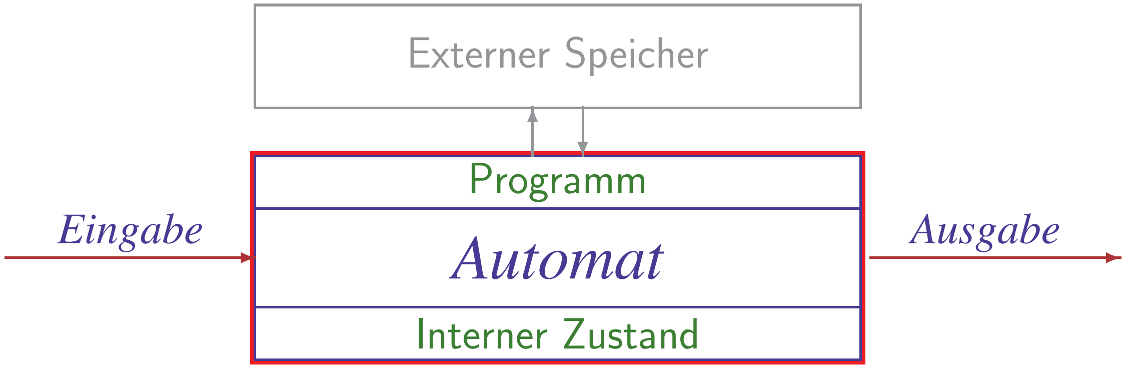
Theoretische Informatik I

Einheit 2

Endliche Automaten
&
Reguläre Sprachen

1. Deterministische endliche Automaten
2. Nichtdeterministische Automaten
3. Reguläre Ausdrücke
4. Grammatiken
5. Eigenschaften regulärer Sprachen

Automaten: das einfachste Maschinenmodell



Sichtweisen von Computern

- **Automaten stehen im Kern jeder Berechnung**
  – Schnelle, direkte Verarbeitung von Eingaben
  – Keine interne Speicherung von Daten
  – Speicher sind Teil der Umgebung

- **Endliche Automaten sind leicht zu analysieren**
  – Jede Berechnung endet nach einer festen Anzahl von Schritten
  – Keine Schleifen oder Seiteneffekte

VERWENDUNGSZWECKE FÜR ENDLICHE AUTOMATEN

Grundlegendes und vielseitiges Modellierungskonzept für viele Arten von Hard- & Software

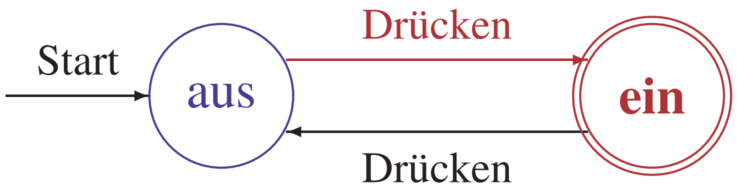
- **Steuerungsautomaten**
  – Alle Formen rein Hardware-gesteuerter automatischer Maschinen
  Waschmaschinen, Autos, Unterhaltungselektronik, Ampelanlagen, Computerprozessoren
- **Entwurf und Überprüfung digitaler Schaltungen**
  – Entwicklungswerkzeuge & Testsoftware beschreiben endliches Verhalten
- **Lexikalische Analyse in Compilern**
  – Schnelle Identifizierung von Bezeichnern, Schlüsselwörtern, …
- **Textsuche in umfangreichen Dokumenten**
  – Z.B. Suche nach Webseiten mithilfe von Schlüsselwörtern
- **Software mit endlichen Alternativen**
  – Kommunikationsprotokolle, Protokolle zum sicheren Datenaustausch …

Automaten beschreiben Sprachen

- **Generierte Sprache**
  – Menge aller möglichen Ausgaben des Automaten
- **Erkannte Sprache**
  – Menge aller Eingaben, die zur Ausgabe “ja” führen
  – Alternativ: letzter Zustand des Automaten muß ein “Endzustand” sein
- **Sprachen endlicher Automaten sind einfach**
  – Nur sehr einfach strukturierte Sprachen können beschrieben werden
  – Durch endliche Automaten beschreibbare Sprachen heißen **regulär**

Modelle zur Beschreibung regulärer Sprachen

- **Automaten: erkennen von Wörtern**
  – z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben



  – Zustände: aus, ein – Startzustand: aus – Endzustand: ein
  – Eingabesymbol: Drücken eines Schalters
  – Endzustand wird erreicht bei ungerader Anzahl von Drücken

- **Mathematische Mengennotation**
  – z.B.: $\{w \in \{\text{Drücken}\}^* \mid \exists i \in \mathbb{N}. |w| = 2i+1\}$, kurz $\{\text{Drücken}^{2i+1} | i \in \mathbb{N}\}$

- **Reguläre Ausdrücke: algebraische Strukturen**
  – z.B.: (DrückenDrücken)*Drücken

- **Grammatiken: Vorschriften für Spracherzeugung**
  – z.B.: $S \to$ Drücken, $S \to S$DrückenDrücken
  – Erzeugt nur ungerade Anzahl von Drücken-Symbolen

