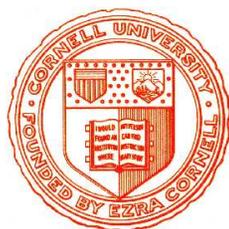


Theoretische Informatik I

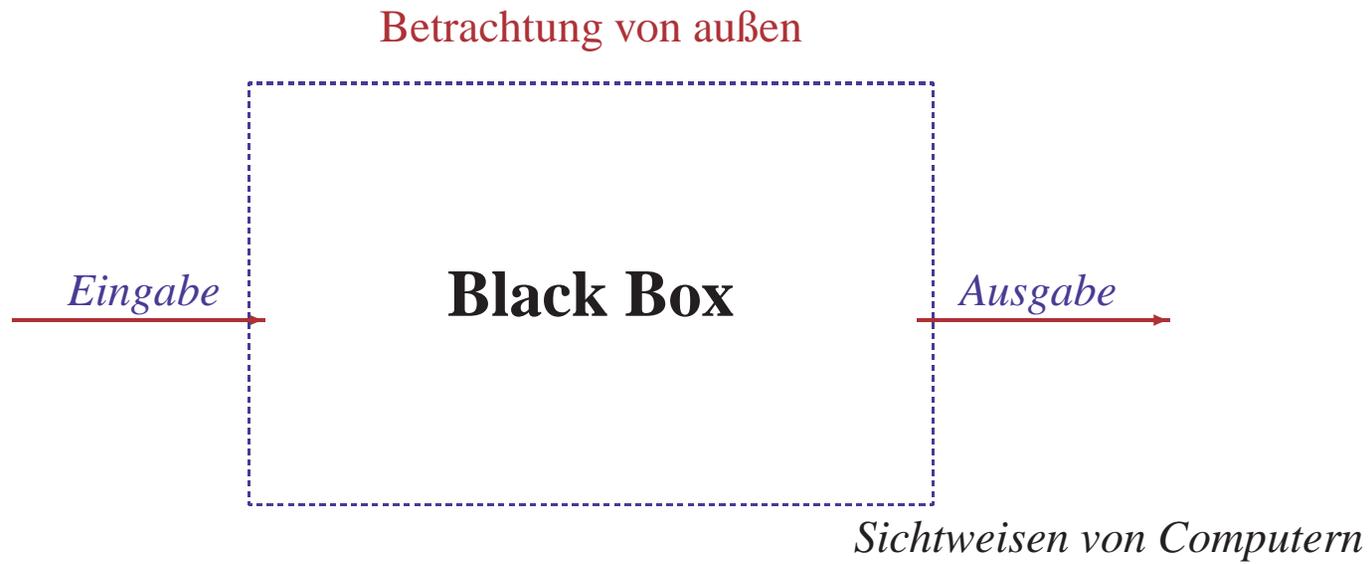
Einheit 2

Endliche Automaten & Reguläre Sprachen

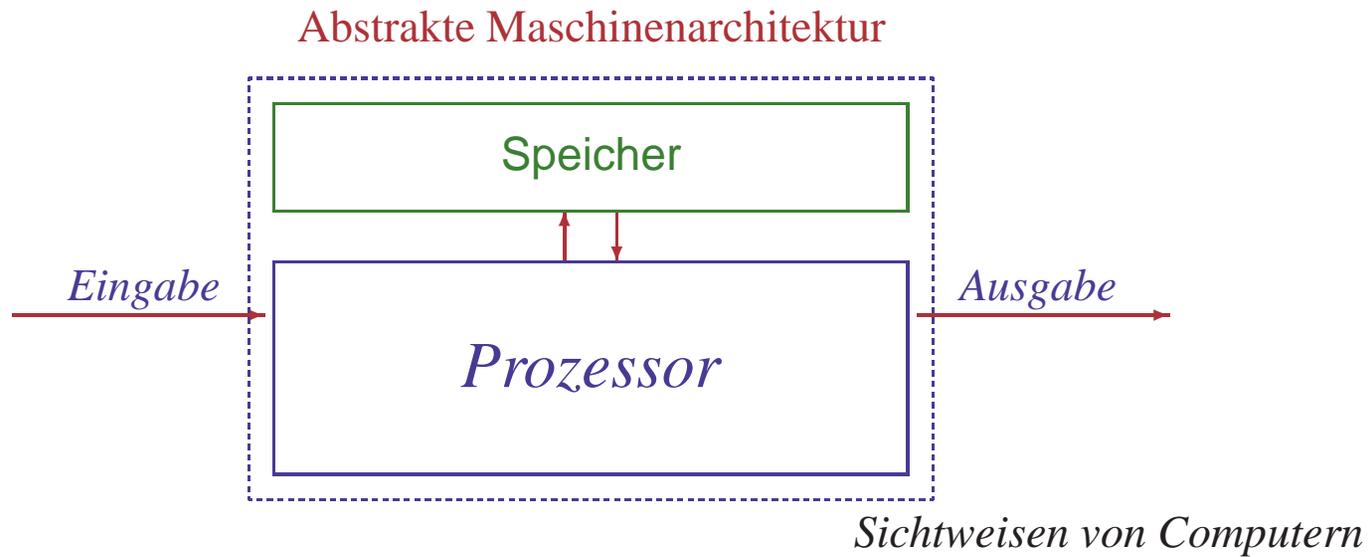


1. Deterministische endliche Automaten
2. Nichtdeterministische Automaten
3. Reguläre Ausdrücke
4. Grammatiken
5. Eigenschaften regulärer Sprachen

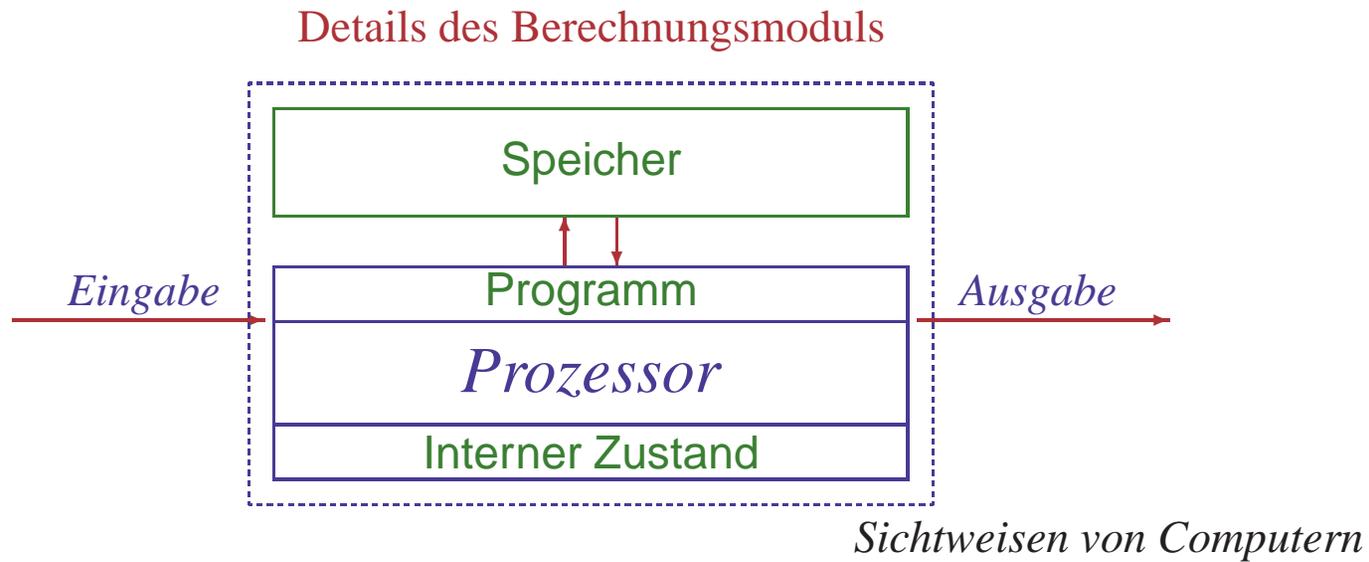
AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL



AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL

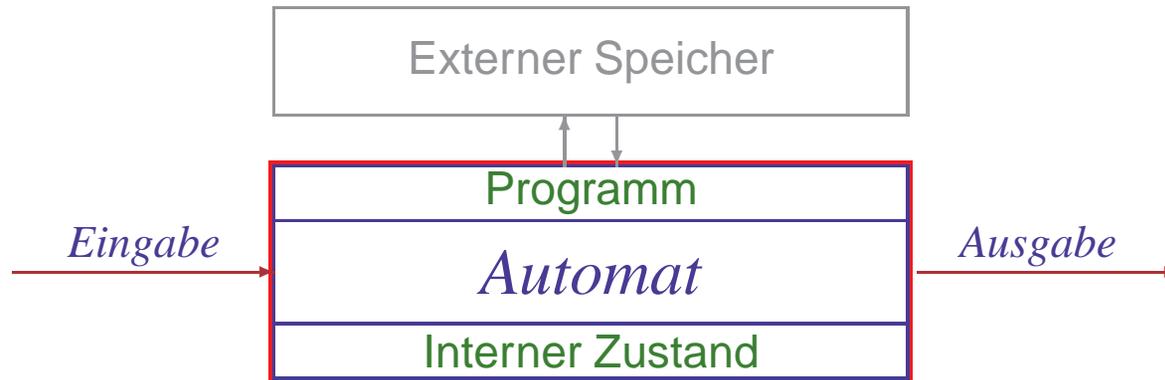


AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL



AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL

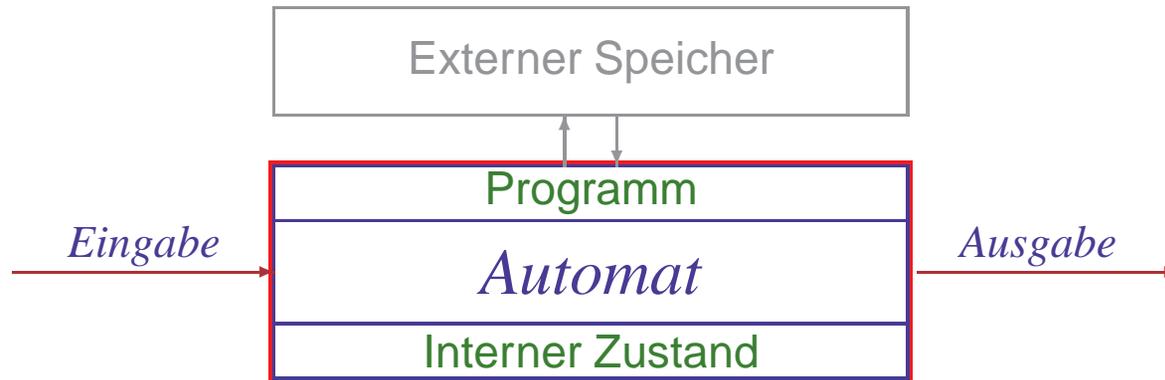
Aus der Sicht des Berechnungsmoduls



Sichtweisen von Computern

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

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

Basismodell 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 ...

