Ende der Berechnung stehen die Ausgabedaten in vorher festgelegten Registern.
- Den Inhalt des Registers i bezeichnen wir mit $c(i)$.



Befehle der Registermaschine (RAM)

| Befehl | Wirkung |
|----------------------------|--|
| LOAD i | $c(0) := c(i); \quad b := b + 1$ |
| STORE i | $c(i) := c(0); \quad b := b + 1$ |
| ADD i | $c(0) := c(0) + c(i); \quad b := b + 1$ |
| SUB i | $c(0) := \max\{0, c(0) - c(i)\}; \quad b := b + 1$ |
| MULT i | $c(0) := c(0) \cdot c(i); \quad b := b + 1$ |
| DIV i | $c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b + 1$ |
| GOTO j | $b := j$ |
| IF $c(0) \# \ell$ GOTO j | $\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b + 1 & \text{sonst} \end{cases}$ |
| END | $b := b$ |

wobei # aus
 $\{\leq, \geq, <, >, \neq, =\}$

Befehle der Registermaschine (RAM)

| Befehl | Wirkung |
|----------------------------|---|
| LOAD i | $c(0) := c(i); \quad b := b + 1$ |
| STORE i | $c(i) := c(0); \quad b := b + 1$ |
| ADD i | $c(0) := c(0) + c(i); \quad b := b + 1$ |
| SUB i | $c(0) := \max\{0, c(0) - c(i)\}; \quad b := b + 1$ |
| MULT i | $c(0) := c(0) \cdot c(i); \quad b := b + 1$ |
| DIV i | $c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b + 1$ |
| GOTO j | $b := j$ |
| IF $c(0) \# \ell$ GOTO j | $\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b + 1 & \text{sonst} \end{cases}$ wobei $\#$ aus $\{\leq, \geq, <, >, \neq, =\}$ |
| END | $b := b$ |

Befehle können modifiziert werden zu:

CLOAD, CSTORE, CADD, CSUB, CMULT, CDIV

Ersetze hierzu immer $c(i)$ durch die Konstante i .

Befehle der Registermaschine (RAM)

| Befehl | Wirkung |
|----------------------------|---|
| LOAD i | $c(0) := c(i); \quad b := b + 1$ |
| STORE i | $c(i) := c(0); \quad b := b + 1$ |
| ADD i | $c(0) := c(0) + c(i); \quad b := b + 1$ |
| SUB i | $c(0) := \max\{0, c(0) - c(i)\}; \quad b := b + 1$ |
| MULT i | $c(0) := c(0) \cdot c(i); \quad b := b + 1$ |
| DIV i | $c(0) := \left\lfloor \frac{c(0)}{c(i)} \right\rfloor; \quad b := b + 1$ |
| GOTO j | $b := j$ |
| IF $c(0) \# \ell$ GOTO j | $\begin{cases} b := j & \text{falls } c(0) \# \ell \\ b := b + 1 & \text{sonst} \end{cases}$ wobei $\#$ aus $\{\leq, \geq, <, >, \neq, =\}$ |
| END | $b := b$ |

Befehle können modifiziert werden zu:

INDLOAD, INDSTORE, INDADD, INDSUB, INDMULT, INDDIV

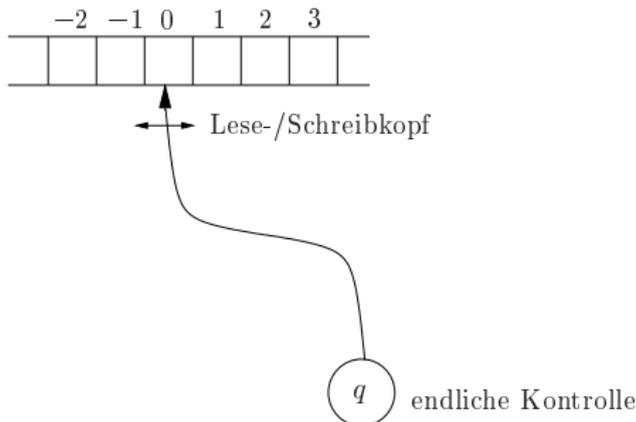
Ersetze hierzu immer $c(i)$ durch $c(c(i))$ (indirekte Addressierung).

- Üblicherweise wird das **uniforme** Kostenmodell verwendet.
- Dabei kostet jede Programmzeile bis auf END eine Einheit.
- Dieses Modell ist gerechtfertigt solange keine großen Zahlen auftreten.
- Ansonsten ist das **logarithmische** Kostenmodell realistischer.
- Kosten entsprechen dann der Länge der benutzten Zahlen.