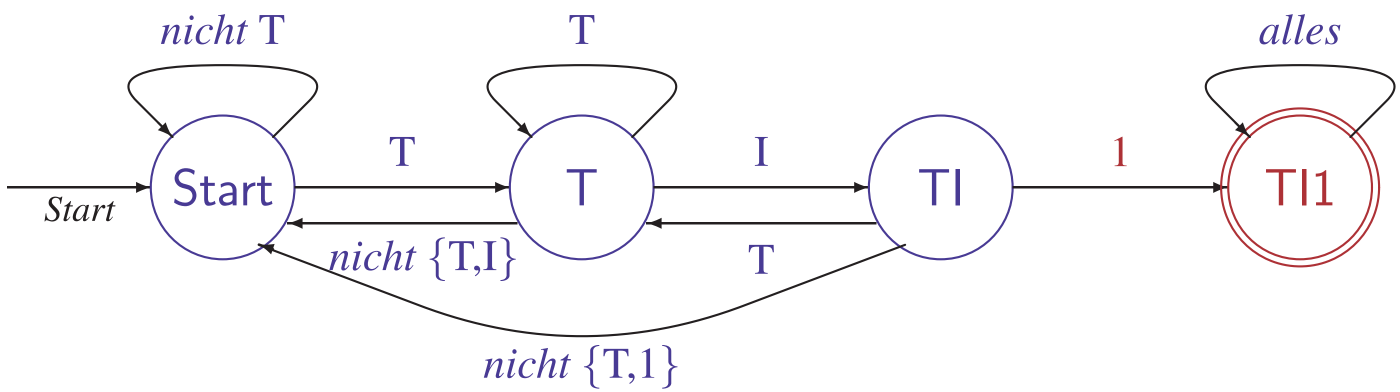
Theoretische Informatik I

Einheit 2.1

Deterministische Endliche Automaten

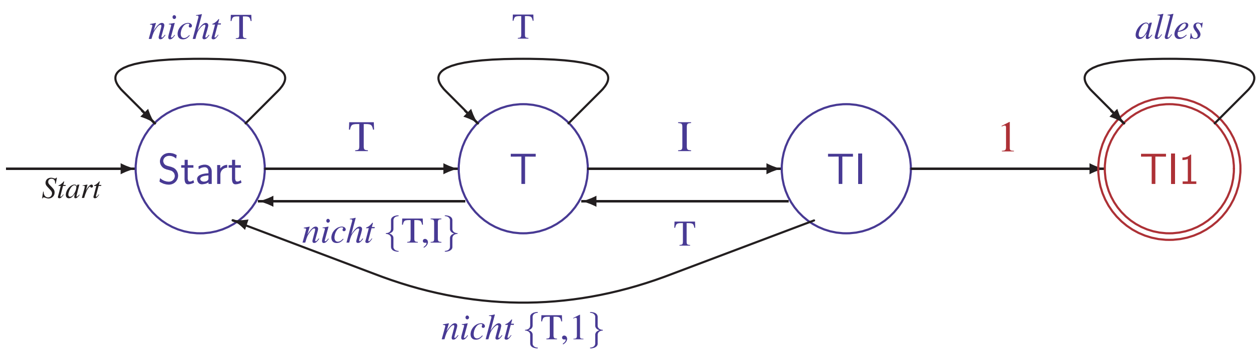
1. Arbeitsweise
2. Akzeptierte Sprache
3. Entwurf und Analyse
4. Automaten mit Ausgabe

Erkennung von Wörtern mit Automaten



- **Endliche Anzahl von Zuständen**
- **Ein Startzustand**
- **Regeln für Zustandsübergänge**
- **Eingabealphabet**: $\{A, .., Z, a, .., z,\ 0, .., 9,\ ?, !, ..\}$
- **Ein oder mehrere akzeptierende Endzustände**

ENDLICHE AUTOMATEN – MATHEMATISCH PRÄZISIERT



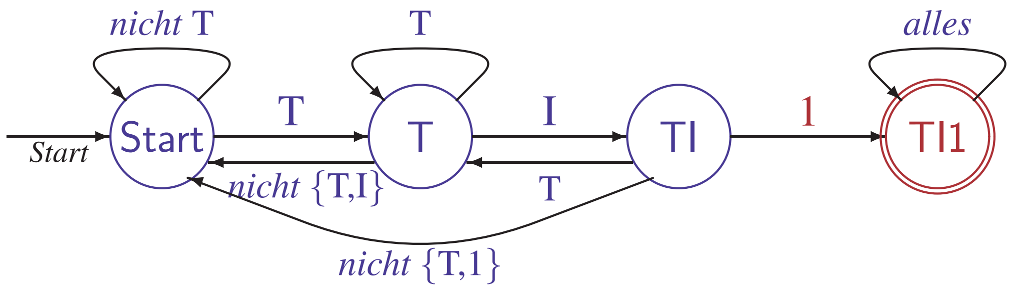
Ein **Deterministischer Endlicher Automat** (**DEA**)

ist ein 5-Tupel $A = (Q, \Sigma, \delta, q_0, F)$ mit

- Q nichtleere endliche **Zustandsmenge**
- Σ (endliches) **Eingabealphabet**
- $\delta{:}Q \times \Sigma \rightarrow Q$ **Zustandsüberführungsfunktion**
- $q_0 \in Q$ **Startzustand** (Anfangszustand)
- $F \subseteq Q$ Menge von **akzeptierenden Zuständen** (Endzustände)
  (Finale Zustände)

Beschreibung von Endlichen Automaten

- **Übergangsdiagramm**

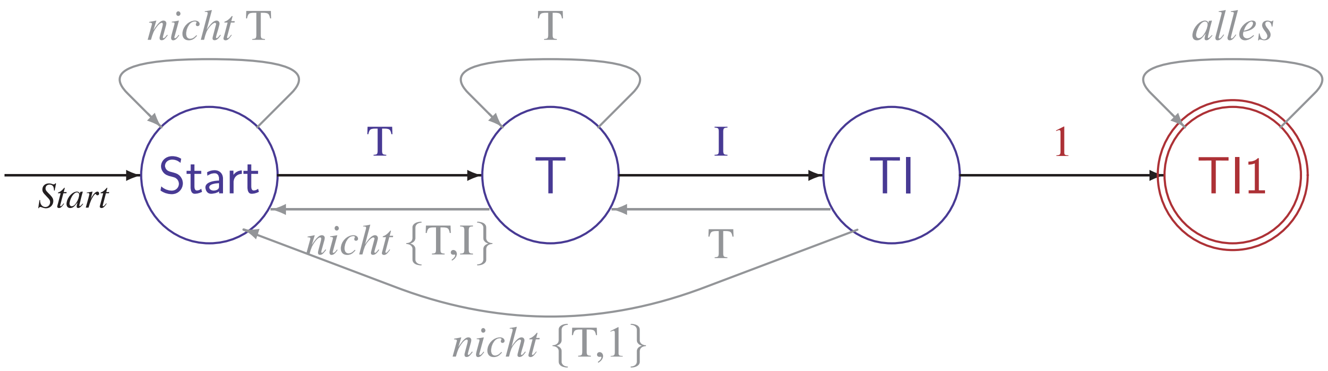


  – Jeder Zustand in Q wird durch einen Knoten (Kreise) dargestellt
  – Ist $\delta(q, a) = p$, so verläuft eine Kante von q nach p mit Beschriftung a (mehrere Beschriftungen derselben Kante möglich)
  – q_0 wird durch einen mit *Start* beschrifteten Pfeil angezeigt
  – Endzustände in F werden durch doppelte Kreise gekennzeichnet
  – Σ meist implizit durch Diagramm bestimmt

- **Übergangstabelle**
  – Tabellarische Darstellung der Funktion δ
  – Kennzeichnung von q_0 durch einen Pfeil
  – Kennzeichnung von F durch Sterne
  – Σ und Q meist implizit durch Tabelle bestimmt

		T	*I*	*1*	*sonst*
$\rightarrow$	S	T	S	S	S
	T	T	I	S	S
	I	T	S	1	S
*	1	1	1	1	1

Arbeitsweise von Endlichen Automaten



- **Anfangssituation**
  – Automat befindet sich im Startzustand q_0
- **Arbeitschritt**
  – Im Zustand q lese Eingabesymbol a,
  – Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p
- **Terminierung**
  – Eingabewort $w = a_1..a_n$ ist komplett gelesen, Automat im Zustand q_n
- **Ergebnis**
  – Eingabewort w wird akzeptiert, wenn $q_n \in F$, sonst wird w abgewiesen