- **Generierte Sprache**

- Menge aller **möglichen** Ausgaben des Automaten

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

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**

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

- **Automaten: erkennen von Wörtern**

- z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben

aus

ein

- **Zustände:** aus, ein

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

- **Automaten: erkennen von Wörtern**

- z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben

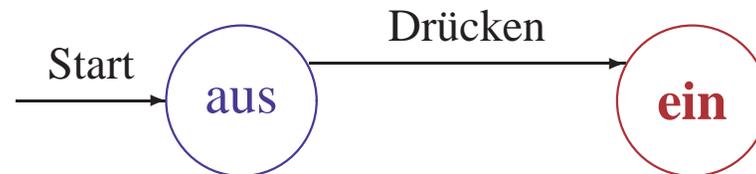


- **Zustände:** aus, ein – **Startzustand:** aus

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

- **Automaten: erkennen von Wörtern**

- z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben



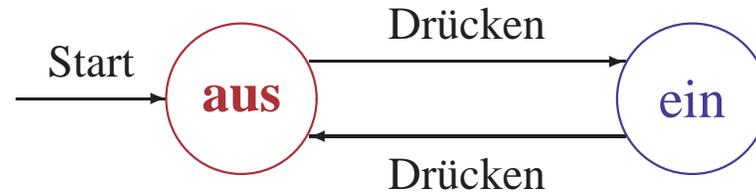
- **Zustände:** aus, ein – **Startzustand:** aus

- **Eingabesymbol:** Drücken

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

- **Automaten: erkennen von Wörtern**

- z.B. Wechselschalter: Verarbeitung von “Drücken”-Eingaben



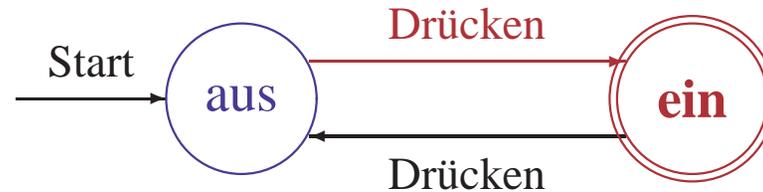
- **Zustände:** aus, ein – **Startzustand:** aus

- **Eingabesymbol:** Drücken

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

● **Automaten: erkennen von Wörtern**

– z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben



– **Zustände:** aus, ein – **Startzustand:** aus – **Endzustand:** ein

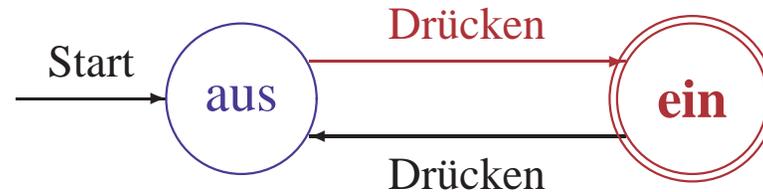
– **Eingabesymbol:** Drücken

– Endzustand wird erreicht bei ungerader Anzahl von Drücken

MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

● **Automaten: erkennen von Wörtern**

- z.B. Wechselschalter: Verarbeitung von “Drücken”-Eingaben



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

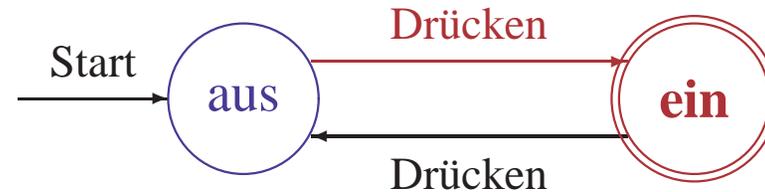
● **Mathematische Mengennotation**

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

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

- Endzustand wird erreicht bei ungerader Anzahl von Drücken

- **Mathematische Mengennotation**

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

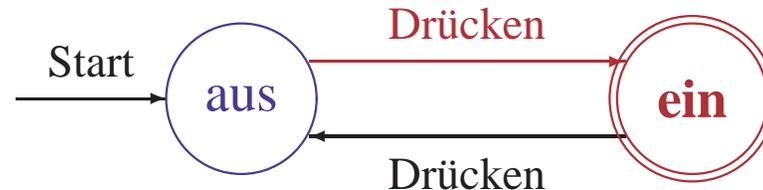
- **Reguläre Ausdrücke: algebraische Strukturen**

- z.B.: $(\text{DrückenDrücken})^* \text{Drücken}$

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

- Endzustand wird erreicht bei ungerader Anzahl von Drücken

- **Mathematische Mengennotation**

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

- **Reguläre Ausdrücke: algebraische Strukturen**

- z.B.: $(\text{DrückenDrücken})^* \text{Drücken}$

- **Grammatiken: Vorschriften für Spracherzeugung**

- z.B.: $S \rightarrow \text{Drücken}$, $S \rightarrow S\text{DrückenDrücken}$

- Erzeugt nur ungerade Anzahl von Drücken-Symbolen

Theoretische Informatik I

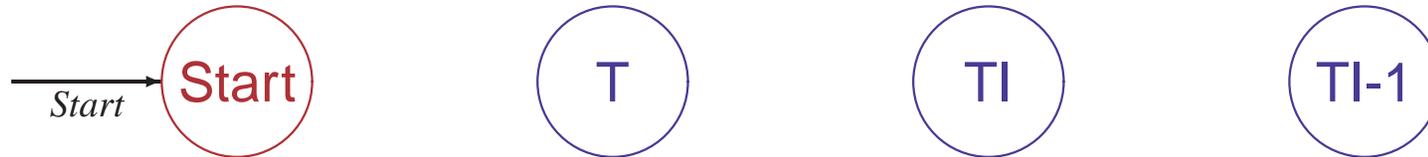
Einheit 2.1

Deterministische Endliche Automaten



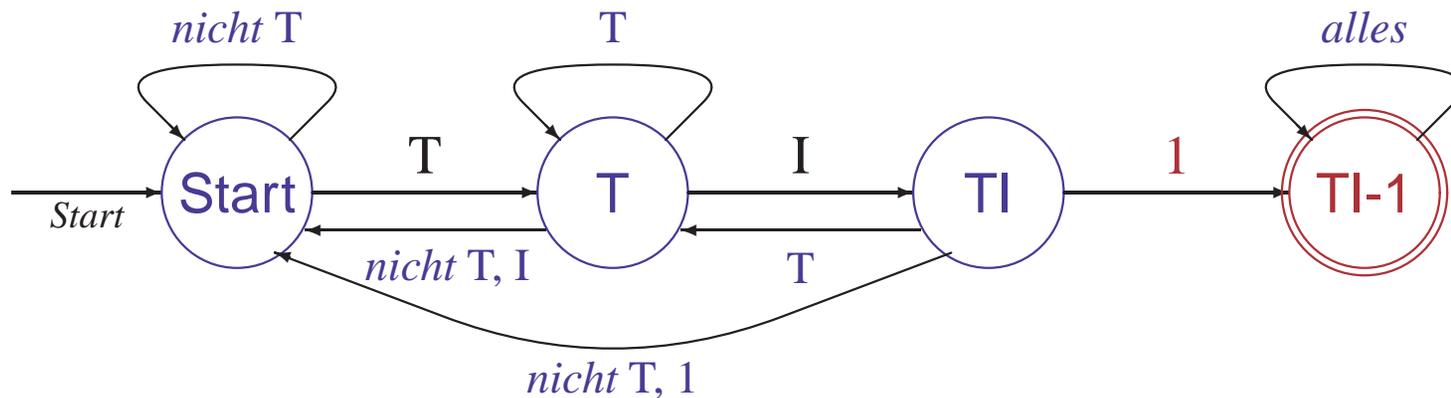
1. Arbeitsweise
2. Akzeptierte Sprache
3. Entwurf und Analyse
4. Automaten mit Ausgabe

ERKENNUNG VON WÖRTERN MIT AUTOMATEN



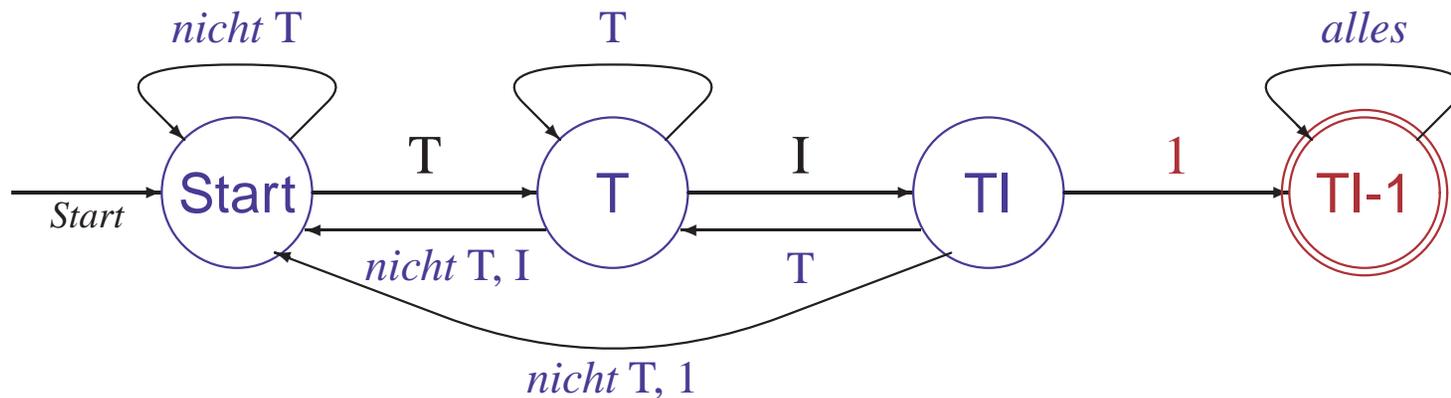
- Endliche Anzahl von **Zuständen**
- Ein **Startzustand**

