

# Theoretische Informatik I

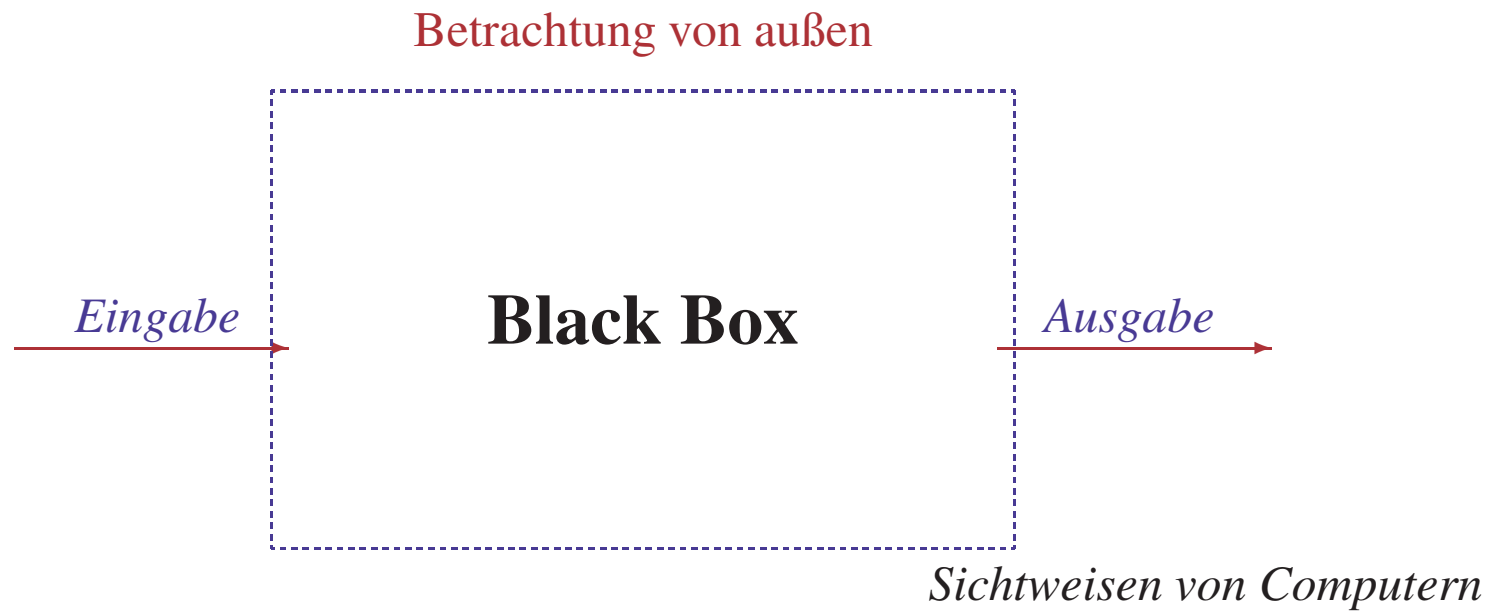
## Einheit 2

### Endliche Automaten & Reguläre Sprachen

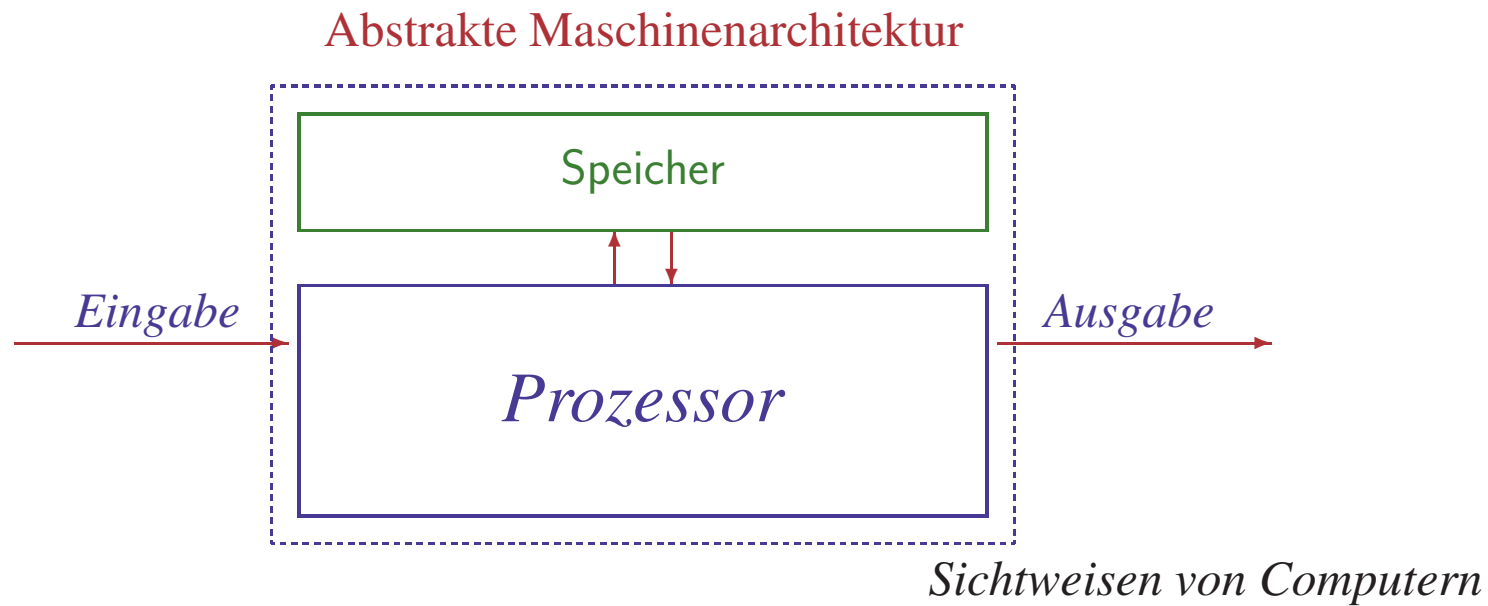


1. Deterministische endliche Automaten
2. Nichtdeterministische Automaten
3. Reguläre Ausdrücke
4. Grammatiken
5. Eigenschaften regulärer Sprachen

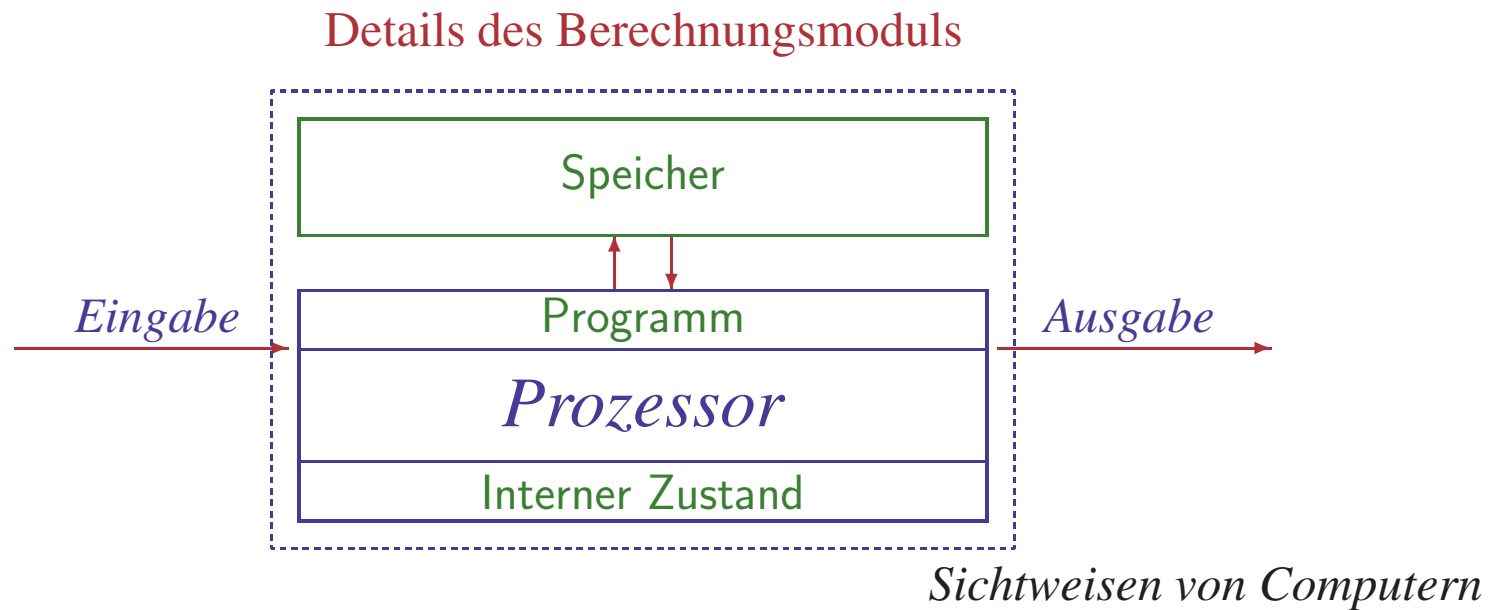
# AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL



# AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL

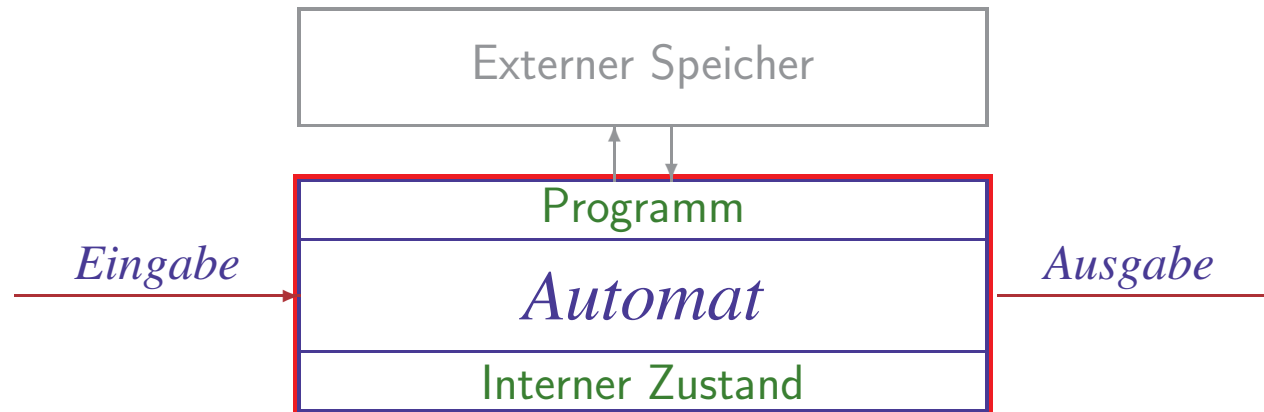


# AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL



# AUTOMATEN: DAS EINFACHSTE MASCHINENMODELL

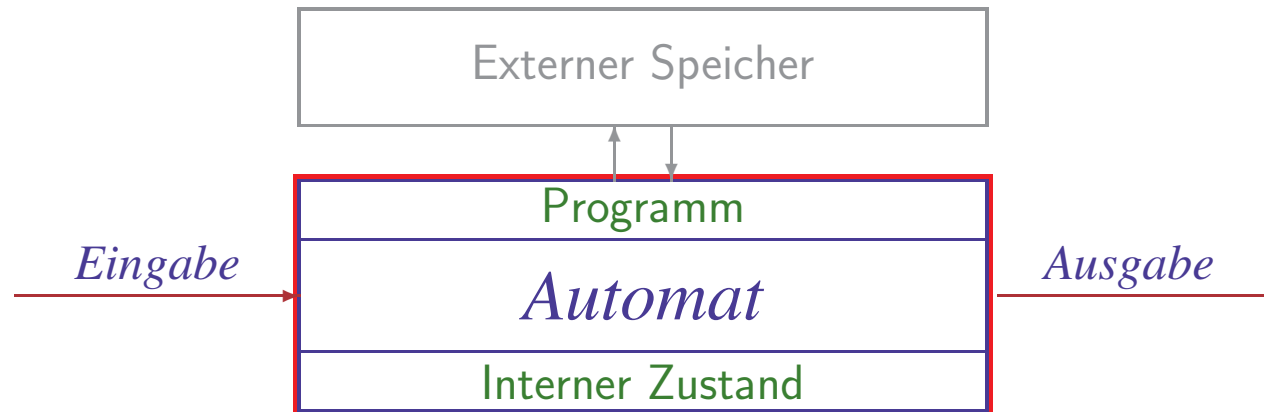
Aus der Sicht des Berechnungsmoduls



*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

## 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 ...

- **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

# MODELLE ZUR BESCHREIBUNG REGULÄRER SPRACHEN

- **Automaten: erkennen von Wörtern**

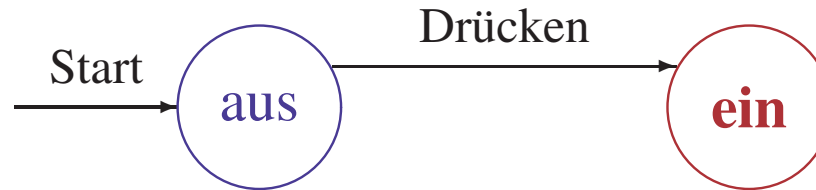
- z.B. Wechselschalter: Verarbeitung von “Drück”-Eingaben



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

- **Automaten: erkennen von Wörtern**

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

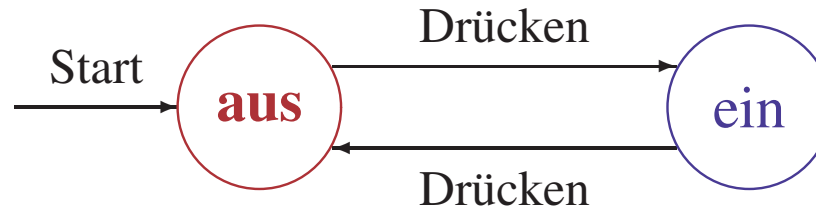


- **Zustände:** aus, ein – **Startzustand:** aus

- **Eingabesymbol:** Drücken eines Schalters

- **Automaten: erkennen von Wörtern**

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

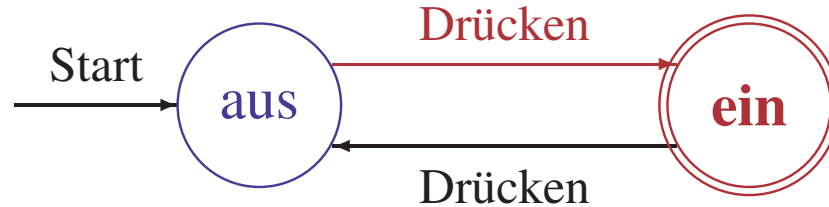


- **Zustände:** aus, ein – **Startzustand:** aus

- **Eingabesymbol:** Drücken eines Schalters

- **Automaten: erkennen von Wörtern**

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

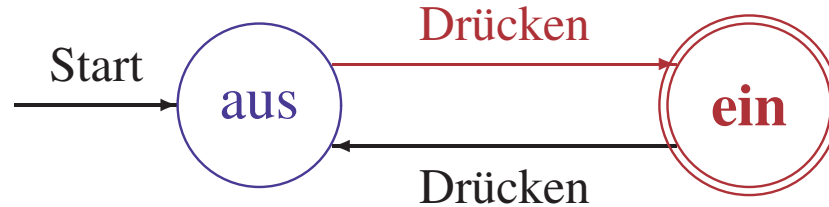


- **Zustände:** aus, ein – **Startzustand:** aus – **Endzustand:** ein
- **Eingabesymbol:** Drücken eines Schalters
- Endzustand wird erreicht bei ungerader Anzahl von Drücken

# 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 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  $\{\text{Drücken}^{2i+1} \mid i \in \mathbb{N}\}$