ARBEITSWEISE VON DEAs – MATHEMATISCH PRÄZISIERT

- **Erweiterte Überführungsfunktion $\hat{\delta} : Q\times\Sigma^*\to Q$**
  - Schrittweise Abarbeitung der Eingabe mit δ von links nach rechts
  - Informal: $\hat{\delta}(q, w_1 w_2 ... w_n) = \delta(...(\delta(\delta(q, w_1), w_2), ...), w_n)$
  - Mathematisch präzise Beschreibung benötigt induktive Definition

$$\hat{\delta}(q, w) = \begin{cases} q & \text{falls } w{=}\epsilon, \\ \delta(\hat{\delta}(q, v), a) & \text{falls } w{=}v\,a \text{ für ein } v\in\Sigma^*, a\in\Sigma \end{cases}$$

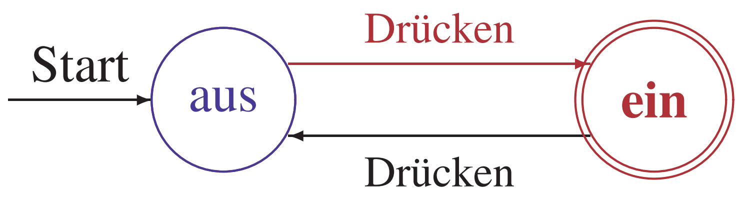
- **Von A akzeptierte Sprache**
  - Menge der Eingabewörter w, für die $\hat{\delta}(q_0, w)$ akzeptierender Zustand ist

$$L(A) = \{w\in\Sigma^* \mid \hat{\delta}(q_0, w)\in F\}$$

  - Auch: die von A erkannte Sprache

- **Reguläre Sprache**
  - Sprache, die von einem DEA A akzeptiert wird

Analyse der Sprache des Wechselschalters



- **Sprache: Eingaben, für die Automat eingeschaltet ist**
  - Teilmenge der Wörter über dem Alphabet $\Sigma = \{$Drücken$\}$
- **Automat A ist ein Wechselschalter**

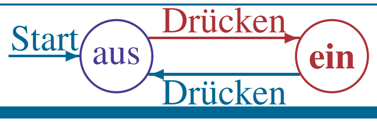
  Nach n-fachem Drücken ist A aus(ein)geschaltet, wenn n (un)gerade ist

  Zwei Eigenschaften sind zu zeigen:
  - $S_1(n)$: Ist n gerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$
  - $S_2(n)$: Ist n ungerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$

  Beweis: simultane Induktion analog zu Einheit 1, Folie, Details auf nächster Folie
- **Formale Beschreibung der Sprache von A**

  $L(A) = \{w \in \text{Drücken}^* \mid \hat{\delta}(\text{aus}, w) \in \{\text{ein}\}\} = \{\textbf{Drücken}^{2i+1} \mid i \in \mathbb{N}\}$

Detaillierter Induktionsbeweis



Um zu zeigen, daß der Automat ein Wechselschalter ist, zeigen wir durch Induktion, daß für alle $n \in \mathbb{N}$ die folgenden beiden Aussagen gelten

$S_1(n)$: n gerade $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^n) = aus

$S_2(n)$: n ungerade $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^n) = ein

Induktionsanfang n=0:

$S_1(n)$: 0 ist gerade und $\hat{\delta}$(aus, Drücken^0) = $\hat{\delta}$(aus, ϵ) = aus, also gilt Aussage $S_1(n)$.

$S_2(n)$: 0 ist nicht ungerade und $\hat{\delta}$(aus, Drücken^0) $\neq$ ein, also gilt die Äquivalenz $S_2(n)$, da jeweils die rechte und linke Seite falsch ist.

Induktionsannahme: $S_1(m)$ und $S_2(m)$ seien für ein beliebiges $m \in \mathbb{N}$ gezeigt.

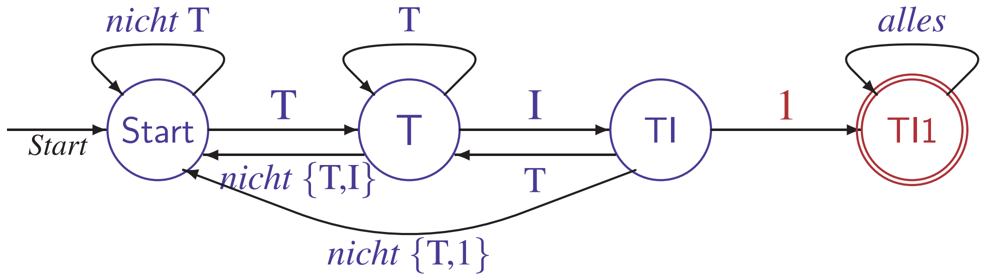
Induktionsschritt: Es sei $n = m+1$.

$S_1(n)$: Es ist $n = m+1$ gerade
- $\Leftrightarrow$ m ist ungerade
- $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^m) = ein (Induktionsannahme $S_2(m)$)
- $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^n) = aus

$S_2(n)$: Es ist $n = m+1$ ungerade
- $\Leftrightarrow$ m ist gerade
- $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^m) = aus (Induktionsannahme $S_1(m)$)
- $\Leftrightarrow$ $\hat{\delta}$(aus, Drücken^n) = ein

Aufgrund des Induktionsprinzips gilt $S_1(n)$ und $S_2(n)$ für alle $n \in \mathbb{N}$

Analyse der Sprache des “TI1”-Automaten

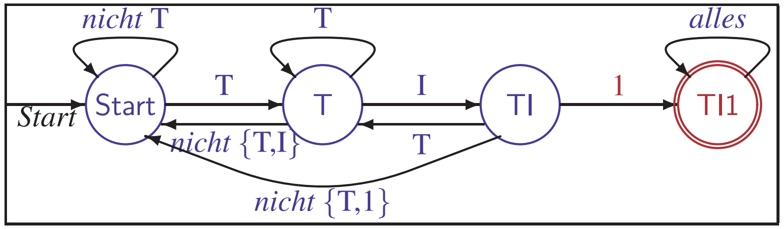


- **Sprache: Eingaben, die den Zustand TI1 erreichen**
  - Vermutung: Menge der Wörter, die `TI1` als Teilwort enthalten
  - Formale Beschreibung: $L(A) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*.\ w{=}u\mathtt{TI1}v\}$
  - Zu beweisen ist also: $\hat{\delta}(\mathsf{Start}, w) \in \{\mathsf{TI1}\} \Leftrightarrow \exists u, v \in \Sigma^*.\ w{=}u\mathtt{TI1}v$
- **Beweis zeigt, welche Wörter welchen Zustand erreichen**

  Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)
  - $S_3(w)$: $\hat{\delta}(\mathsf{Start}, w){=}\mathsf{TI} \Leftrightarrow \exists u \in \Sigma^*.\ w{=}u\mathtt{TI} \wedge \forall u, v \in \Sigma^*.\ w{\neq}u\mathtt{TI1}v$
  - $S_2(w)$: $\hat{\delta}(\mathsf{Start}, w){=}\mathsf{T} \Leftrightarrow \exists u \in \Sigma^*.\ w{=}u\mathtt{T} \wedge \forall u, v \in \Sigma^*.\ w{\neq}u\mathtt{TI1}v$
  - $S_1(w)$: $\hat{\delta}(\mathsf{Start}, w){=}\mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*.\ w{\neq}u\mathtt{TI1}v \wedge w{\neq}u\mathtt{TI} \wedge w{\neq}u\mathtt{T}$

  Die Behauptung folgt dann aus S_3 (und einer Induktion innerhalb von TI1)

Beweis der Sprache des "TI1"-Automaten



Beweise durch simultane (strukturelle) Induktion:

$S_3(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{TI} \;\Leftrightarrow\; \exists u \in \Sigma^*.\ w=u\mathtt{TI} \wedge \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v$

$S_2(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{T} \;\Leftrightarrow\; \exists u \in \Sigma^*.\ w=u\mathtt{T} \wedge \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v$

$S_1(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{Start} \Leftrightarrow \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v \wedge w\neq u\mathtt{TI} \wedge w\neq u\mathtt{T}$

Induktionsanfang $w=\epsilon$:

$S_1(w)$: $\hat\delta(\mathsf{Start}, \epsilon)=\mathsf{Start}$ gilt und $w=\epsilon$ hat keine der drei "verbotenen" Formen √

$S_2(w)$: $\hat\delta(\mathsf{Start}, \epsilon)=\mathsf{T}$ gilt nicht und $w=\epsilon$ endet nicht mit $\mathtt{T}$ √

$S_3(w)$: $\hat\delta(\mathsf{Start}, \epsilon)=\mathsf{TI}$ gilt nicht und $w=\epsilon$ endet nicht mit $\mathtt{TI}$ √

Induktionsannnahme: die Aussagen $S_1(w')$–$S_3(w')$ seien für ein beliebiges $w' \in \Sigma^*$ gezeigt.

Induktionsschritt: Es sei $w = w'a$ für ein beliebiges $a \in \Sigma$.

$S_1(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{Start}$

$\Leftrightarrow (\hat\delta(\mathsf{Start}, w')=\mathsf{Start} \wedge a\neq\mathtt{T}) \vee (\hat\delta(\mathsf{Start}, w')=\mathsf{T} \wedge a\notin\{\mathtt{T},\mathtt{I}\}) \vee (\hat\delta(\mathsf{Start}, w')=\mathsf{TI} \wedge a\notin\{\mathtt{T},\mathtt{1}\})$

$\Leftrightarrow \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v \wedge w\neq u\mathtt{TI} \wedge w\neq u\mathtt{T}$ mit $S_1(w')$, $S_2(w')$, $S_3(w')$ √

$S_2(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{T}$

$\Leftrightarrow (\hat\delta(\mathsf{Start}, w')=\mathsf{Start} \wedge a=\mathtt{T}) \vee (\hat\delta(\mathsf{Start}, w')=\mathsf{T} \wedge a=\mathtt{T}) \vee (\hat\delta(\mathsf{Start}, w')=\mathsf{TI} \wedge aa=\mathtt{T})$

$\Leftrightarrow \exists u \in \Sigma^*.\ w=u\mathtt{T} \wedge \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v$ mit $S_1(w')$, $S_2(w')$, $S_3(w')$ √

$S_3(w)$: $\hat\delta(\mathsf{Start}, w)=\mathsf{TI}$

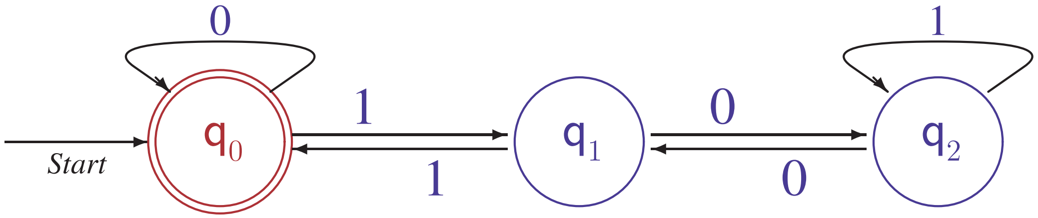
$\Leftrightarrow (\hat\delta(\mathsf{Start}, w')=\mathsf{T} \wedge a=\mathtt{I})$

$\Leftrightarrow \exists u \in \Sigma^*.\ w=u\mathtt{TI} \wedge \forall u,v \in \Sigma^*.\ w\neq u\mathtt{TI1}v$ mit $S_2(w')$ √

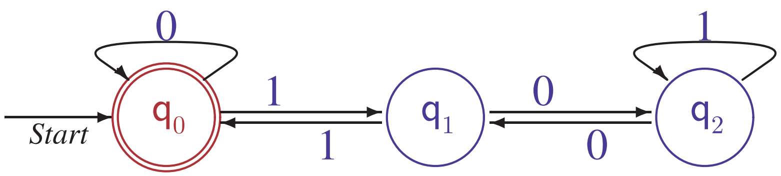
Aufgrund des Induktionsprinzips gilt $S_1(w)$, $S_2(w)$ und $S_3(w)$ für alle $w \in \Sigma^*$.