ERKENNUNG VON WÖRTERN MIT AUTOMATEN



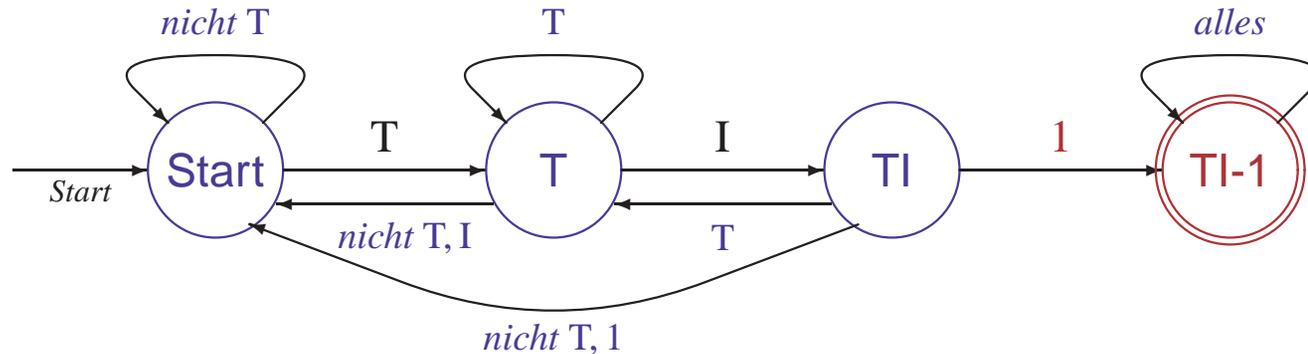
- Endliche Anzahl von **Zuständen**
- Ein **Startzustand**
- Regeln für **Zustandsübergänge**
- **Eingabealphabet:** $\{A, \dots, Z, a, \dots, z, \text{ }, ?, !, \dots\}$

ERKENNUNG VON WÖRTERN MIT AUTOMATEN



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

ENDLICHE AUTOMATEN – MATHEMATISCH PRÄZISIERT



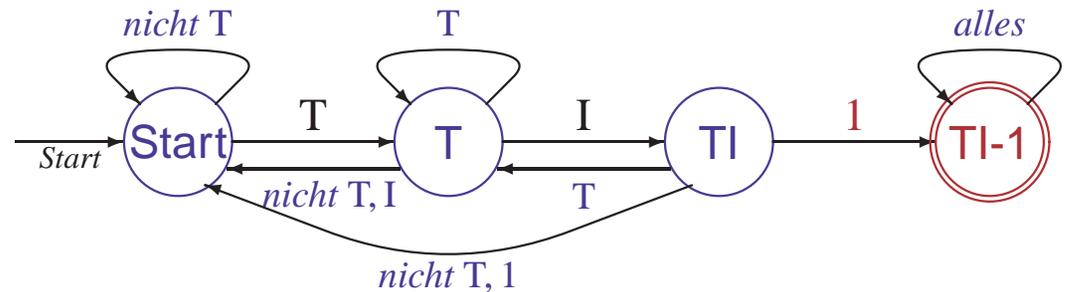
Ein **Deterministischer Endlicher Automat (DEA)**

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

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

BESCHREIBUNG VON ENDLICHEN AUTOMATEN

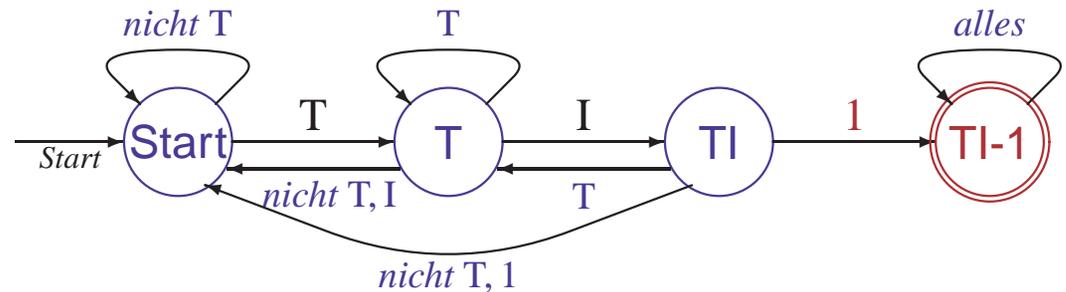
• Übergangsdiagramm



- 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

BESCHREIBUNG VON ENDLICHEN AUTOMATEN

• Übergangsdiagramm



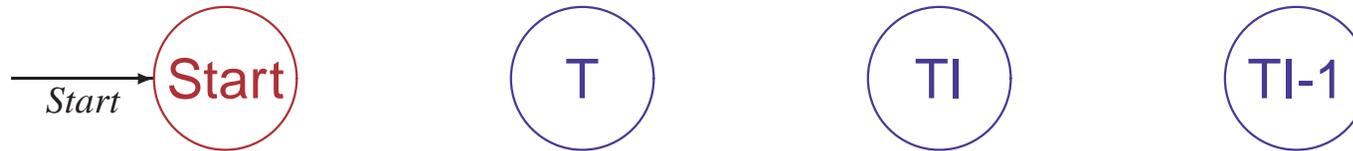
- Jeder Zustand in Q wird durch einen **Knoten** (Kreise) dargestellt
- Ist $\delta(q, a) = p$, so verläuft eine **Kante** von q nach p mit **Beschriftung** a (mehrere Beschriftungen derselben Kante möglich)
- q_0 wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände in F werden durch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

		<i>T</i>	<i>I</i>	<i>1</i>	<i>sonst</i>
\rightarrow	<i>S</i>	<i>T</i>	<i>S</i>	<i>S</i>	<i>S</i>
	<i>T</i>	<i>T</i>	<i>I</i>	<i>S</i>	<i>S</i>
	<i>I</i>	<i>T</i>	<i>S</i>	<i>1</i>	<i>S</i>
<i>*</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

- Automat befindet sich im Startzustand q_0

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

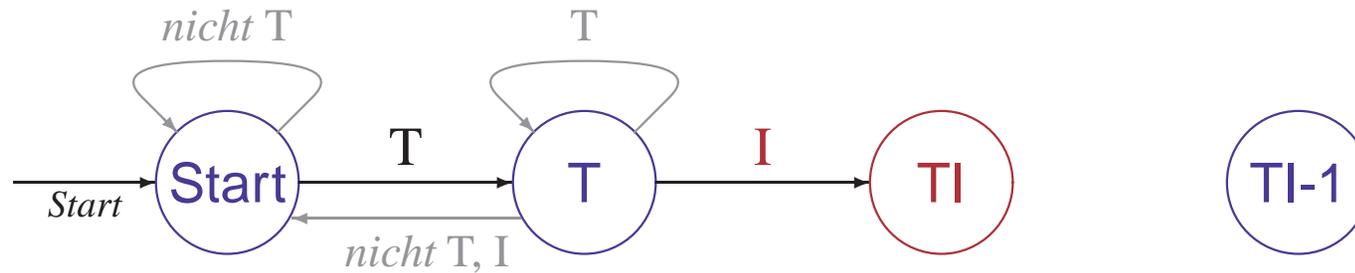
- Automat befindet sich im Startzustand q_0

- **Arbeitsschritt**

- Im Zustand q lese Eingabesymbol a ,

- Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

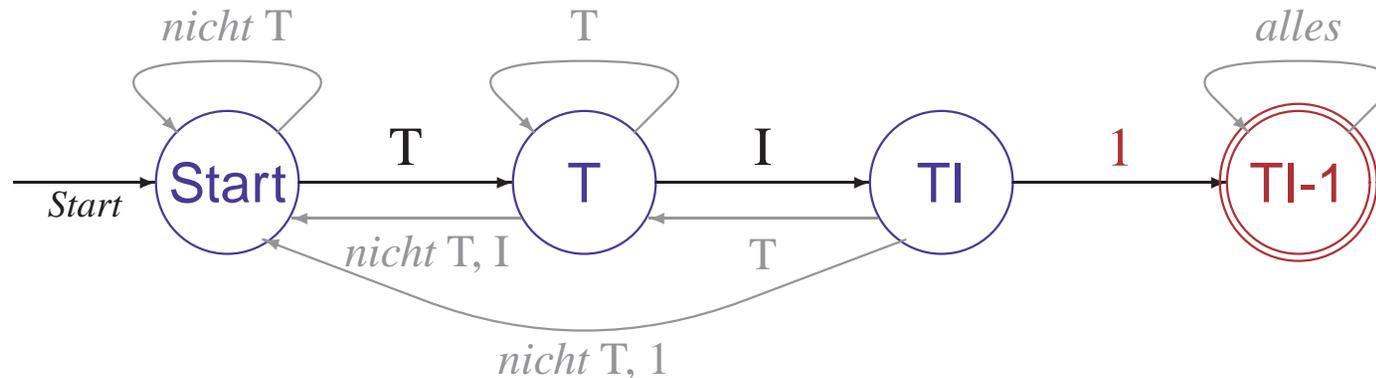
- Automat befindet sich im Startzustand q_0