- **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

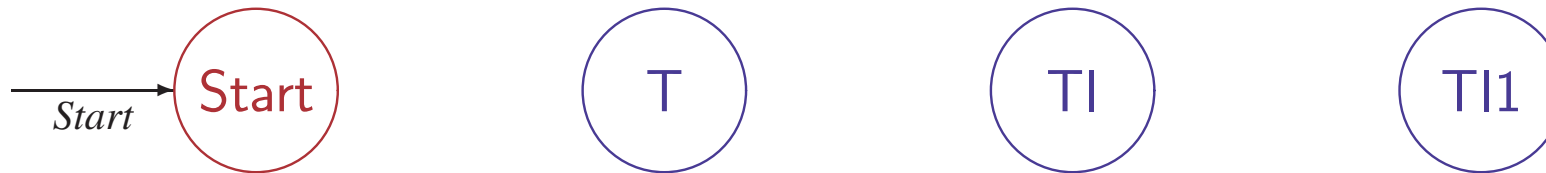
## Einheit 2.1

### Deterministische Endliche Automaten



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

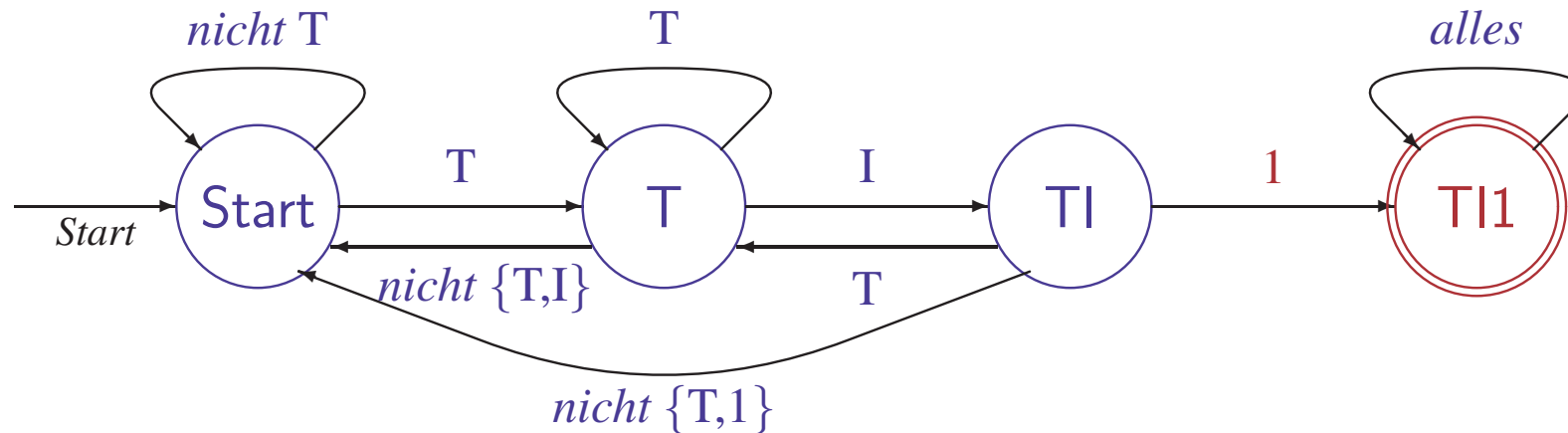
# ERKENNUNG VON WÖRTERN MIT AUTOMATEN



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

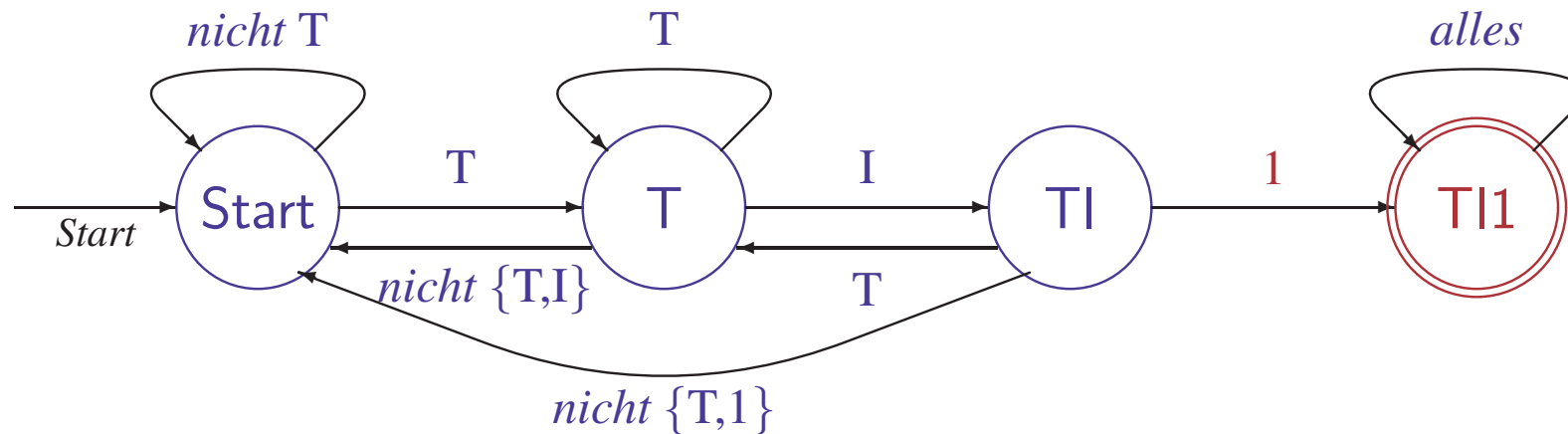


# ERKENNUNG VON WÖRTERN MIT AUTOMATEN



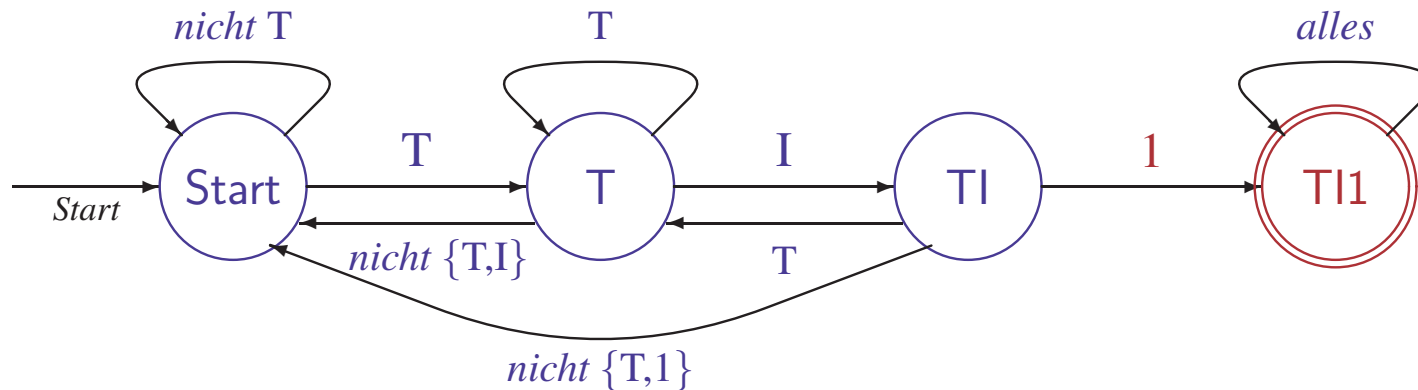
- Endliche Anzahl von **Zuständen**
- Ein **Startzustand**
- Regeln für **Zustandsübergänge**

# ERKENNUNG VON WÖRTERN MIT AUTOMATEN



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

# ENDLICHE AUTOMATEN – MATHEMATISCH PRÄZISIERT



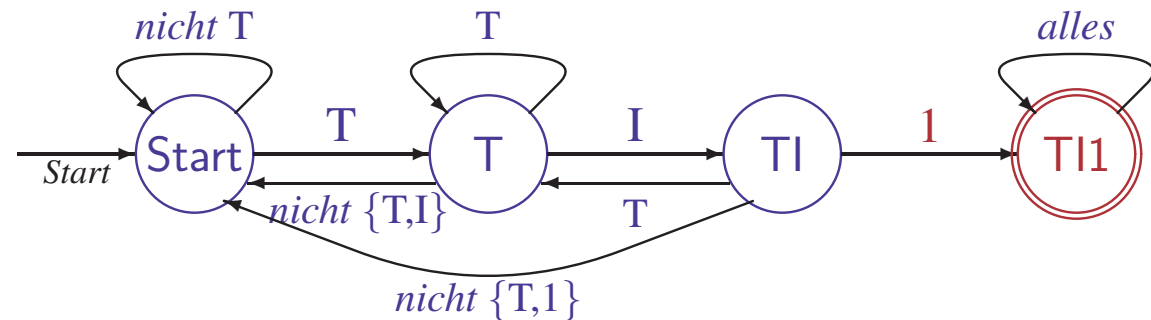
Ein **Deterministischer Endlicher Automat (DEA)**

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

- $Q$  nichtleere endliche **Zustandsmenge**
- $\Sigma$  (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



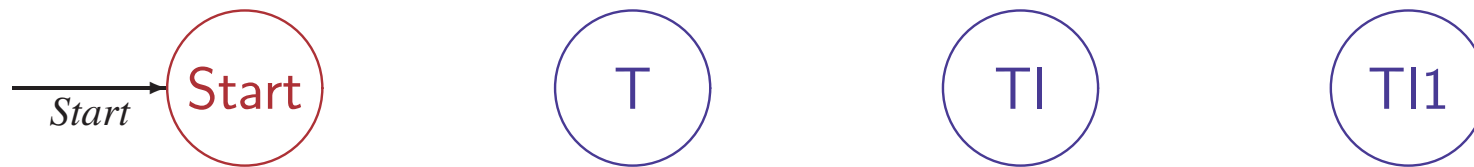
- Jeder Zustand in  $Q$  wird durch einen **Knoten** (Kreise) dargestellt
- Ist  $\delta(q, a) = p$ , so verläuft eine **Kante** von  $q$  nach  $p$  mit **Beschriftung**  $a$  (mehrere Beschriftungen derselben Kante möglich)
- $q_0$  wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände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

## • Übergangstabelle

- Tabellarische Darstellung der Funktion  $\delta$
- Kennzeichnung von  $q_0$  durch einen Pfeil
- Kennzeichnung von  $F$  durch **Sterne**
- $\Sigma$  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>TI</i>	<i>S</i>	<i>S</i>
	<i>TI</i>	<i>T</i>	<i>S</i>	<i>TI1</i>	<i>S</i>
*	<i>TI1</i>	<i>TI1</i>	<i>TI1</i>	<i>TI1</i>	<i>TI1</i>