Erkenne, ob eine Binärzahl durch 3 teilbar ist

- **Binärzahl wird von links nach rechts gelesen**
  - Eingabewort $w = w_0..w_n$ entspricht der Zahl $n = r_b(w) = \Sigma_{j=0}^{n} w_j \cdot 2^{n-j}$
- **Konstruktion des Automaten aus induktiver Beweisidee**
  - Das Eingabewort $w = \epsilon$ entspricht der Zahl $n = 0$
  - Entspricht v der Zahl i, dann entspricht $w = va$ der Zahl $n = 2i{+}a$
  - Die Zahl n ist durch 3 teilbar wenn $n \bmod 3 = 0$ (Divisionsrest)
  - Wir wissen $2i{+}a \bmod 3 = 2(i \bmod 3) + a \bmod 3$
- **Drei Zustände sind erforderlich**
  - Zustand $q_i \hat{=}$ bisher gelesene Binärzahl hat Divisionsrest i (modulo 3)
  - Zustandsübergänge erhalten "Bedeutung": $\delta(q_i, a) = q_{2i+a \bmod 3}$
  - Resultierender DEA über Alphabet $\Sigma = \{0, 1\}$



ZEIGE $L(A) = \{w \mid r_b(w) \bmod 3 = 0\}$



- **Zeige durch simultane strukturelle Induktion über w**
  – $S_j(w)$: $\hat{\delta}(q_0, w)=q_j \Leftrightarrow r_b(w) \bmod 3 = j$ für $j \in \{0,1,2\}$
- **Induktionsanfang $w=\epsilon$:**
  – Es ist $\hat{\delta}(q_0, \epsilon)=q_0$ und $r_b(\epsilon) \bmod 3 = 0$
  – Damit gilt $S_0(w)$ und auch $S_1(w)$ und $S_2(w)$ (beide Seiten von $\Leftrightarrow$ sind falsch) $\checkmark$
- **Induktionsannnahme: $S_j(w')$ sei gezeigt für $w' \in \Sigma^*$ und $j \in \{0,1,2\}$**
- **Induktionsschritt: sei $w = w'a$ für ein beliebiges $a \in \Sigma$**

$$\begin{aligned}
& \hat{\delta}(q_0, w)=q_j \\
\Leftrightarrow\; & \exists i. \hat{\delta}(q_0, w')=q_i \wedge \delta(q_i, a) = q_j \\
\Leftrightarrow\; & \exists i. r_b(w') \bmod 3 = i \wedge j = 2i+a \bmod 3 \\
\Leftrightarrow\; & r_b(w) \bmod 3 \;=\; 2r_b(w')+a \bmod 3 \;=\; 2(r_b(w') \bmod 3)+a \bmod 3 \;=\; j \qquad \checkmark
\end{aligned}$$

- **Damit gilt $S_j(w)$ für alle und es folgt $w \in \Sigma^*$ und $j \in \{0,1,2\}$**
  $L(A) = \{w \mid \hat{\delta}(q_0, w)=q_0\} = \{w \mid r_b(w) \bmod 3 = 0\}$

Alternative Beschreibung der Arbeitsweise von DEAs

- **Konfiguration: 'Gesamtzustand' von Automaten**
  - Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
  - Formal dargestellt als Tupel $\boldsymbol{K} = (\boldsymbol{q},\boldsymbol{w}) \in Q \times \Sigma^*$
- **Konfigurationsübergangsrelation $\vdash^*$**
  - Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
  - $(\boldsymbol{q},\boldsymbol{aw}) \vdash (\boldsymbol{p},\boldsymbol{w})$, falls $\delta(q,a) = p$
  - $\boldsymbol{K_1} \vdash^* \boldsymbol{K_2}$, falls $K_1 = K_2$ oder

    es gibt eine Konfiguration K mit $K_1 \vdash K$ und $K \vdash^* K_2$
- **Akzeptierte Sprache**
  - Menge der Eingaben, für die $\vdash^*$ zu akzeptierendem Zustand führt

$$\boldsymbol{L(A) = \{w \in \Sigma^* \mid \exists p \in F.\ (q_0, w) \vdash^* (p,\epsilon)\}}$$

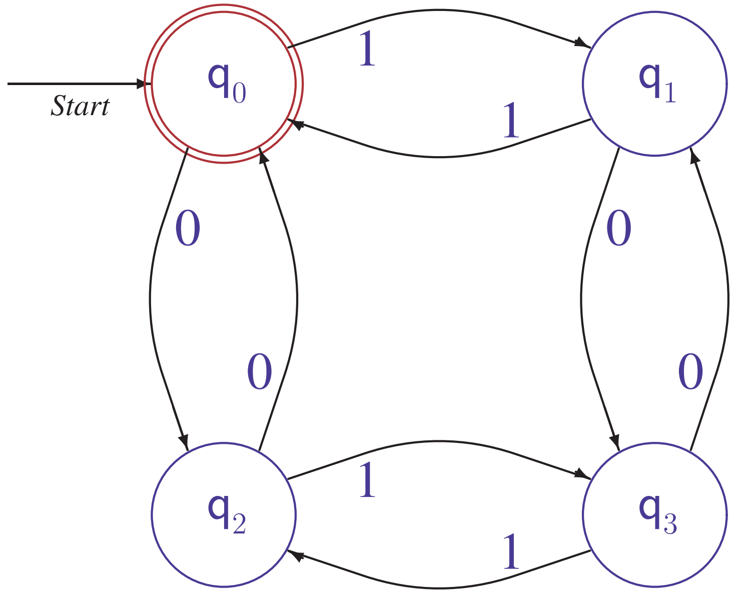
Für DEAs weniger intuitiv, aber leichter zu verallgemeinern

DEA FÜR $L = \{w \in \{0,1\}^* \mid w \textit{ enthält gerade Anzahl von 0 und 1}\}$

Codiere Anzahl der gelesener 0/1 im Zustand

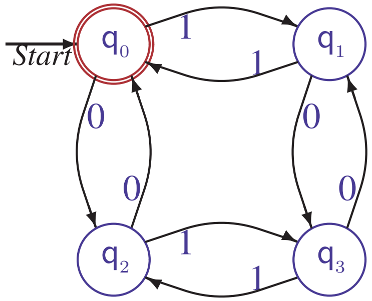
$q_0 \hat{=}$ (gerade,gerade) $q_1 \hat{=}$ (gerade,ungerade)

$q_2 \hat{=}$ (ungerade,gerade) $q_3 \hat{=}$ (ungerade,ungerade)



Korrektheit: gegenseitige strukturelle Induktion

Korrektheitsbeweis mit Konfigurationen



- **Zeige simultan für alle Wörter $w, v \in \{0,1\}^*$:**

  (1) $(q_0, w\,v) \vdash^* (q_0, v) \;\Leftrightarrow\;$ es gilt $g_0(w)$ und $g_1(w)$

  (2) $(q_0, w\,v) \vdash^* (q_1, v) \;\Leftrightarrow\;$ es gilt $g_0(w)$ und $u_1(w)$

  (3) $(q_0, w\,v) \vdash^* (q_2, v) \;\Leftrightarrow\;$ es gilt $u_0(w)$ und $g_1(w)$

  (4) $(q_0, w\,v) \vdash^* (q_3, v) \;\Leftrightarrow\;$ es gilt $u_0(w)$ und $u_1(w)$

  $g_0(w) \,\hat{=}\, w$ hat gerade Anzahl von Nullen, $u_0(w) \,\hat{=}\, w$ hat ungerade Anzahl von Nullen, …

- **Basisfall $w = \epsilon$:**

  – Per Definition gilt $(q_0, v) \vdash^* (q_0, v)$ und $g_0(w)$ und $g_1(w)$ $\surd$

- **Schrittfall $w = ua$ für ein $u \in \Sigma^*$, $a \in \Sigma$:**

  (1) Es gelte $(q_0, w\,v) \vdash^* (q_0, v)$.