- **Arbeitsschritt**

- Im Zustand q lese Eingabesymbol a ,

- Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

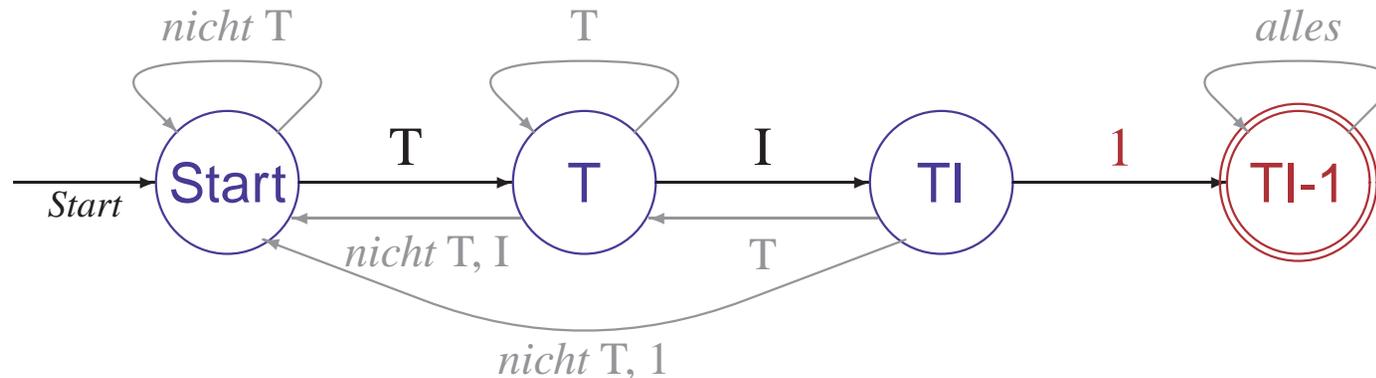
- Automat befindet sich im Startzustand q_0

- **Arbeitsschritt**

- Im Zustand q lese Eingabesymbol a ,

- Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

- Automat befindet sich im Startzustand q_0

- **Arbeitsschritt**

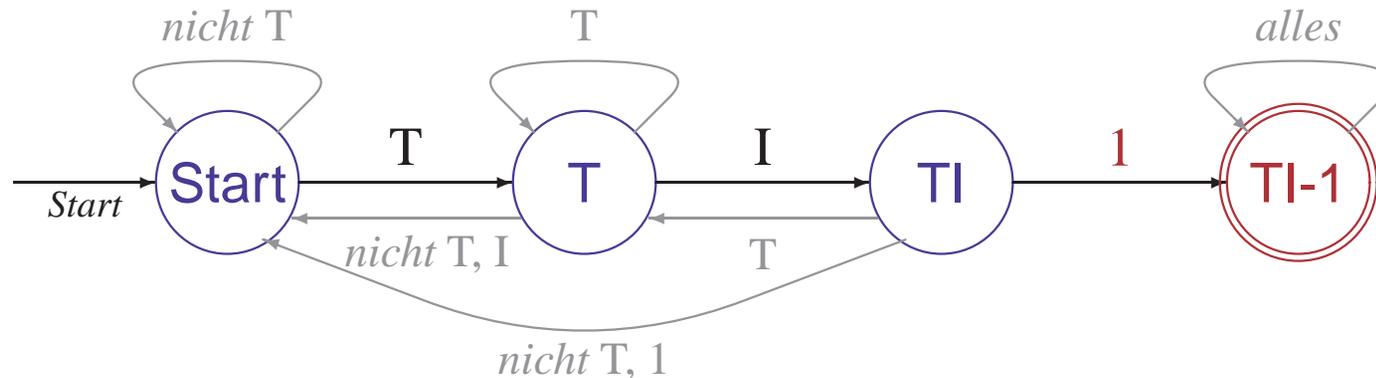
- Im Zustand q lese Eingabesymbol a ,

- 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

ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

- Automat befindet sich im Startzustand q_0

- **Arbeitsschritt**

- 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 Überföhrungsfunktion $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$**
 - Schrittweise Abarbeitung der Eingabe mit δ von links nach rechts
 - Informal: $\hat{\delta}(q, w_1w_2\dots w_n) = \delta(\dots(\delta(\delta(q, w_1), w_2), \dots), w_n)$

- **Erweiterte Überföhrungsfunktion $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$**
 - Schrittweise Abarbeitung der Eingabe mit δ von links nach rechts
 - Informal: $\hat{\delta}(q, w_1w_2\dots w_n) = \delta(\dots(\delta(\delta(q, w_1), w_2), \dots), w_n)$
 - Mathematisch präzise Beschreibung benötigt **induktive Definition**

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

ARBEITSWEISE VON DEAs – MATHEMATISCH PRÄZISIERT

- **Erweiterte Überföhrungsfunktion $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$**

- Schrittweise Abarbeitung der Eingabe mit δ von links nach rechts

- Informal: $\hat{\delta}(q, w_1w_2\dots w_n) = \delta(\dots(\delta(\delta(q, w_1), w_2), \dots), w_n)$

- Mathematisch präzise Beschreibung benötigt **induktive Definition**

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

- **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**

- **Erweiterte Überföhrungsfunktion $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$**
 - Schrittweise Abarbeitung der Eingabe mit δ von links nach rechts
 - Informal: $\hat{\delta}(q, w_1w_2\dots w_n) = \delta(\dots(\delta(\delta(q, w_1), w_2), \dots), w_n)$
 - Mathematisch präzise Beschreibung benötigt **induktive Definition**

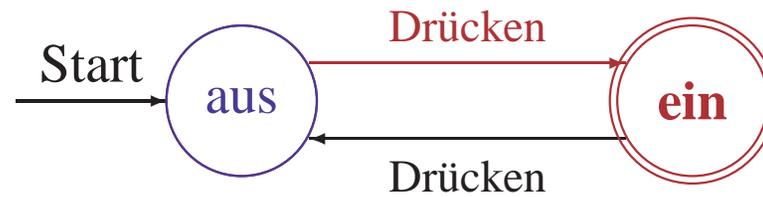
$$\hat{\delta}(q, w) = \begin{cases} q & \text{falls } w = \epsilon, \\ \delta(\hat{\delta}(q, v), a) & \text{falls } w = va \text{ für ein } v \in \Sigma^*, a \in \Sigma \end{cases}$$

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

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

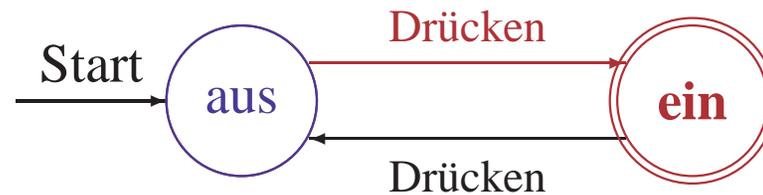
- Auch: die **von A erkannte Sprache**
- **Reguläre Sprache**
 - Sprache, die von einem DEA A akzeptiert wird

ANALYSE DER SPRACHE DES WECHSELSCHALTERS



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

ANALYSE DER SPRACHE DES WECHSELSCHALTERS



- **Sprache: Eingaben, für die Automat eingeschaltet ist**

- Teilmenge der Wörter über dem Alphabet $\Sigma = \{\text{Drücken}\}$

- **Automat A ist ein Wechselschalter**

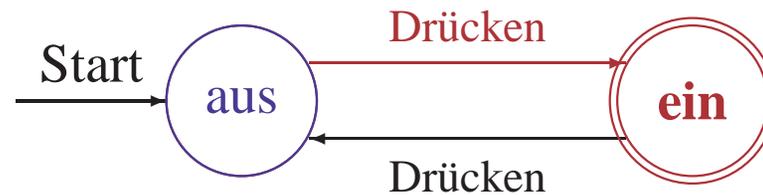
Zwei Eigenschaften sind zu zeigen:

- $S_1(n)$: Ist n gerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$

- $S_2(n)$: Ist n ungerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$

Beweis durch simultane Induktion (vgl. Einheit 1, Folie 10)

ANALYSE DER SPRACHE DES WECHSELSCHALTERS



- **Sprache: Eingaben, für die Automat eingeschaltet ist**

- Teilmenge der Wörter über dem Alphabet $\Sigma = \{\text{Drücken}\}$

- **Automat A ist ein Wechselschalter**

Zwei Eigenschaften sind zu zeigen:

- $S_1(n)$: Ist n gerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$

- $S_2(n)$: Ist n ungerade, so ist $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$

Beweis durch simultane Induktion (vgl. Einheit 1, Folie 10)

- **Formale Beschreibung der Sprache von A**

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

ENTWURF UND ANALYSE ENDLICHER AUTOMATEN

Entwerfe Automaten für $L = \{u01v \mid u, v \in \{0, 1\}^*\}$

Entwerfe Automaten für $L = \{u01v \mid u, v \in \{0, 1\}^*\}$

- **Drei Zustände sind erforderlich**

- Zustand q_0 : A hat noch keine 0 gelesen 1^i bleibt in q_0
- Zustand q_1 : A hat eine 0 aber noch keine 1 gelesen $1^i 0^{j+1}$ bleibt in q_1
- Zustand q_2 : A hat eine Zeichenkette 01 gelesen $1^i 0^j 01 v$ bleibt in q_2

Entwerfe Automaten für $L = \{u01v \mid u, v \in \{0, 1\}^*\}$

- **Drei Zustände sind erforderlich**

- Zustand q_0 : A hat noch keine 0 gelesen 1^i bleibt in q_0
- Zustand q_1 : A hat eine 0 aber noch keine 1 gelesen $1^i 0^{j+1}$ bleibt in q_1
- Zustand q_2 : A hat eine Zeichenkette 01 gelesen $1^i 0^j 01v$ bleibt in q_2

- **Zustandsübergänge erhalten “Bedeutung”**

- Zustand q_0 : Mit 1 bleibe in q_0 , sonst wechsle nach q_1
- Zustand q_1 : Mit 0 bleibe in q_1 , sonst wechsle nach q_2
- Zustand q_2 : Bleibe bei jeder Eingabe in q_2 , Endzustand