# ARBEITSWEISE VON ENDLICHEN AUTOMATEN



- **Anfangssituation**

- Automat befindet sich im Startzustand  $q_0$

# ARBEITSWEISE VON ENDLICHEN AUTOMATEN



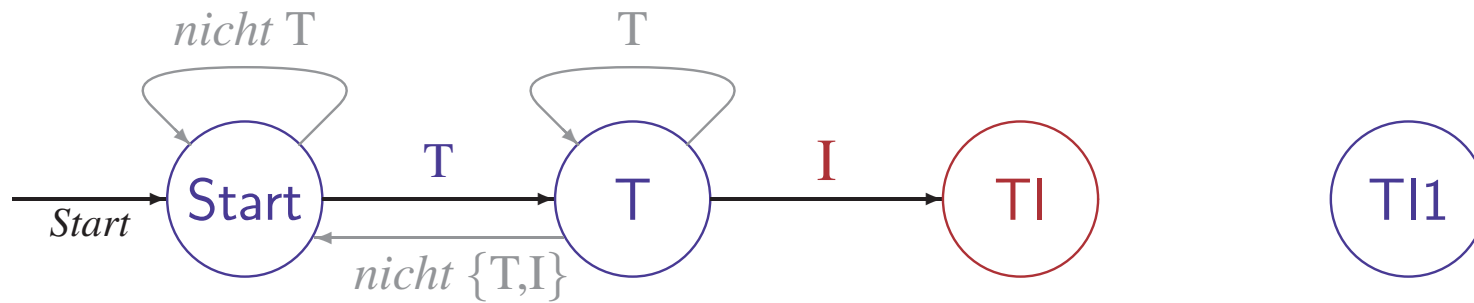
- **Anfangssituation**

- 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**

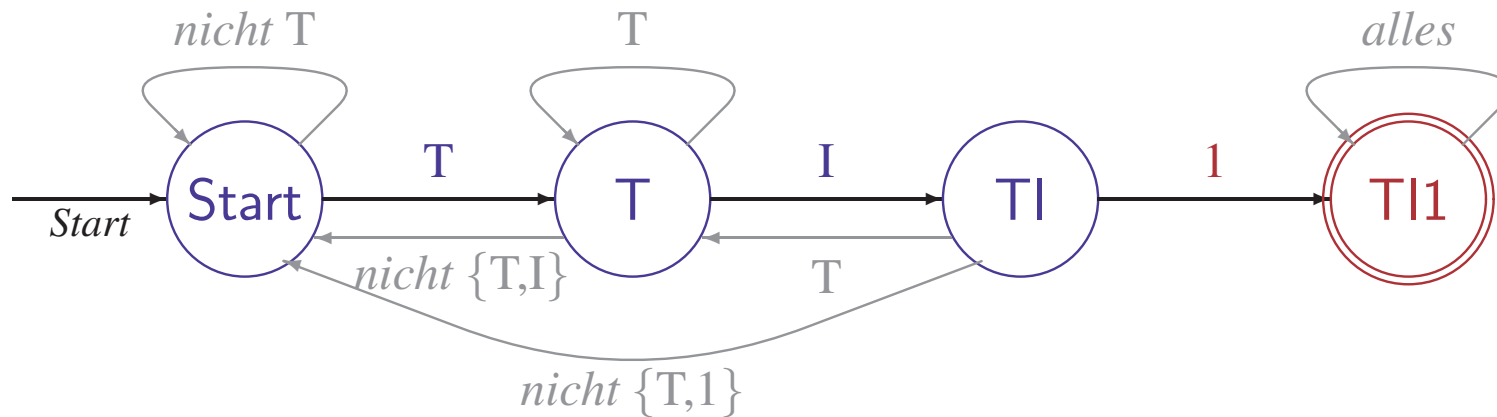
- Automat befindet sich im Startzustand  $q_0$

- **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  $q_0$

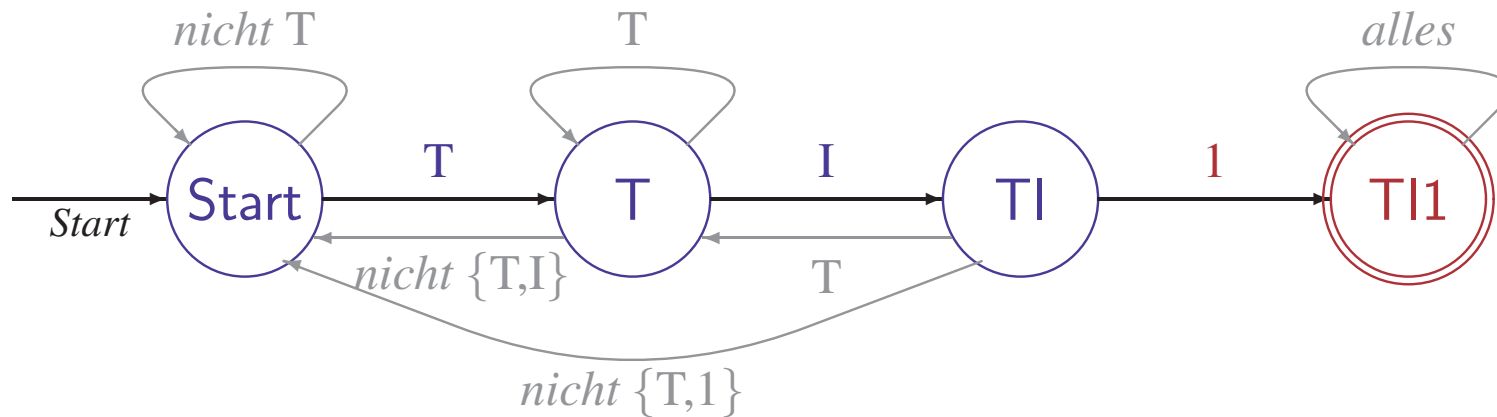
- **Arbeitsschritt**

- 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 und Ergebnis**

- Automat stoppt, wenn Eingabewort  $w = a_1..a_n$  komplett gelesen ist

- Der Automat befindet sich jetzt in einem Zustand  $q_n$

- Eingabe  $w$  wird akzeptiert, wenn  $q_n \in F$ , sonst wird  $w$  abgewiesen

- **Erweiterte Überföhrungsfunktion  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$** 
  - Schrittweise Abarbeitung der Eingabe mit  $\delta$  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**

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

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

- Schrittweise Abarbeitung der Eingabe mit  $\delta$  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**

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

- **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**

- **Erweiterte Überföhrungsfunktion  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$** 
  - Schrittweise Abarbeitung der Eingabe mit  $\delta$  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**

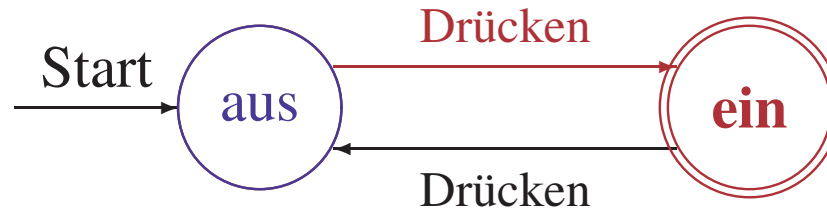
$$\hat{\delta}(q, w) = \begin{cases} q & \text{falls } w = \epsilon, \\ \delta(\hat{\delta}(q, v), a) & \text{falls } w = v a \text{ für ein } v \in \Sigma^*, a \in \Sigma \end{cases}$$

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

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

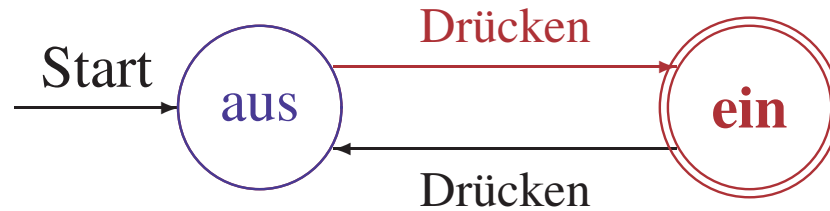
- Auch: die **von  $A$  erkannte Sprache**
- **Reguläre Sprache**
  - Sprache, die von einem DEA  $A$  akzeptiert wird

# ANALYSE DER SPRACHE DES WECHSELSCHALTERS



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

# 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**

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

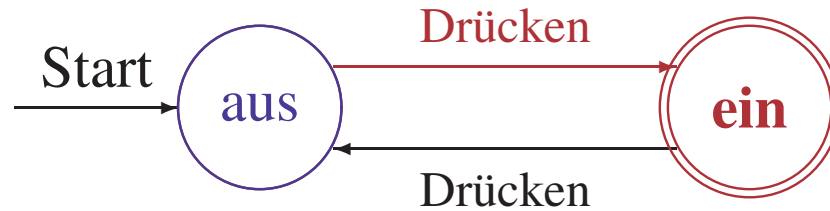
Zwei Behauptungen zu Eigenschaften von  $A$  sind zu zeigen:

- $B_1(n)$ : Ist  $n$  gerade, so ist  $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$

- $B_2(n)$ : Ist  $n$  ungerade, so ist  $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$

Beweis: simultane Induktion analog zu Einheit 1, Folie 29/30, Details auf nächster Folie

# 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**

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

Zwei Behauptungen zu Eigenschaften von  $A$  sind zu zeigen:

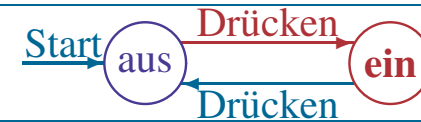
- $B_1(n)$ : Ist  $n$  gerade, so ist  $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$

- $B_2(n)$ : Ist  $n$  ungerade, so ist  $\hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$

Beweis: simultane Induktion analog zu Einheit 1, Folie 29/30, Details auf nächster Folie

- **Formale Beschreibung der Sprache von  $A$**

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



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

$$B_1(n): n \text{ gerade} \Leftrightarrow \hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$$

$$B_2(n): n \text{ ungerade} \Leftrightarrow \hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{ein}$$

**Induktionsanfang  $n=0$ :**

$B_1(n)$ : 0 ist gerade und  $\hat{\delta}(\text{aus}, \text{Drücken}^0) = \hat{\delta}(\text{aus}, \epsilon) = \text{aus}$ , also gilt Aussage  $B_1(n)$ .