  Dann gilt $(q_0, u\,a\,v) \vdash^* (p, a\,v) \vdash (q_0, v)$ für einen Zustand p.

  Falls $a = 0$, dann ist $p = q_2$ und nach (3) folgt $u_0(u)$ und $g_1(u)$.

  Falls $a = 1$, dann ist $p = q_1$ und nach (2) folgt $g_0(u)$ und $u_1(u)$.

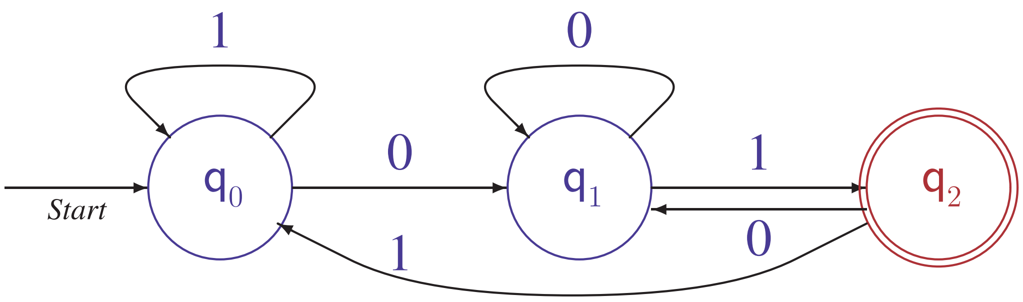
  Für $w = ua$ folgt somit $g_0(w)$ und $g_1(w)$. $\surd$

  Gegenrichtung durch Umkehrung des Arguments. (2), (3), (4) analog.

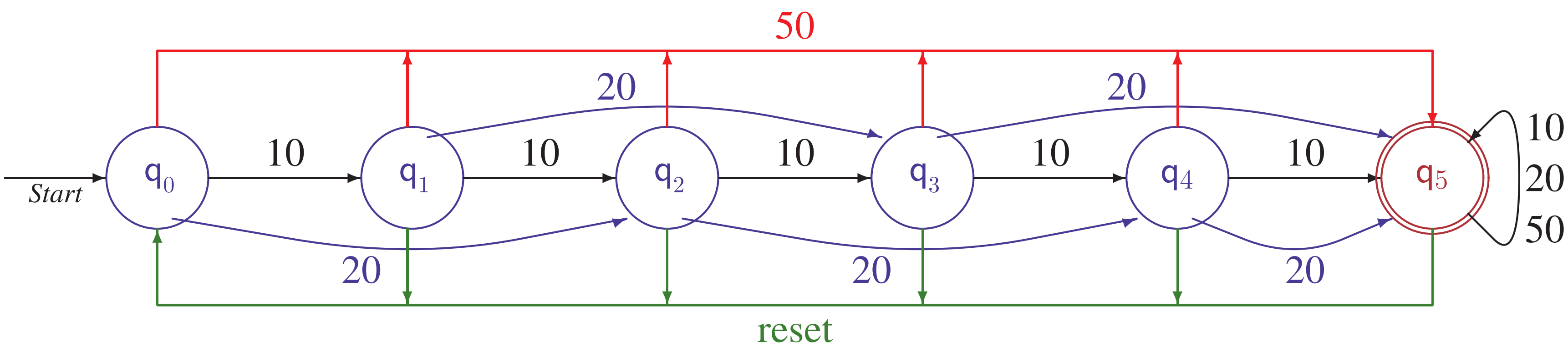
- **Es folgt** $w \in L(A) \;\Leftrightarrow\; (q_0, w) \vdash^* (q_0, \epsilon) \;\Leftrightarrow\; g_0(w) \wedge g_1(w) \;\Leftrightarrow\; w \in L$

Weitere Beispiele endlicher Automaten

- **Erkenne Strings, die mit `01` enden**

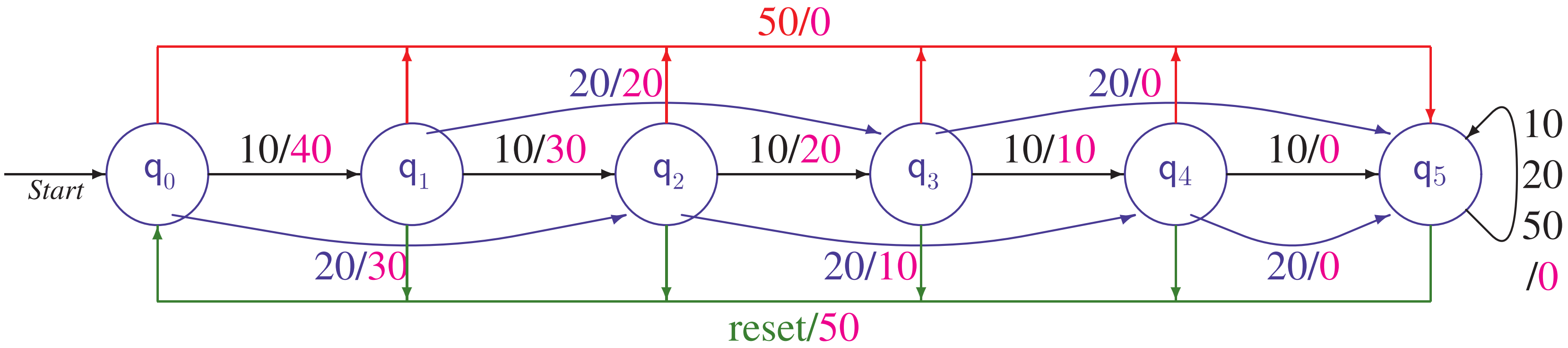


- **50c Kaffeeautomat**
  - Akzeptiert 10,20,50c Münzen, gibt kein Geld zurück, mit Reset-Taste