Entwerfe Automaten für $L = \{u01v \mid u, v \in \{0, 1\}^*\}$

- **Drei Zustände sind erforderlich**

- Zustand q_0 : A hat noch keine 0 gelesen

1^i bleibt in q_0

- Zustand q_1 : A hat eine 0 aber noch keine 1 gelesen

$1^i 0^{j+1}$ bleibt in q_1

- Zustand q_2 : A hat eine Zeichenkette 01 gelesen

$1^i 0^j 01 v$ bleibt in q_2

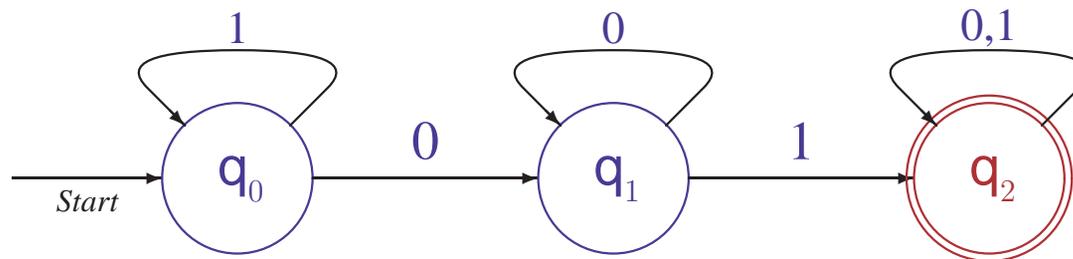
- **Zustandsübergänge erhalten “Bedeutung”**

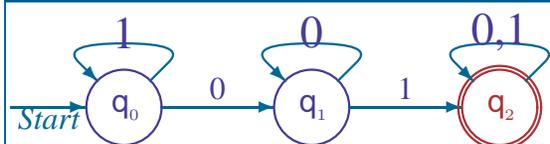
- Zustand q_0 : Mit 1 bleibe in q_0 , sonst wechsele nach q_1

- Zustand q_1 : Mit 0 bleibe in q_1 , sonst wechsele nach q_2

- Zustand q_2 : Bleibe bei jeder Eingabe in q_2 , Endzustand

- **Zugehöriger DEA mit Alphabet $\Sigma = \{0, 1\}$**

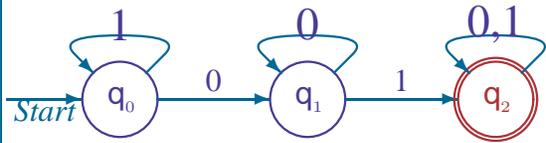




ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• **Zeige durch strukturelle Induktion über w :**

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ **es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$**

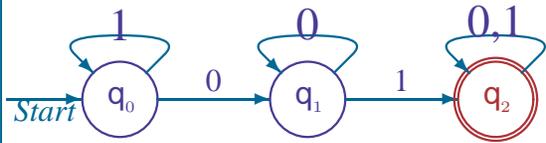


ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• Zeige durch strukturelle Induktion über w :

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$

Basisfall $w = \epsilon$: Per Definition ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $w = 1^i$ für $i = 0$ ✓



ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• Zeige durch strukturelle Induktion über w :

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$

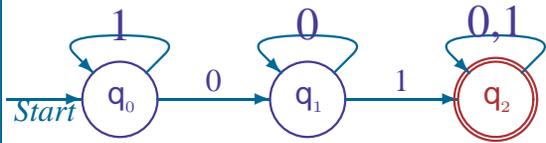
Basisfall $w = \epsilon$: Per Definition ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $w = 1^i$ für $i = 0$ ✓

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

• Es gelte $\hat{\delta}(q_0, w) = q_0$. Dann ist $\hat{\delta}(q_0, u) = q_0$ und $\delta(q_0, a) = q_0$.

Es folgt $a = 1$ und per Annahme $u = 1^i$ für ein i , also $w = 1^{i+1}$. ✓

• Es gelte $w = 1^i$. Dann ist $a = 1$ und $u = 1^{i-1}$. Mit der Induktionsannahme folgt $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, u), a) = \delta(q_0, a) = q_0$ ✓



ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• Zeige durch strukturelle Induktion über w :

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$

Basisfall $w = \epsilon$: Per Definition ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $w = 1^i$ für $i = 0$ ✓

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

• Es gelte $\hat{\delta}(q_0, w) = q_0$. Dann ist $\hat{\delta}(q_0, u) = q_0$ und $\delta(q_0, a) = q_0$.

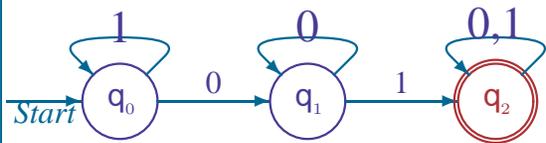
Es folgt $a = 1$ und per Annahme $u = 1^i$ für ein i , also $w = 1^{i+1}$. ✓

• Es gelte $w = 1^i$. Dann ist $a = 1$ und $u = 1^{i-1}$. Mit der Induktionsannahme folgt $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, u), a) = \delta(q_0, a) = q_0$ ✓

– $\hat{\delta}(q_0, w) = q_1 \Leftrightarrow$ es gibt $i, j \in \mathbb{N}$ mit $w = 1^i 0^{j+1}$

analog

– $\hat{\delta}(q_0, w) = q_2 \Leftrightarrow$ es gibt $i, j \in \mathbb{N}, v \in \Sigma^*$ mit $w = 1^i 0^{j+1} 1 v$



ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• **Zeige durch strukturelle Induktion über w :**

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ **es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$**

Basisfall $w = \epsilon$: Per Definition ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $w = 1^i$ für $i = 0$ ✓

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

• Es gelte $\hat{\delta}(q_0, w) = q_0$. Dann ist $\hat{\delta}(q_0, u) = q_0$ und $\delta(q_0, a) = q_0$.

Es folgt $a = 1$ und per Annahme $u = 1^i$ für ein i , also $w = 1^{i+1}$. ✓

• Es gelte $w = 1^i$. Dann ist $a = 1$ und $u = 1^{i-1}$. Mit der Induktionsannahme folgt $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, u), a) = \delta(q_0, a) = q_0$ ✓

– $\hat{\delta}(q_0, w) = q_1 \Leftrightarrow$ **es gibt $i, j \in \mathbb{N}$ mit $w = 1^i 0^{j+1}$**

analog

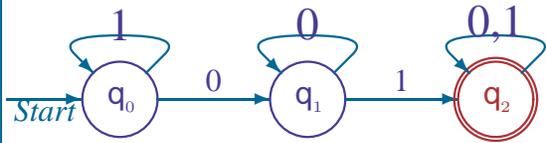
– $\hat{\delta}(q_0, w) = q_2 \Leftrightarrow$ **es gibt $i, j \in \mathbb{N}, v \in \Sigma^*$ mit $w = 1^i 0^{j+1} 1v$**

• **Zeige:** $w \in L \Leftrightarrow$ **es gibt $i, j \in \mathbb{N}, v \in \Sigma^*$ mit $w = 1^i 0^j 01v$**

\Rightarrow Für $w \in L$ gibt es $u, v \in \Sigma^*$ mit $w = u01v$

Wenn u nicht die Form $1^i 0^j$ hat, dann folgt in u eine 1 auf eine 0.

Das erste solche Vorkommen von 01 liefert die gewünschte Zerlegung ✓



ZEIGE $L(A) = L = \{u01v \mid u, v \in \{0, 1\}^*\}$

• **Zeige durch strukturelle Induktion über w :**

– $\hat{\delta}(q_0, w) = q_0 \Leftrightarrow$ es gibt ein $i \in \mathbb{N}$ mit $w = 1^i$

Basisfall $w = \epsilon$: Per Definition ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $w = 1^i$ für $i = 0$ ✓

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

• Es gelte $\hat{\delta}(q_0, w) = q_0$. Dann ist $\hat{\delta}(q_0, u) = q_0$ und $\delta(q_0, a) = q_0$.

Es folgt $a = 1$ und per Annahme $u = 1^i$ für ein i , also $w = 1^{i+1}$. ✓

• Es gelte $w = 1^i$. Dann ist $a = 1$ und $u = 1^{i-1}$. Mit der Induktionsannahme folgt $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, u), a) = \delta(q_0, a) = q_0$ ✓

– $\hat{\delta}(q_0, w) = q_1 \Leftrightarrow$ es gibt $i, j \in \mathbb{N}$ mit $w = 1^i 0^j + 1$

analog

– $\hat{\delta}(q_0, w) = q_2 \Leftrightarrow$ es gibt $i, j \in \mathbb{N}, v \in \Sigma^*$ mit $w = 1^i 0^j + 1v$

• **Zeige: $w \in L \Leftrightarrow$ es gibt $i, j \in \mathbb{N}, v \in \Sigma^*$ mit $w = 1^i 0^j 01v$**

\Rightarrow Für $w \in L$ gibt es $u, v \in \Sigma^*$ mit $w = u01v$

Wenn u nicht die Form $1^i 0^j$ hat, dann folgt in u eine 1 auf eine 0.