$B_2(n)$ : 0 ist nicht ungerade und  $\hat{\delta}(\text{aus}, \text{Drücken}^0) \neq \text{ein}$ , also gilt die Äquivalenz  $B_2(n)$ , da jeweils die rechte und linke Seite falsch ist.

**Induktionsannahme:**  $B_1(m)$  und  $B_2(m)$  seien für ein beliebiges  $m \in \mathbb{N}$  gezeigt.

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

$B_1(n)$ : Es ist  $n = m+1$  gerade

$$\Leftrightarrow m \text{ ist ungerade}$$

$$\Leftrightarrow \hat{\delta}(\text{aus}, \text{Drücken}^m) = \text{ein} \quad (\text{Induktionsannahme } B_2(m))$$

$$\Leftrightarrow \hat{\delta}(\text{aus}, \text{Drücken}^n) = \text{aus}$$

$B_2(n)$ : Es ist  $n = m+1$  ungerade

$$\Leftrightarrow m \text{ ist gerade}$$

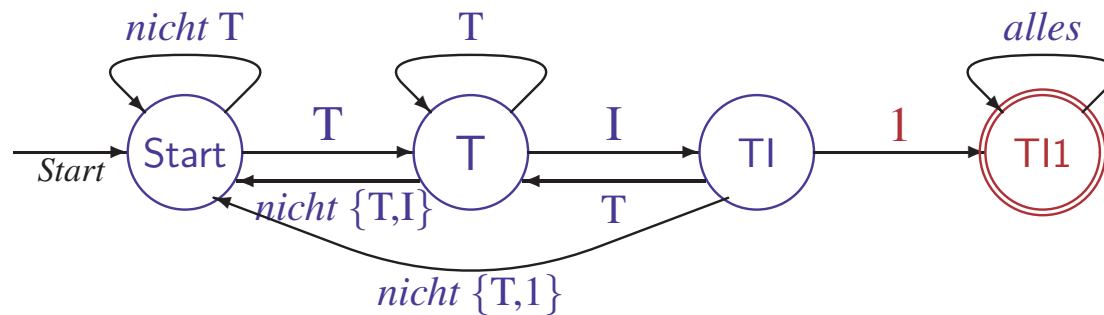
$$\Leftrightarrow \hat{\delta}(\text{aus}, \text{Drücken}^m) = \text{aus} \quad (\text{Induktionsannahme } B_1(m))$$

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

**Aufgrund des Induktionsprinzips gilt  $B_1(n)$  und  $B_2(n)$  für alle  $n \in \mathbb{N}$**



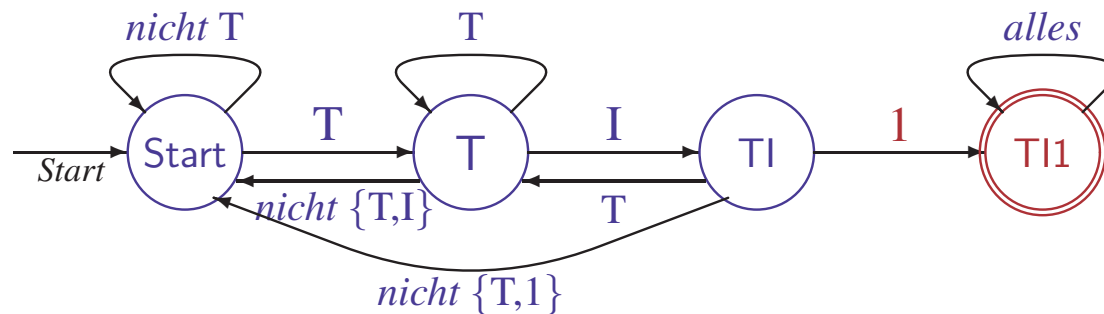
# ANALYSE DER SPRACHE DES “TI1”-AUTOMATEN



- **Sprache: Eingaben, die den Zustand **TI1** erreichen**

- Vermutung: Menge der Wörter, die TI1 als Teilwort enthalten
- Formale Beschreibung:  $L(A) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*. w = uTI1v\}$
- Zu beweisen ist also:  $\hat{\delta}(\text{Start}, w) \in \{TI1\} \Leftrightarrow \exists u, v \in \Sigma^*. w = uTI1v$

# ANALYSE DER SPRACHE DES “TI1”-AUTOMATEN



- **Sprache: Eingaben, die den Zustand TI1 erreichen**

- Vermutung: Menge der Wörter, die TI1 als Teilwort enthalten
- Formale Beschreibung:  $L(A) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*. w = uTI1v\}$
- Zu beweisen ist also:  $\hat{\delta}(\text{Start}, w) \in \{TI1\} \Leftrightarrow \exists u, v \in \Sigma^*. w = uTI1v$

- **Beweis zeigt, welche Wörter welchen Zustand erreichen**

Beweise drei Eigenschaften durch simultane Induktion:

(nächste Folie)

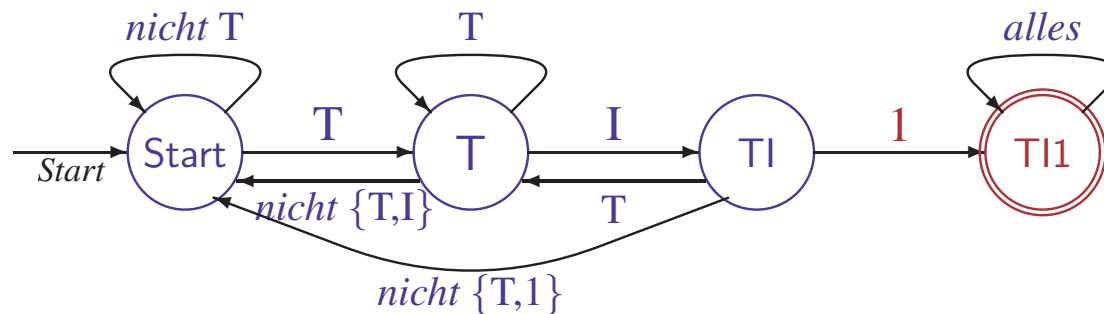
- $B_3(w)$ :  $\hat{\delta}(\text{Start}, w) = TI \Leftrightarrow \exists u \in \Sigma^*. w = uTI$

- $B_2(w)$ :  $\hat{\delta}(\text{Start}, w) = T \Leftrightarrow \exists u \in \Sigma^*. w = uT$

- $B_1(w)$ :  $\hat{\delta}(\text{Start}, w) = \text{Start}$  in allen anderen Fällen

Die Behauptung folgt dann aus  $B_3$  (und einer Induktion innerhalb von TI1)

# ANALYSE DER SPRACHE DES “TI1”-AUTOMATEN



- **Sprache: Eingaben, die den Zustand TI1 erreichen**

- Vermutung: Menge der Wörter, die TI1 als Teilwort enthalten
- Formale Beschreibung:  $L(A) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*. w = uTI1v\}$
- Zu beweisen ist also:  $\hat{\delta}(\text{Start}, w) \in \{TI1\} \Leftrightarrow \exists u, v \in \Sigma^*. w = uTI1v$

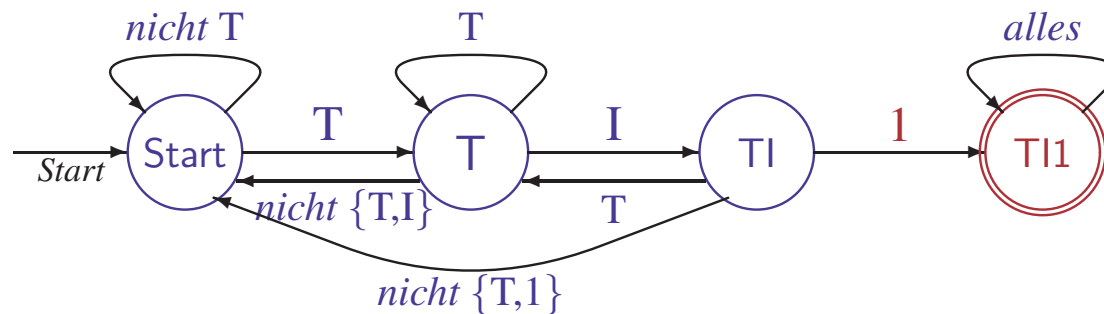
- **Beweis zeigt, welche Wörter welchen Zustand erreichen**