Endliche Automaten mit Ausgabefunktion

- **50c Kaffeeautomat mit Restbetragsanzeige**



  – Münzeinwurf führt zu Zustandsänderung und erzeugt Ausgabe

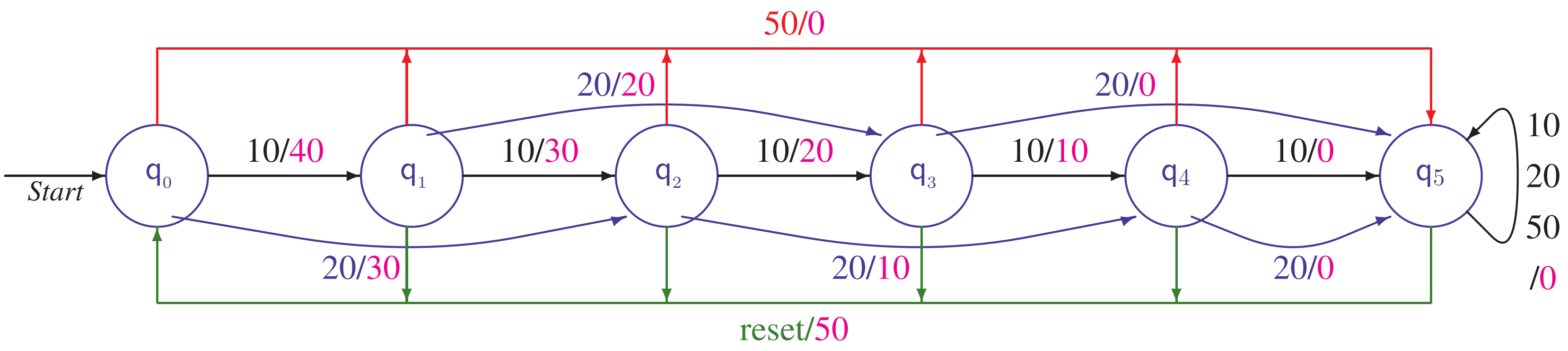
- **Formalisierungen von Automaten mit Ausgabe**
  – Mealy-Automaten: Ausgabefunktion abhängig von Eingabe & Zustand
  – Moore-Automaten: Ausgabefunktion nur von Zustand abhängig

- **Automaten mit Ausgabe sind keine echte Erweiterung**
  – Mealy- und Moore-Automaten sind äquivalent
  – DEAs können Mealy-/Moore-Automaten simulieren und umgekehrt

Mehr dazu im Anhang

ANHANG

Mealy-Automaten – mathematisch präzisiert



Ein **Mealy-Automat** ist ein 6-Tupel $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

- Q nichtleere endliche **Zustandsmenge**
- Σ (endliches) **Eingabealphabet**
- Δ (endliches) **Ausgabealphabet**
- $\delta : Q \times \Sigma \to Q$ **Zustandsüberführungsfunktion**
- $\lambda : Q \times \Sigma \to \Delta$ **Ausgabefunktion**
- $q_0 \in Q$ **Startzustand**

Arbeitsweise von Mealy-Automaten analog zu DEAs

- **Anfangssituation:** Automat im Startzustand q_0
- **Arbeitschritt**
  - Im Zustand q lese Eingabesymbol a,
  - Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p
  - Bestimme $x = \lambda(q,a)$ und gebe dieses Symbol aus
- **Terminierung:** Eingabewort $w = a_1..a_n$ ist komplett gelesen
- **Ausgabewort:** Verkettung der ausgegebenen Symbole $x_1..x_n$

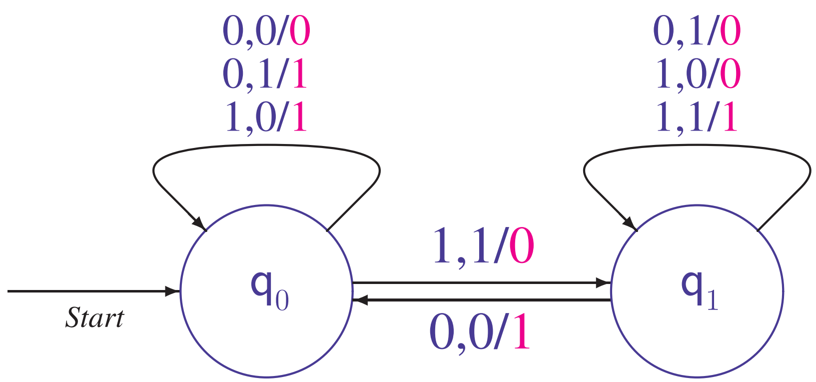
- **Erweiterte Ausgabefunktion $\hat{\lambda} : Q\times\Sigma^*\to\Delta^*$**
  - Schrittweise Erzeugung der Ausgabe mit Abarbeitung der Eingabe
  - Formal: Induktive Definition

$$\hat{\lambda}(q,w) = \begin{cases} \epsilon & \text{falls } w{=}\epsilon, \\ \hat{\lambda}(q,v)\circ\lambda(\hat{\delta}(q,v),a) & \text{falls } w{=}v\,a \text{ für ein } a\in\Sigma \end{cases}$$

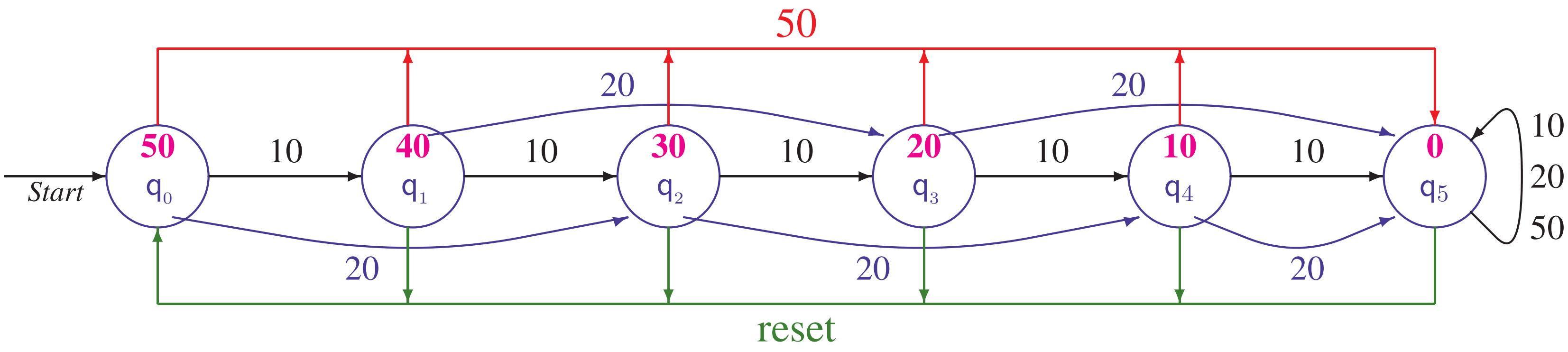
- **Von M berechnete Funktion: $f_M(w) = \hat{\lambda}(q_0,w)$**