Das erste solche Vorkommen von 01 liefert die gewünschte Zerlegung ✓

• **Es folgt $w \in L \Leftrightarrow \hat{\delta}(q_0, w) = q_2 \in F \Leftrightarrow w \in L(A)$**

- **Konfiguration:** ‘Gesamtzustand’ von Automaten
 - Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
 - Formal dargestellt als Tupel $K = (q, w) \in Q \times \Sigma^*$

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

- **Konfiguration:** ‘Gesamtzustand’ von Automaten
 - Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
 - Formal dargestellt als Tupel $K = (q, w) \in Q \times \Sigma^*$
- **Konfigurationsübergangsrelation** \vdash^*
 - Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
 - $(q, aw) \vdash (p, w)$, falls $\delta(q, a) = p$
 - $K_1 \vdash^* K_2$, 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
$$L(A) = \{w \in \Sigma^* \mid \exists p \in F. (q_0, w) \vdash^* (p, \epsilon)\}$$

- **Konfiguration:** ‘Gesamtzustand’ von Automaten
 - Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
 - Formal dargestellt als Tupel $K = (q, w) \in Q \times \Sigma^*$
- **Konfigurationsübergangsrelation** \vdash^*
 - Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
 - $(q, aw) \vdash (p, w)$, falls $\delta(q, a) = p$
 - $K_1 \vdash^* K_2$, 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
$$L(A) = \{w \in \Sigma^* \mid \exists p \in F. (q_0, w) \vdash^* (p, \epsilon)\}$$

Für DEAs weniger intuitiv, aber leichter zu verallgemeinern

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

Codiere Anzahl der gelesener 0/1 im Zustand

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

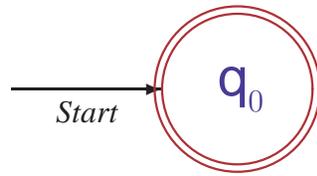
$q_2 \hat{=} (\text{ungerade, gerade})$ $q_3 \hat{=} (\text{ungerade, ungerade})$

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

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

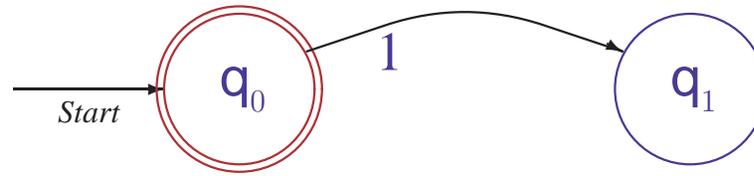


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

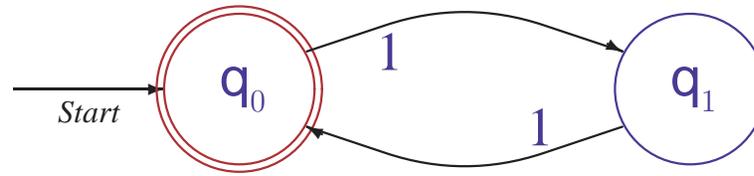


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

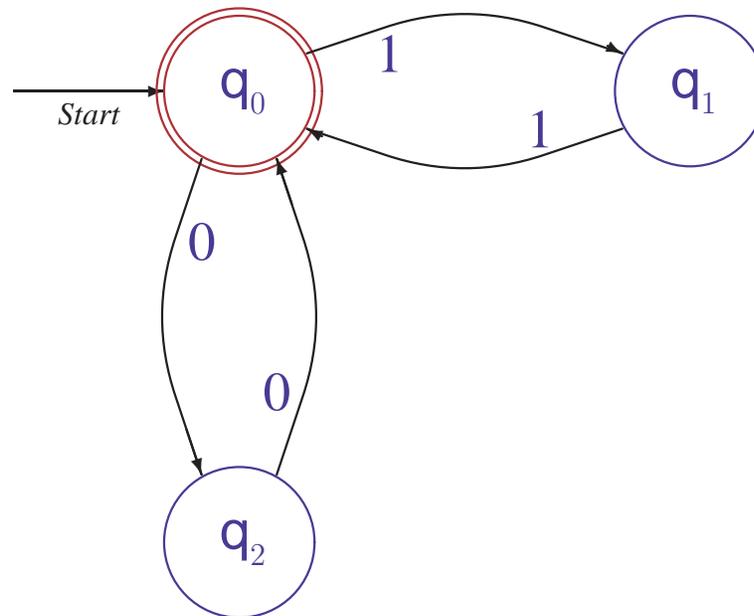


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

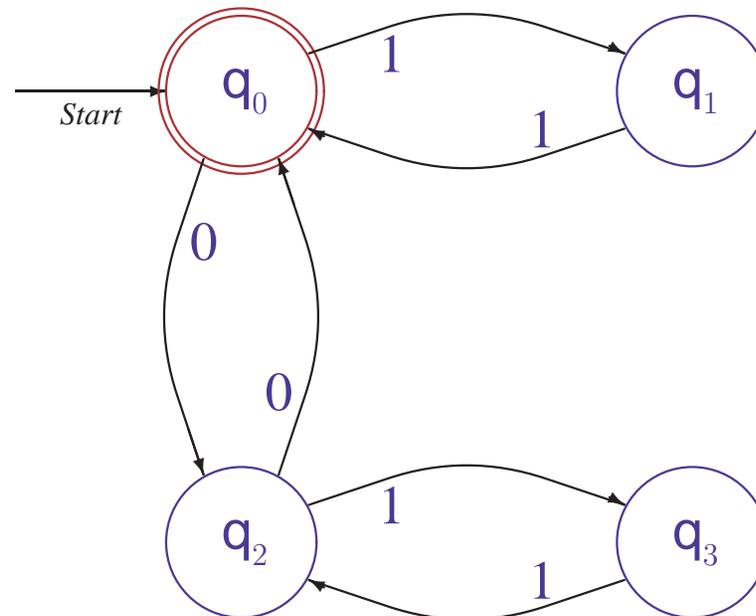


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

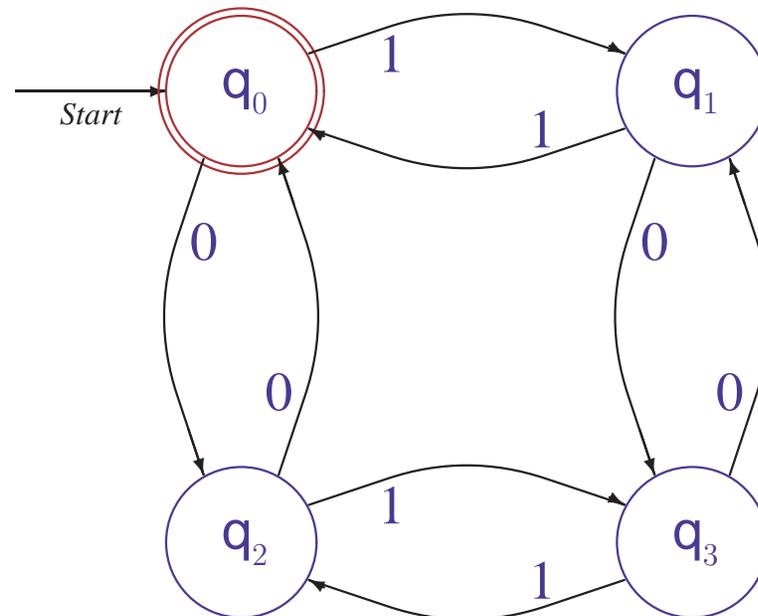


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codiere Anzahl der gelesener 0/1 im Zustand

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

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

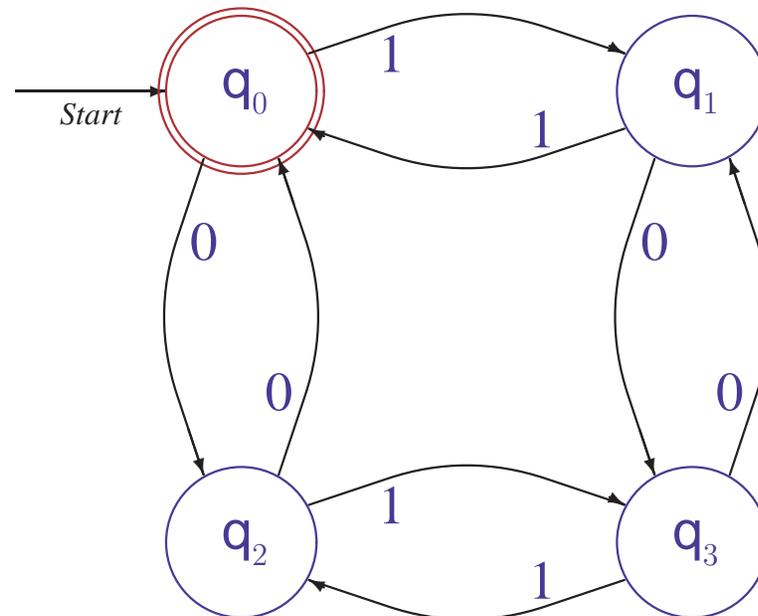


DEA FÜR $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

Codierte Anzahl der gelesener 0/1 im Zustand

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

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



Korrektheit: gegenseitige strukturelle Induktion

KORREKTHEITSBEWWEIS MIT KONFIGURATIONEN

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

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

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

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

(4) $(q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$ 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, ...

KORREKTHEITSBEWWEIS MIT KONFIGURATIONEN

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

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

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

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

(4) $(q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$ 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 $g_1(w)$



KORREKTHEITSBEWEIF MIT KONFIGURATIONEN

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

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

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

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

(4) $(q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$ 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 $g_1(w)$

✓

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

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

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

KORREKTHEITSBEWWEIS MIT KONFIGURATIONEN

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

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

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

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

(4) $(q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$ 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 $g_1(w)$ ✓

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

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

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

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

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

Fall $a=1$ analog. Gegenrichtung durch Umkehrung des Arguments. (2), (3), (4) analog.

KORREKTHEITSBEWWEIS MIT KONFIGURATIONEN

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

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

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

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

(4) $(q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$ 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 $g_1(w)$ ✓

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

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

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

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

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

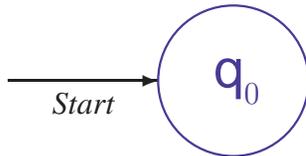
Fall $a=1$ analog. Gegenrichtung durch Umkehrung des Arguments. (2), (3), (4) analog.