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

- $B_3(w)$ :  $\hat{\delta}(\text{Start}, w) = TI \Leftrightarrow \exists u \in \Sigma^*. w = uTI$  fehlt da nicht etwas ?
- $B_2(w)$ :  $\hat{\delta}(\text{Start}, w) = T \Leftrightarrow \exists u \in \Sigma^*. w = uT$
- $B_1(w)$ :  $\hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq uTI1v \wedge w \neq uTI \wedge w \neq uT$

Die Behauptung folgt dann aus  $B_3$  (und einer Induktion innerhalb von TI1)

# ANALYSE DER SPRACHE DES “TI1”-AUTOMATEN



- **Sprache: Eingaben, die den Zustand TI1 erreichen**

- Vermutung: Menge der Wörter, die TI1 als Teilwort enthalten
- Formale Beschreibung:  $L(A) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*. w = uTI1v\}$
- Zu beweisen ist also:  $\hat{\delta}(\text{Start}, w) \in \{TI1\} \Leftrightarrow \exists u, v \in \Sigma^*. w = uTI1v$

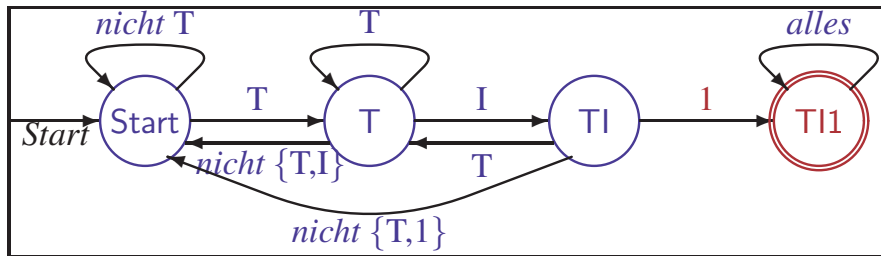
- **Beweis zeigt, welche Wörter welchen Zustand erreichen**

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

- $B_3(w)$ :  $\hat{\delta}(\text{Start}, w) = TI \Leftrightarrow \exists u \in \Sigma^*. w = uTI \wedge \forall u, v \in \Sigma^*. w \neq uTI1v$
- $B_2(w)$ :  $\hat{\delta}(\text{Start}, w) = T \Leftrightarrow \exists u \in \Sigma^*. w = uT \wedge \forall u, v \in \Sigma^*. w \neq uTI1v$
- $B_1(w)$ :  $\hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq uTI1v \wedge w \neq uTI \wedge w \neq uT$

Die Behauptung folgt dann aus  $B_3$  (und einer Induktion innerhalb von TI1)

# BEWEIS DER SPRACHE DES “TI1”-AUTOMATEN



Beweise durch simultane (strukturelle) Induktion:

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T}$$

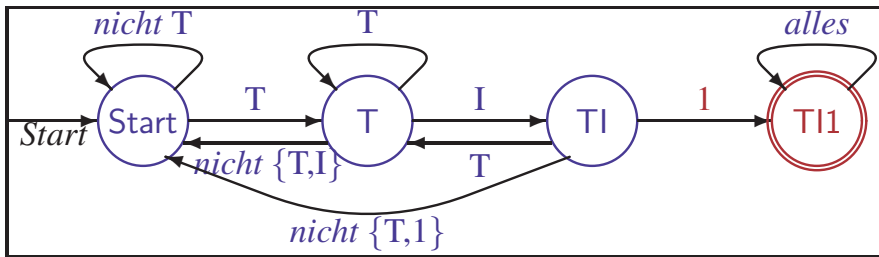
**Induktionsanfang  $w = \epsilon$ :**

$B_1(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{Start}$  gilt und  $w = \epsilon$  hat keine der drei “verbotenen” Formen ✓

$B_2(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{T}$  gilt nicht und  $w = \epsilon$  endet nicht mit T ✓

$B_3(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{TI}$  gilt nicht und  $w = \epsilon$  endet nicht mit TI ✓

# BEWEIS DER SPRACHE DES “TI1”-AUTOMATEN



Beweis durch simultane (strukturelle) Induktion:

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T}$$

**Induktionsanfang**  $w = \epsilon$ :

$B_1(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{Start}$  gilt und  $w = \epsilon$  hat keine der drei “verbotenen” Formen ✓

$B_2(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{T}$  gilt nicht und  $w = \epsilon$  endet nicht mit T ✓

$B_3(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{TI}$  gilt nicht und  $w = \epsilon$  endet nicht mit TI ✓

**Induktionsannahme:** die Aussagen  $B_1(w')$ – $B_3(w')$  seien für ein beliebiges  $w' \in \Sigma^*$  gezeigt.

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

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start}$$

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

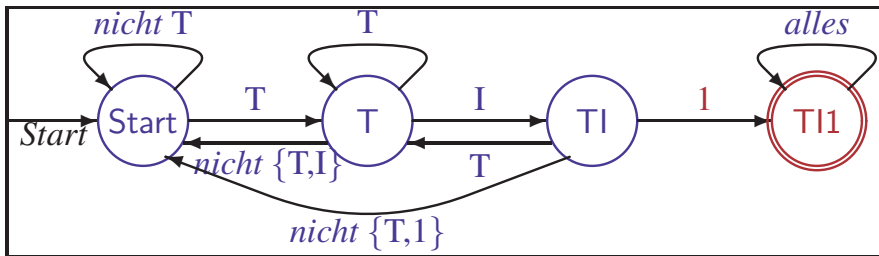
$$\Leftrightarrow \forall u, v \in \Sigma^*. w' \neq u\text{TI}1v \wedge w' \neq u\text{TI} \wedge w' \neq u\text{T} \wedge a \neq \text{T} \quad B_1(w')$$

$$\vee \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge a \notin \{\text{T}, \text{I}\} \quad B_2(w')$$

$$\vee \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge a \notin \{\text{T}, 1\} \quad B_3(w')$$

$$\Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T} \quad \checkmark$$

# BEWEIS DER SPRACHE DES “TI1”-AUTOMATEN



Beweise durch simultane (strukturelle) Induktion:

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T}$$

**Induktionsanfang**  $w = \epsilon$ :

$B_1(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{Start}$  gilt und  $w = \epsilon$  hat keine der drei “verbotenen” Formen ✓

$B_2(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{T}$  gilt nicht und  $w = \epsilon$  endet nicht mit T ✓

$B_3(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{TI}$  gilt nicht und  $w = \epsilon$  endet nicht mit TI ✓

**Induktionsannahme:** die Aussagen  $B_1(w')$ – $B_3(w')$  seien für ein beliebiges  $w' \in \Sigma^*$  gezeigt.

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

$B_1(w)$ :  $\hat{\delta}(\text{Start}, w) = \text{Start}$

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

$$\Leftrightarrow \forall u, v \in \Sigma^*. w' \neq u\text{TI}1v \wedge w' \neq u\text{TI} \wedge w' \neq u\text{T} \wedge a \neq \text{T} \quad B_1(w')$$

$$\vee \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge a \notin \{\text{T}, \text{I}\} \quad B_2(w')$$

$$\vee \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge a \notin \{\text{T}, 1\} \quad B_3(w')$$

$$\Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T} \quad \checkmark$$

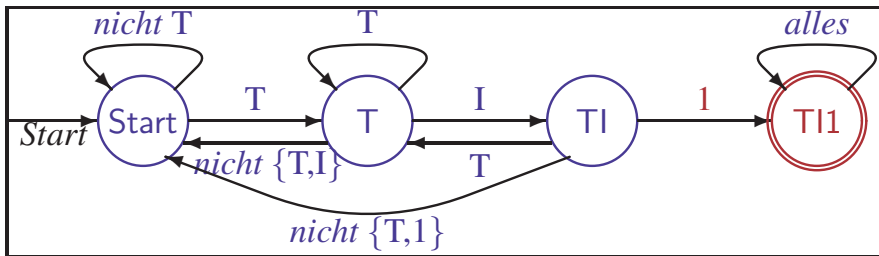
$B_2(w)$ :  $\hat{\delta}(\text{Start}, w) = \text{T}$

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

$$\Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

mit  $B_1(w')$ ,  $B_2(w')$ ,  $B_3(w')$  ✓

# BEWEIS DER SPRACHE DES “TI1”-AUTOMATEN



Beweise durch simultane (strukturelle) Induktion:

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T}$$

**Induktionsanfang**  $w = \epsilon$ :

$B_1(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{Start}$  gilt und  $w = \epsilon$  hat keine der drei “verbotenen” Formen ✓

$B_2(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{T}$  gilt nicht und  $w = \epsilon$  endet nicht mit T ✓

$B_3(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{TI}$  gilt nicht und  $w = \epsilon$  endet nicht mit TI ✓

**Induktionsannahme:** die Aussagen  $B_1(w')$ – $B_3(w')$  seien für ein beliebiges  $w' \in \Sigma^*$  gezeigt.

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

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start}$$

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

$$\Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T} \quad \text{mit } B_1(w'), B_2(w'), B_3(w') \quad \checkmark$$