Die Turing-Maschine (TM)

Eine TM besteht aus

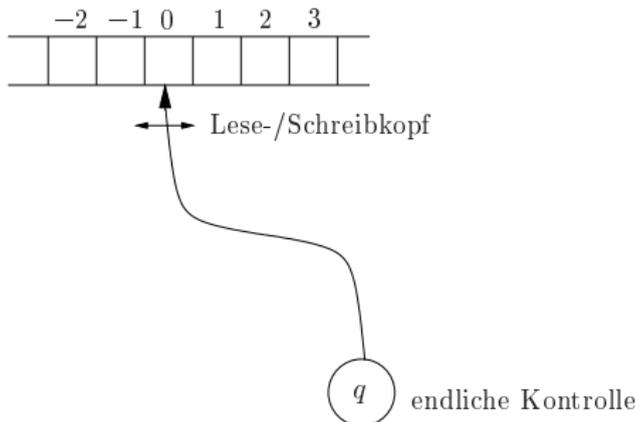
- beidseitig unendlichen Eingabe- und Rechenband,
- freibeweglichem Lese-/Schreibkopf, und
- endlicher Kontrolle.



Die Turing-Maschine (TM)

Die Kontrolle

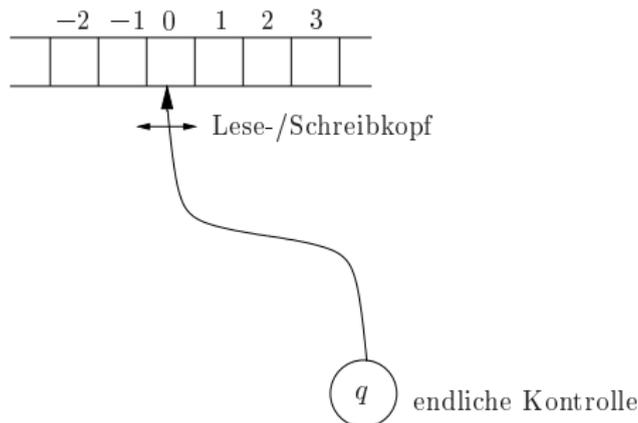
- ist immer in einem von endlich vielen Zuständen, und
- entspricht dem Befehlszähler der RAM.



Die Turing-Maschine (TM)

Das Eingabe- und Rechenband

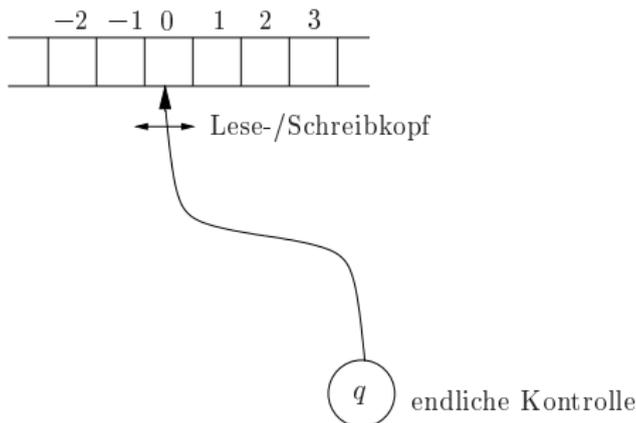
- enthält eine Folge von Symbolen (höchstens eins pro Zelle), und
- entspricht den Registern der RAM.



Die Turing-Maschine (TM)

Ausgehend vom aktuellen Zustand verhält sich die TM wie folgt:

- lese das Symbol auf der aktuellen Position des Lese-/ Schreibkopfes,
- gehe in einen Folgezustand über,
- überschreibe evtl. das Symbol, und
- bewege den Lese-/ Schreibkopf nach rechts, links oder gar nicht.

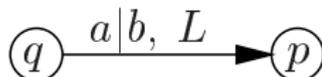


Formale Definition der Turing-Maschine

Eine deterministische Turing-Maschine ((D)TM) besteht aus:

- Q , einer endlicher Zustandsmenge,
- Σ , einem endlichen Eingabealphabet,
- \sqcup , einem Blanksymbol mit $\sqcup \notin \Sigma$,
- Γ , einem endlichen Bandalphabet mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$,
- $s \in Q$, einem Startzustand,
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$, einer Übergangsfunktion.
Dabei bedeutet L eine Bewegung des Lese-/Schreibkopfes nach links, R eine Bewegung nach rechts und N ein Stehenbleiben. Die Übergangsfunktion beschreibt, wie das aktuell eingelesene Zeichen verarbeitet werden soll.
- $F \subseteq Q$, einer Menge von Endzuständen.
Die Menge der Endzustände kann auch entfallen.