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

$$\Leftrightarrow g_0(w) \text{ und } g_1(w) \Leftrightarrow w \in L$$

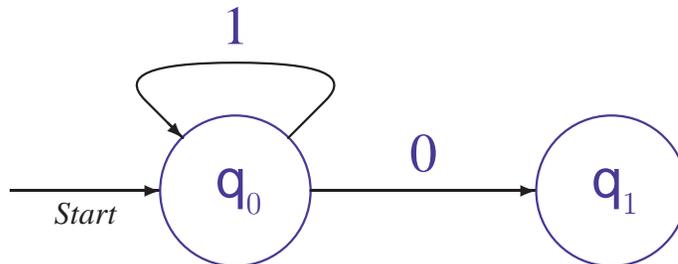
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



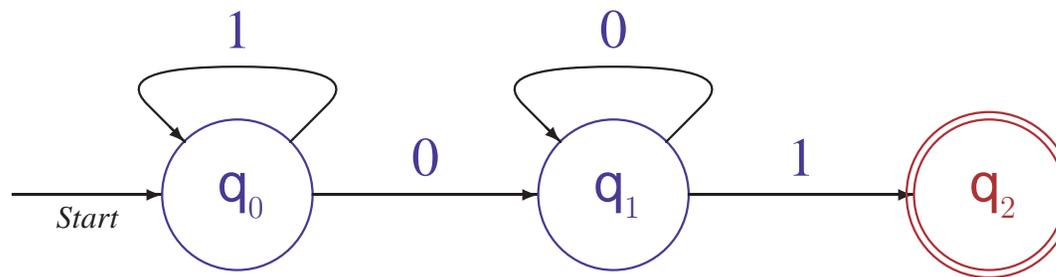
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



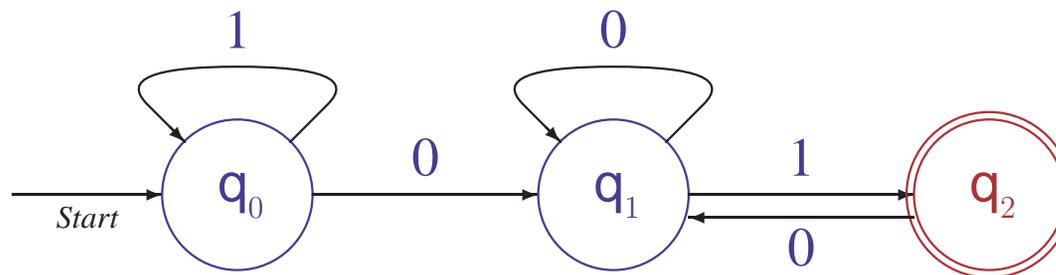
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



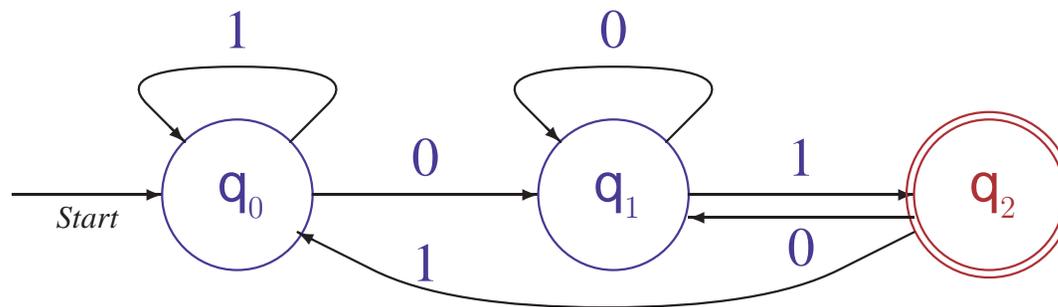
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



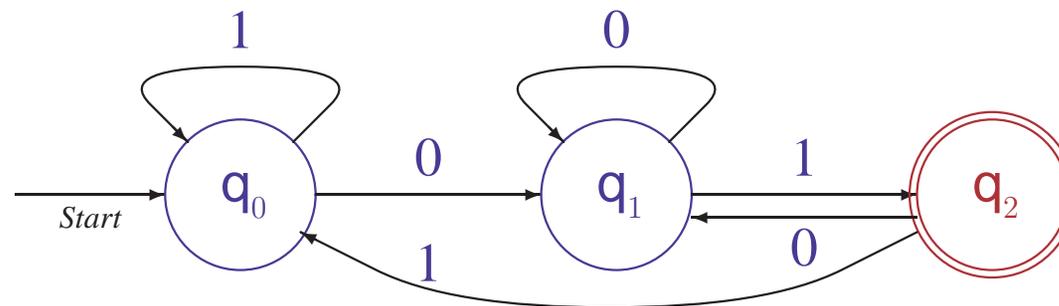
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



WEITERE BEISPIELE ENDLICHER AUTOMATEN

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

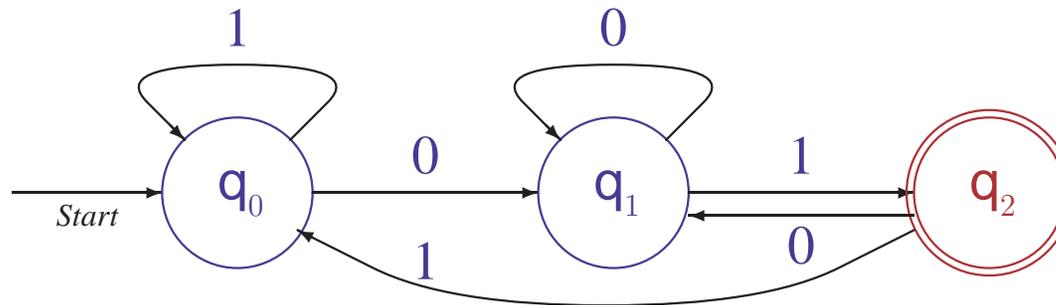


- **50c Kaffeeautomat**

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

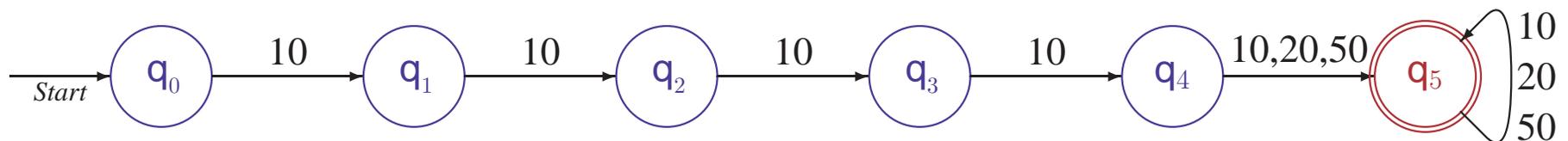
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



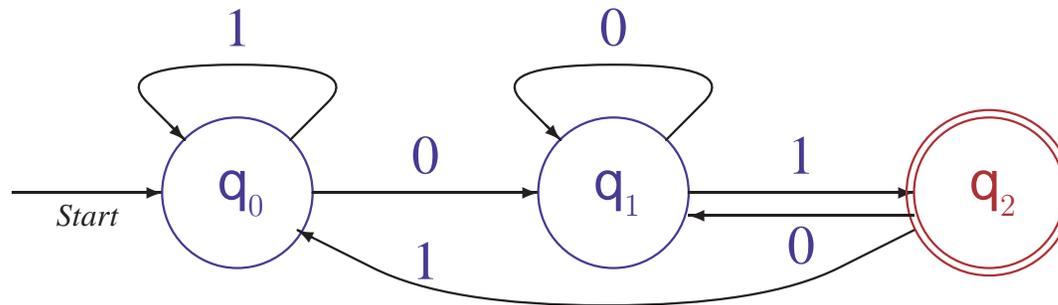
- **50c Kaffeeautomat**

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



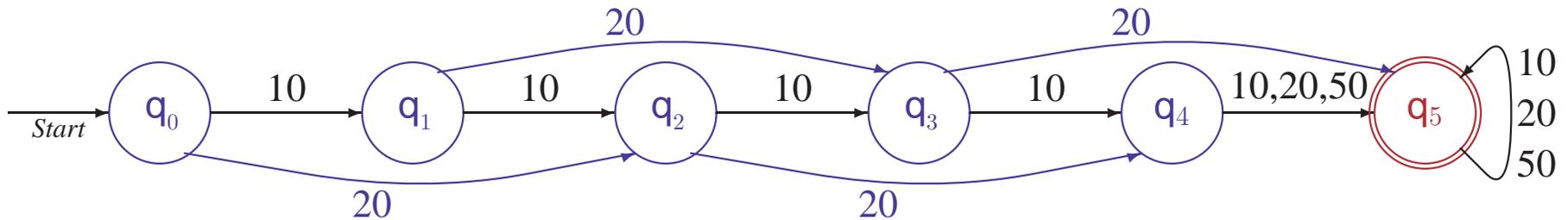
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