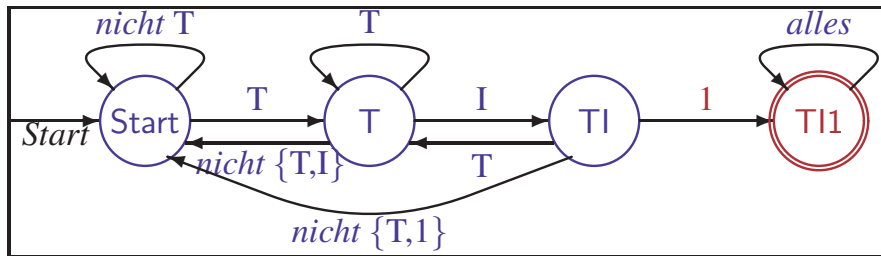
$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T}$$

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

$$\Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \quad \text{mit } B_1(w'), B_2(w'), B_3(w') \quad \checkmark$$



# BEWEIS DER SPRACHE DES “TI1”-AUTOMATEN



Beweise durch simultane (strukturelle) Induktion:

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T} \Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v$$

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start} \Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T}$$

**Induktionsanfang**  $w = \epsilon$ :

$B_1(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{Start}$  gilt und  $w = \epsilon$  hat keine der drei “verbotenen” Formen ✓

$B_2(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{T}$  gilt nicht und  $w = \epsilon$  endet nicht mit T ✓

$B_3(w)$ :  $\hat{\delta}(\text{Start}, \epsilon) = \text{TI}$  gilt nicht und  $w = \epsilon$  endet nicht mit TI ✓

**Induktionsannahme:** die Aussagen  $B_1(w')$ – $B_3(w')$  seien für ein beliebiges  $w' \in \Sigma^*$  gezeigt.

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

$$B_1(w): \hat{\delta}(\text{Start}, w) = \text{Start}$$

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

$$\Leftrightarrow \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \wedge w \neq u\text{TI} \wedge w \neq u\text{T} \quad \text{mit } B_1(w'), B_2(w'), B_3(w') \quad \checkmark$$

$$B_2(w): \hat{\delta}(\text{Start}, w) = \text{T}$$

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

$$\Leftrightarrow \exists u \in \Sigma^*. w = u\text{T} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \quad \text{mit } B_1(w'), B_2(w'), B_3(w') \quad \checkmark$$

$$B_3(w): \hat{\delta}(\text{Start}, w) = \text{TI}$$

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

$$\Leftrightarrow \exists u \in \Sigma^*. w = u\text{TI} \wedge \forall u, v \in \Sigma^*. w \neq u\text{TI}1v \quad \text{mit } B_2(w') \quad \checkmark$$

**Aufgrund des Induktionsprinzips gilt  $B_1(w)$ ,  $B_2(w)$  und  $B_3(w)$  für alle  $w \in \Sigma^*$ .**

## Erkenne, ob eine Binärzahl durch 3 teilbar ist

- **Binärzahl wird von links nach rechts gelesen**

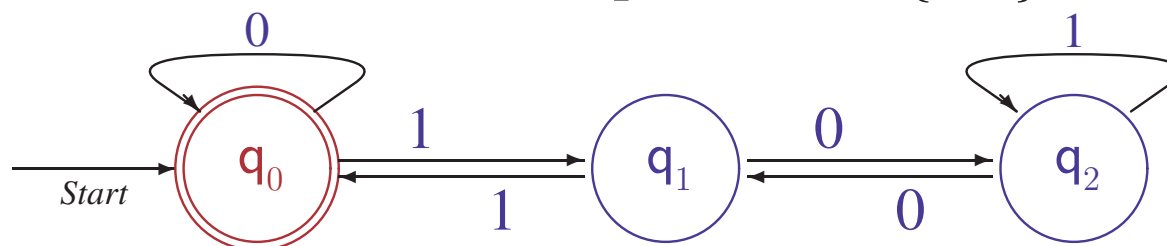
- Eingabewort  $w = w_0..w_n$  entspricht der Zahl  $n = r_b(w) = \sum_{j=0}^n w_j \cdot 2^{n-j}$

## Erkenne, ob eine Binärzahl durch 3 teilbar ist

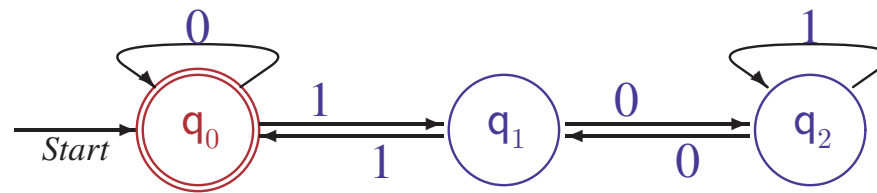
- **Binärzahl wird von links nach rechts gelesen**
  - Eingabewort  $w = w_0..w_n$  entspricht der Zahl  $n = r_b(w) = \sum_{j=0}^n w_j \cdot 2^{n-j}$
- **Konstruktion des Automaten aus induktiver Beweisidee**
  - Das Eingabewort  $w = \epsilon$  entspricht der Zahl  $n = 0$
  - Entspricht  $v$  der 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  $2i+a \bmod 3 = 2(i \bmod 3) + a \bmod 3$

## Erkenne, ob eine Binärzahl durch 3 teilbar ist

- **Binärzahl wird von links nach rechts gelesen**
  - Eingabewort  $w = w_0..w_n$  entspricht der Zahl  $n = r_b(w) = \sum_{j=0}^n w_j \cdot 2^{n-j}$
- **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  $\Sigma = \{0, 1\}$

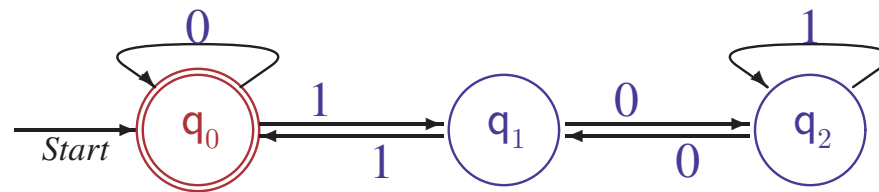


ZEIGE  $L(A) = \{w \mid r_b(w) \bmod 3 = 0\}$



- **Zeige durch simultane strukturelle Induktion über  $w$** 
  - $B_j(w)$ :  $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \bmod 3 = j$  für  $j \in \{0, 1, 2\}$

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



- **Zeige durch simultane strukturelle Induktion über  $w$**

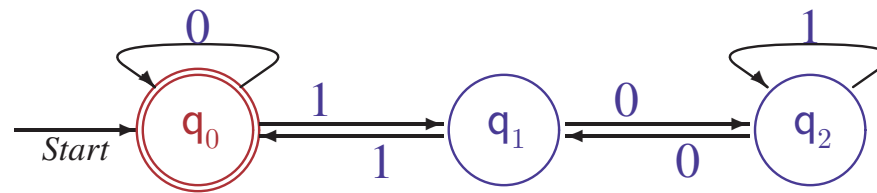
- $B_j(w)$ :  $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \bmod 3 = j$  für  $j \in \{0, 1, 2\}$

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

- Es ist  $\hat{\delta}(q_0, \epsilon) = q_0$  und  $r_b(\epsilon) \bmod 3 = 0$

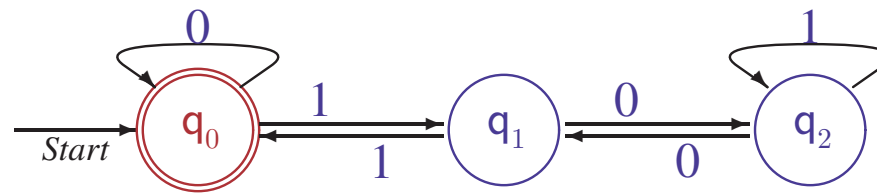
- Damit gilt  $B_0(w)$  und auch  $B_1(w)$  und  $B_2(w)$  (beide Seiten von  $\Leftrightarrow$  sind falsch) ✓

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



- **Zeige durch simultane strukturelle Induktion über  $w$** 
  - $B_j(w)$ :  $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \bmod 3 = j$  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  $B_0(w)$  und auch  $B_1(w)$  und  $B_2(w)$  (beide Seiten von  $\Leftrightarrow$  sind falsch) ✓
- **Induktionsannahme:  $B_j(w')$  sei gezeigt für  $w' \in \Sigma^*$  und  $j \in \{0, 1, 2\}$**

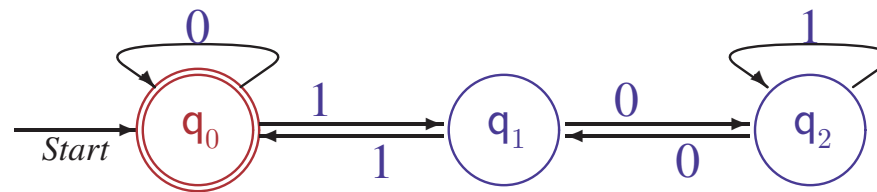
ZEIGE  $L(A) = \{w \mid r_b(w) \bmod 3 = 0\}$