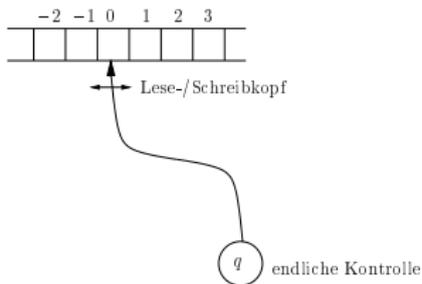
- Der Übergang $\delta(q, a) = (p, b, L)$ wird graphisch wie folgt dargestellt



Bedeutung:

Ist die Turing-Maschine im Zustand q und liest das Symbol a , so

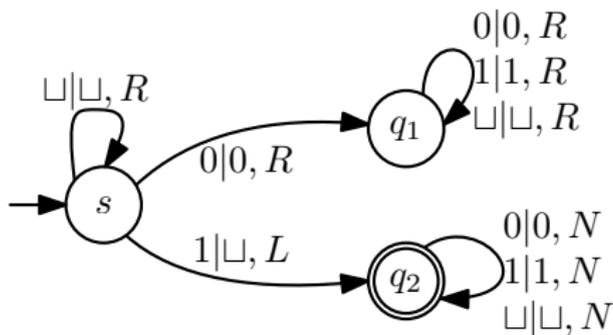
- überschreibt sie dieses a mit b ,
- geht auf dem Band eine Stelle nach links, und
- wechselt in den Zustand p .



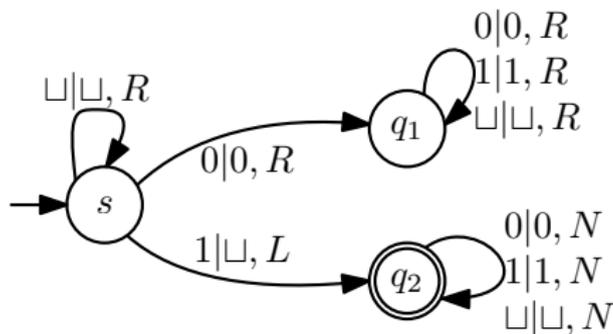
Konventionen

- Die Turing-Maschine startet im Zustand s .
- Der Lese-/Schreibkopf startet an der linken Stelle des Bandes, in der ein Eingabesymbol steht.
- Die Turing-Maschine stoppt, wenn sie
 - zum ersten Mal in einen Endzustand kommt, oder
 - in einem Zustand q ein Symbol a liest und $\delta(q, a) = (q, a, N)$ ist.
- Das bedeutet, dass Übergänge, die aus Endzuständen herausführen, überflüssig sind.

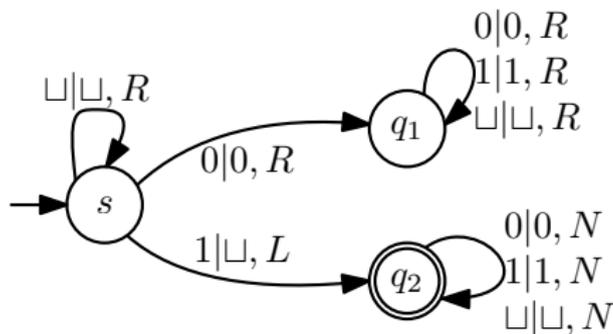
Beispiel-Turing-Maschine



Frage: Was erkennt / berechnet diese TM ?

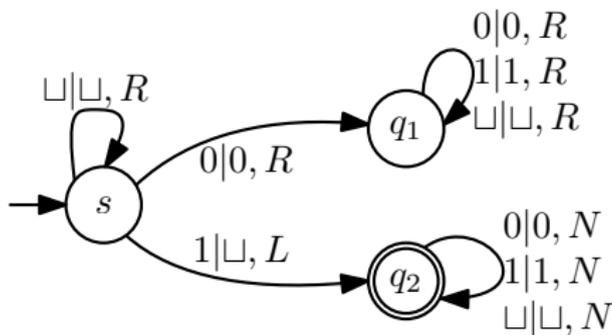


- Die TM erkennt alle Wörter aus $\{0, 1\}^*$, die mit einer Eins beginnen.
- Die TM löscht die führende Eins, falls vorhanden.
- Alles andere auf dem Band bleibt unverändert.
- Der Lese-/Schreibkopf steht nach dem Stop links neben der Stelle an der die führende Eins gelesen wurde.
- Der Zustand q_1 ist unwesentlich.