Mealy-Automat für (inverse) Binäraddition

- **Addition von Bitpaaren von rechts nach links**
  - Eingabealphabet $\Sigma = \{0,1\}\times\{0,1\}$
  - Ausgabealphabet $\Delta = \{0,1\}$
- **Zwei Zustände sind erforderlich**
  - Zustand q_0: A kann Addition zweier Bits direkt ausführen
  - Zustand q_1: A hat bei Addition einen Übertrag zu berücksichtigen
- **Zugehöriger Mealy-Automat**



Moore-Automaten – mathematisch präzisiert



Ein **Moore-Automat** ist ein 6-Tupel $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

- Q nichtleere endliche **Zustandsmenge**
- Σ (endliches) **Eingabealphabet**
- Δ (endliches) **Ausgabealphabet**
- $\delta{:}Q\times\Sigma \to Q$ **Zustandsüberführungsfunktion**
- $\lambda{:}Q \to \Delta$ **Ausgabefunktion**
- $q_0 \in Q$ **Startzustand**

ARBEITSWEISE VON MOORE-AUTOMATEN ANALOG ZU DEAS

- **Anfangssituation:** Automat im Startzustand q_0, Ausgabe $x_0 = \lambda(q_0)$
- **Arbeitschritt**
  - Im Zustand q lese Eingabesymbol a,
  - Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p
  - Bestimme $x = \lambda(p)$ und gebe dieses Symbol aus
- **Terminierung:** Eingabewort $w = a_1..a_n$ ist komplett gelesen
- **Ausgabewort:** Verkettung der ausgegebenen Symbole $x_0x_1..x_n$

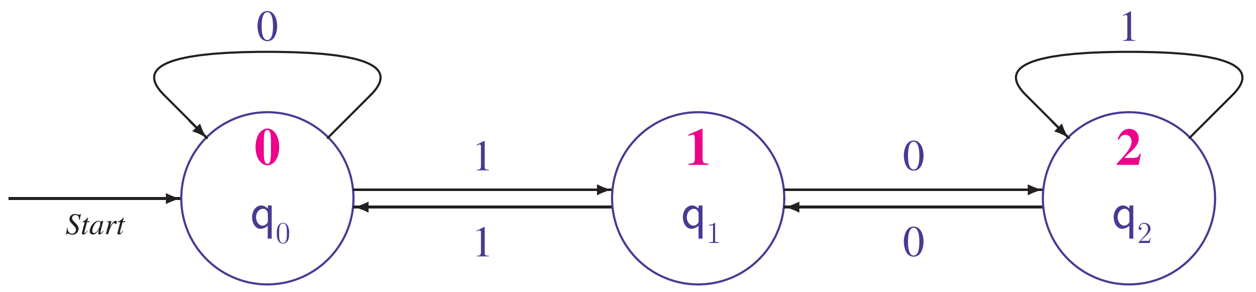
- **Erweiterte Ausgabefunktion $\hat{\lambda} : Q\times\Sigma^*\to\Delta^*$**
  - Schrittweise Erzeugung der Ausgabe mit Abarbeitung der Eingabe
  - Formal: Induktive Definition

$$\hat{\lambda}(q,w) = \begin{cases} \lambda(q) & \text{falls } w{=}\epsilon, \\ \hat{\lambda}(q,v)\circ\lambda(\hat{\delta}(q,v\,a)) & \text{falls } w{=}v\,a \text{ für ein } a\in\Sigma \end{cases}$$

- **Von M berechnete Funktion: $f_M(w) = \hat{\lambda}(q_0,w)$**

Moore-Automat für Divisionsrest

- **Eingabe einer Bitfolge von links nach rechts**
  - Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl n
  - Ausgabe ist jeweils “n mod 3”
  - Eingabealphabet $\Sigma = \{0, 1\}$, Ausgabealphabet $\Delta = \{0, 1, 2\}$
- **Drei Zustände sind erforderlich**
  - Zustand q_0: Bisheriger Divisionsrest ist 0 (Ausgabe 0)
  - Zustand q_1: Bisheriger Divisionsrest ist 1 (Ausgabe 1)
  - Zustand q_2: Bisheriger Divisionsrest ist 2 (Ausgabe 2)
  - Zustandsüberführungsregel $\delta(q_i, j) = q_{2*i+j \, mod \, 3}$
- **Zugehöriger Moore-Automat**



Moore-Automaten sind äquivalent zu DEAs

Gegenseitige Simulation ist möglich

- **Jede Sprache L ist als Funktion beschreibbar**
  - $\chi_L(w) = \begin{cases} 1 & \text{falls } w \in L, \\ 0 & \text{sonst} \end{cases}$ charakteristische Funktion von L
  - Charakteristische Funktionen akzeptierter Sprachen sind berechenbar

  Satz: **L regulär $\Leftrightarrow$ χ_L "Moore-berechenbar"**

- **Jede Funktion f ist als Menge beschreibbar**
  - **graph(f)** = $\{(w, v) \mid f(w) = v\}$
  - **graph*(f)** = $\{(w_1, v_0, v_1)..(w_n, v_n) \mid f(w_1..w_n) = v_0..v_n\}$
  - DEAs können Graphen berechneter Funktionen akzeptieren

  Satz: **f Moore-berechenbar $\Leftrightarrow$ graph*(f) reguläre Sprache**

Beweis der Äquivalenz (Skizze)

- **L regulär $\Leftrightarrow$ χ_L "Moore-berechenbar"**
  - Zu $A = (Q, \Sigma, \delta, q_0, F)$ konstruiere $M = (Q, \Sigma, \{0{,}1\}, \delta, \lambda, q_0)$

    mit $\lambda(q) = \begin{cases} 1 & \text{falls } q \in F, \\ 0 & \text{sonst} \end{cases}$
  - Dann ist $w \in L(A)$ genau dann, wenn $f_M(w) = v1$ für ein $v \in \{0,1\}^*$

    $\chi_L(w)$ ist das letzte Ausgabesymbol von $f_M(w)$

- **f Moore-berechenbar $\Leftrightarrow$ graph$^*(f)$ regulär**
  - Zu $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ konstruiere $A = (Q \cup \{q_s, q_f\}, \Sigma', \delta', q_s, Q)$

    mit $\Sigma' = \Sigma \times (\Delta \cup \{\lambda(q_0)\} \times \Delta)$

    $$\delta'(q, (a, b)) = \begin{cases} \delta(q_0, a) & \text{falls } q{=}q_s, b = (\lambda(q_0), \lambda(\delta(q_0, a)), \\ \delta(q, a) & \text{falls } \lambda(\delta(q, a)) = b, \\ q_f & \text{sonst} \end{cases}$$
  - Dann $f_M(w_1..w_n) = v_0..v_n$ genau dann, wenn $(w_1, v_0, v_1)..(w_n, v_n) \in L(A)$

Mehr zu Automaten mit Ausgabe im Buch von Vossen & Witt