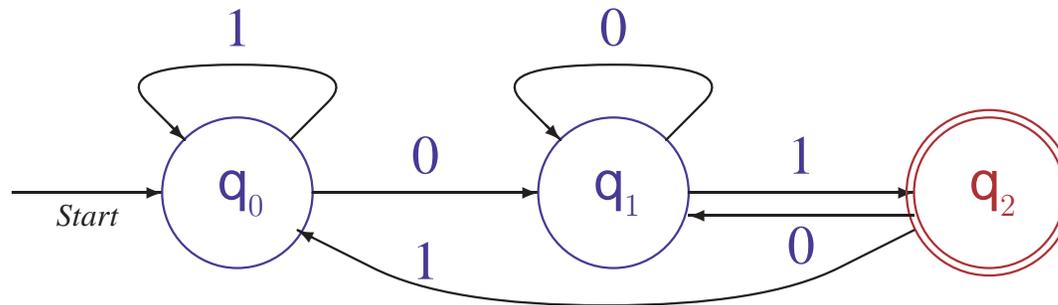
- **50c Kaffeeautomat**

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



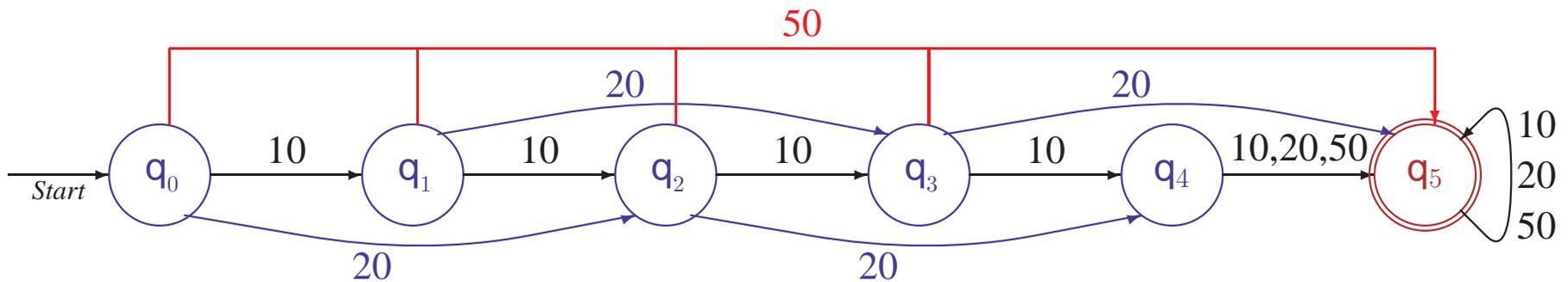
WEITERE BEISPIELE ENDLICHER AUTOMATEN

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



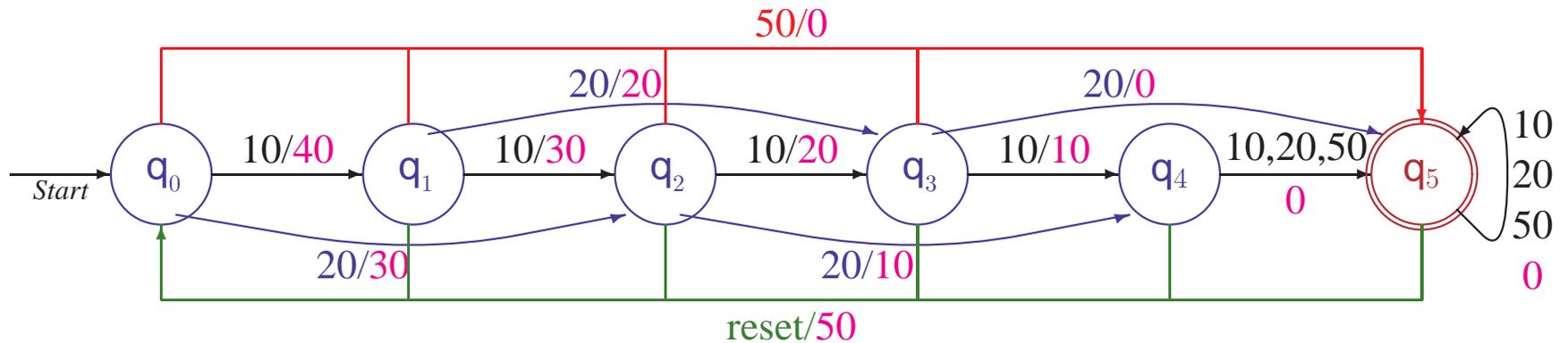
- **50c Kaffeeautomat**

– 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

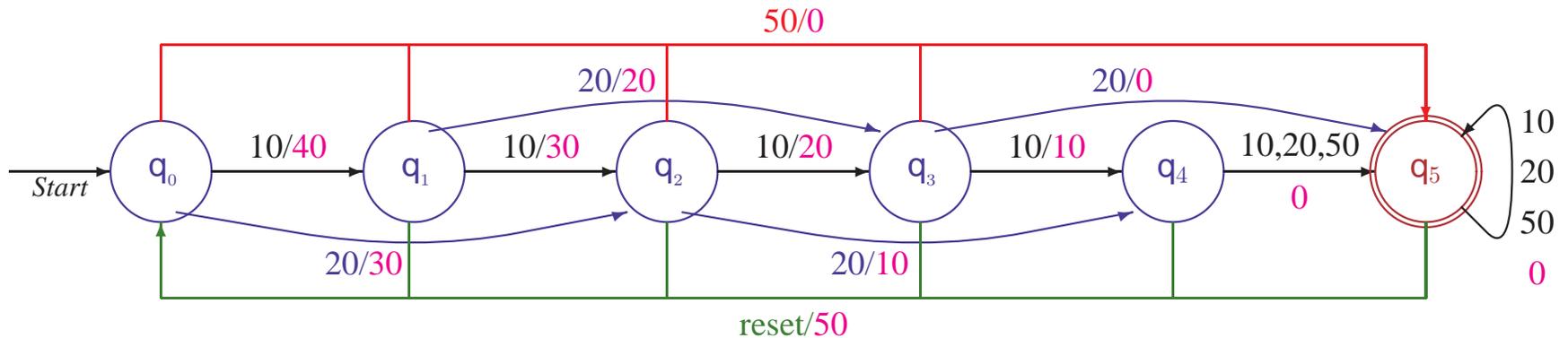
● Formalisierungen von Automaten mit Ausgabe

– **Mealy-Automaten**: Ausgabefunktion abhängig von Eingabe & Zustand

– **Moore-Automaten**: Ausgabefunktion nur von Zustand abhängig

Beide Modelle sind äquivalent

MEALY-AUTOMATEN – MATHEMATISCH PRÄZISIERT



Ein **Mealy-Automat** ist ein 6-Tupel

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ mit

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

- **Anfangssituation:** Automat im Startzustand q_0

- **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

- **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

- **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 $x_1..x_n$

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

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

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

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

- **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 $\Sigma = \{0, 1\} \times \{0, 1\}$

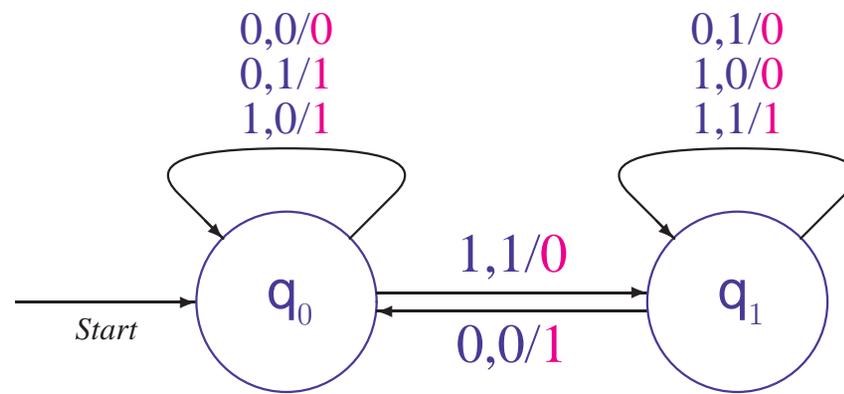
- Ausgabealphabet $\Delta = \{0, 1\}$

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

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**



MEALY-AUTOMATEN SIND ÄQUIVALENT ZU DEAS

Gegenseitige Simulation ist möglich

MEALY-AUTOMATEN SIND ÄQUIVALENT ZU DEAS

Gegenseitige Simulation ist möglich

- **Jede Funktion f ist als Menge beschreibbar**

- **graph**(f) = $\{(w, v) \mid f(w) = v\}$

- **graph***(f) = $\{(w_1, v_1)..(w_n, v_n) \mid f(w_1..w_n) = v_1..v_n\}$

- DEAs können Graphen berechneter Funktionen akzeptieren

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

MEALY-AUTOMATEN SIND ÄQUIVALENT ZU DEAS

Gegenseitige Simulation ist möglich

- **Jede Funktion f ist als Menge beschreibbar**

- **graph**(f) = $\{(w, v) \mid f(w) = v\}$
- **graph***(f) = $\{(w_1, v_1)..(w_n, v_n) \mid f(w_1..w_n) = v_1..v_n\}$
- DEAs können Graphen berechneter Funktionen akzeptieren

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

- **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 **“Mealy-berechenbar”**

BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **f Mealy-berechenbar $\Leftrightarrow \text{graph}^*(f)$ regulär**

– Zu $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ konstruiere $A = (Q \cup \{q_f\}, \Sigma \times \Delta, \delta', q_0, Q)$

$$\text{mit } \delta'(q, (a, b)) = \begin{cases} \delta(q, a) & \text{falls } \lambda(q, a) = b, \\ q_f & \text{sonst} \end{cases}$$

– Dann $f_M(w_1..w_n) = v_1..v_n$ genau dann, wenn $(w_1, v_1)..(w_n, v_n) \in L(A)$

BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **f Mealy-berechenbar $\Leftrightarrow \text{graph}^*(f)$ regulär**

- Zu $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ konstruiere $A = (Q \cup \{q_f\}, \Sigma \times \Delta, \delta', q_0, Q)$

- mit $\delta'(q, (a, b)) = \begin{cases} \delta(q, a) & \text{falls } \lambda(q, a) = b, \\ q_f & \text{sonst} \end{cases}$

- Dann $f_M(w_1..w_n) = v_1..v_n$ genau dann, wenn $(w_1, v_1)..(w_n, v_n) \in L(A)$

- **L regulär $\Leftrightarrow \chi_L$ “Mealy-berechenbar”**

- Zu $A = (Q, \Sigma, \delta, q_0, F)$ konstruiere $M = (Q, \Sigma, \{0,1\}, \delta, \lambda, q_0)$

- mit $\lambda(q, a) = \begin{cases} 1 & \text{falls } \delta(q, a) \in F, \\ 0 & \text{sonst} \end{cases}$

- 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 $f_M(w)$

BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **f Mealy-berechenbar $\Leftrightarrow \text{graph}^*(f)$ regulär**

- Zu $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ konstruiere $A = (Q \cup \{q_f\}, \Sigma \times \Delta, \delta', q_0, Q)$

- mit $\delta'(q, (a, b)) = \begin{cases} \delta(q, a) & \text{falls } \lambda(q, a) = b, \\ q_f & \text{sonst} \end{cases}$

- Dann $f_M(w_1..w_n) = v_1..v_n$ genau dann, wenn $(w_1, v_1)..(w_n, v_n) \in L(A)$

- **L regulär $\Leftrightarrow \chi_L$ “Mealy-berechenbar”**

- Zu $A = (Q, \Sigma, \delta, q_0, F)$ konstruiere $M = (Q, \Sigma, \{0,1\}, \delta, \lambda, q_0)$

- mit $\lambda(q, a) = \begin{cases} 1 & \text{falls } \delta(q, a) \in F, \\ 0 & \text{sonst} \end{cases}$

- 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 $f_M(w)$

Mehr zu Automaten mit Ausgabe im Buch von Vossen & Witt