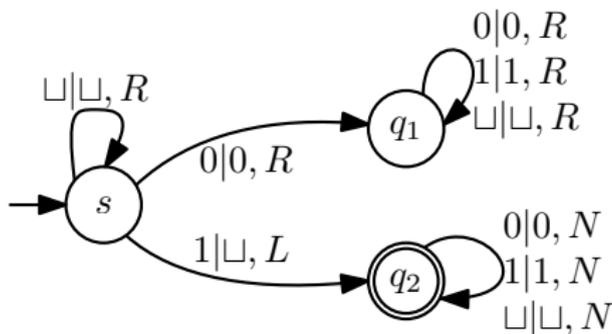


- Es gibt Eingaben, für die eine Turing-Maschine unter Umständen niemals stoppt.
- **Welche Eingaben sind dies in diesem Beispiel?**

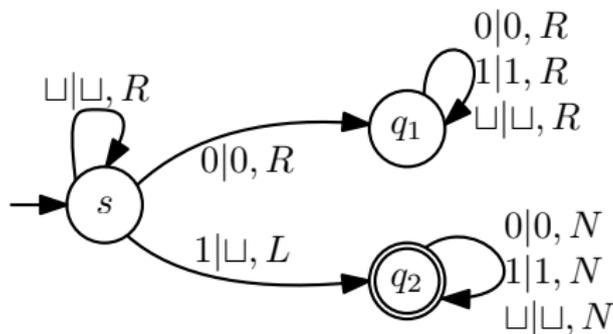
Beispiel-Turing-Maschine



- Die TM stoppt nicht, falls die Eingabe nicht mit Eins beginnt.



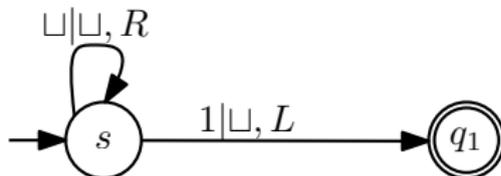
- Eine Turing-Maschine erkennt nicht nur eine Sprache,
- sondern sie verändert auch die Eingabe, und
- hat insofern auch eine Ausgabe
(= Inhalt des Bandes nach der Bearbeitung).
- Die Turing-Maschine realisiert also eine partielle Funktion $f: \Sigma^* \rightarrow \Gamma^*$.



- Die Turing-Maschine realisiert also eine partielle Funktion $f: \Sigma^* \rightarrow \Gamma^*$.
- Im Beispiel ist

$$f(w) = \begin{cases} v & \text{falls } w = 1v \\ \text{undefiniert} & \text{sonst} \end{cases}$$

- Oft werden wir die Turing-Maschine beziehungsweise deren Übergangsfunktion nur unvollständig beschreiben.
- Beispiel:



- Eine Vervollständigung ist immer möglich.
- Wenn für eine bestimmte Kombination q, a kein Übergang $\delta(q, a)$ definiert ist, dann stoppt die Turing-Maschine im Zustand q .
(z.B. setze $\delta(q, a) := (q, a, N)$)

- Eine Turing-Maschine **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn sie nach Lesen von w in einem Zustand aus F stoppt.
- Sie **akzeptiert** eine Sprache L genau dann, wenn sie ausschließlich Wörter $w \in L$ als Eingabe akzeptiert.
- Eine Sprache $L \subseteq \Sigma^*$ heißt **rekursiv** oder **entscheidbar**, wenn es eine Turing-Maschine gibt, die auf allen Eingaben stoppt und eine Eingabe w genau dann akzeptiert, wenn $w \in L$ gilt.
- Eine Sprache $L \subseteq \Sigma^*$ heißt **rekursiv-aufzählbar** oder **semi-entscheidbar**, wenn es eine Turing-Maschine gibt, die genau die Eingaben w akzeptiert für die $w \in L$.

Das Verhalten der Turing-Maschine für Eingaben $w \notin L$ ist damit nicht definiert. D.h., die Turing-Maschine stoppt entweder nicht in einem Endzustand oder aber stoppt gar nicht.