- **Zeige durch simultane strukturelle Induktion über  $w$** 
  - $B_j(w)$ :  $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \bmod 3 = j$  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  $B_0(w)$  und auch  $B_1(w)$  und  $B_2(w)$  (beide Seiten von  $\Leftrightarrow$  sind falsch) ✓
- **Induktionsannahme:  $B_j(w')$  sei gezeigt für  $w' \in \Sigma^*$  und  $j \in \{0, 1, 2\}$**
- **Induktionsschritt: sei  $w = w'a$  für ein beliebiges  $a \in \Sigma$** 
  - $\hat{\delta}(q_0, w) = q_j$
  - $\Leftrightarrow \exists i. \hat{\delta}(q_0, w') = q_i \wedge \delta(q_i, a) = q_j$
  - $\Leftrightarrow \exists i. r_b(w') \bmod 3 = i \wedge j = 2i + a \bmod 3$
  - $\Leftrightarrow r_b(w) \bmod 3 = 2r_b(w') + a \bmod 3 = 2(r_b(w') \bmod 3) + a \bmod 3 = j$  ✓



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



- **Zeige durch simultane strukturelle Induktion über  $w$** 
  - $B_j(w)$ :  $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \bmod 3 = j$  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  $B_0(w)$  und auch  $B_1(w)$  und  $B_2(w)$  (beide Seiten von  $\Leftrightarrow$  sind falsch) ✓
- **Induktionsannahme:  $B_j(w')$  sei gezeigt für  $w' \in \Sigma^*$  und  $j \in \{0, 1, 2\}$**
- **Induktionsschritt: sei  $w = w'a$  für ein beliebiges  $a \in \Sigma$** 
  - $\hat{\delta}(q_0, w) = q_j$
  - $\Leftrightarrow \exists i. \hat{\delta}(q_0, w') = q_i \wedge \delta(q_i, a) = q_j$
  - $\Leftrightarrow \exists i. r_b(w') \bmod 3 = i \wedge j = 2i + a \bmod 3$
  - $\Leftrightarrow r_b(w) \bmod 3 = 2r_b(w') + a \bmod 3 = 2(r_b(w') \bmod 3) + a \bmod 3 = j$  ✓
- **Damit gilt  $B_j(w)$  für alle  $w \in \Sigma^*$  und  $j \in \{0, 1, 2\}$  und es folgt**

$$L(A) = \{w \mid \hat{\delta}(q_0, w) = q_0\} = \{w \mid r_b(w) \bmod 3 = 0\}$$

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

Für DEAs weniger intuitiv, aber leichter zu verallgemeinern

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

## Codierte Anzahl der gelesener 0/1 im Zustand

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

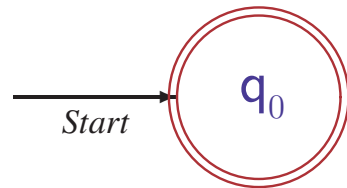
$q_2 \hat{=} (\text{ungerade, gerade})$       $q_3 \hat{=} (\text{ungerade, ungerade})$

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

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

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

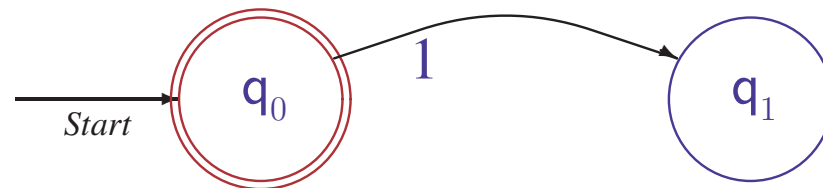


DEA FÜR  $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

## Codiere Anzahl der gelesener 0/1 im Zustand

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

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

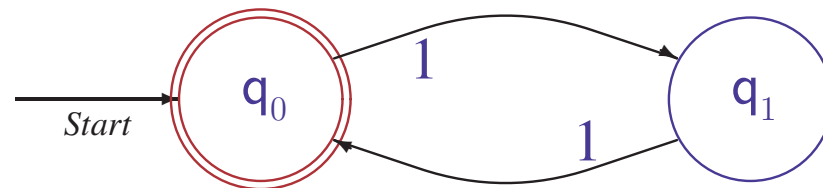


DEA FÜR  $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

## Codiere Anzahl der gelesener 0/1 im Zustand

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

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



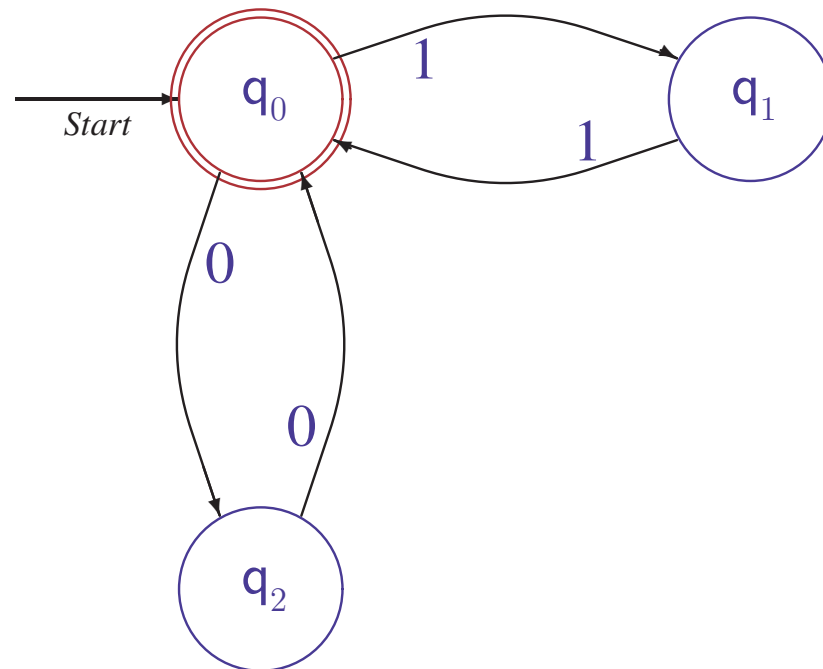


DEA FÜR  $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

## Codiere Anzahl der gelesener 0/1 im Zustand

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

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

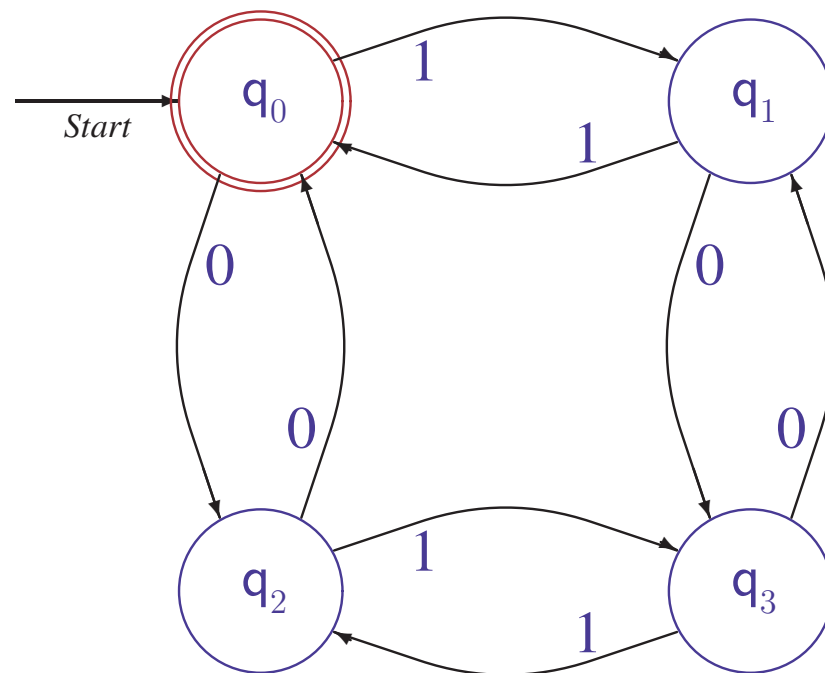


DEA FÜR  $L = \{w \in \{0, 1\}^* \mid w \text{ enthält gerade Anzahl von } 0 \text{ und } 1\}$

## 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**

# KORREKTHEITSBEWEIF MIT KONFIGURATIONEN

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

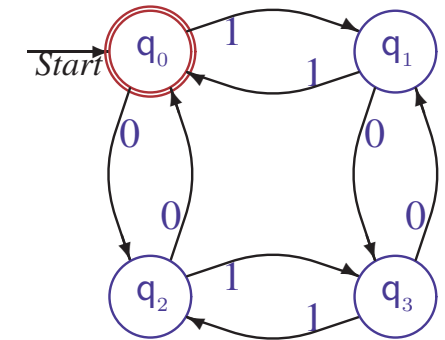
$B_0(w): (q_0, wv) \vdash^* (q_0, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $g_1(w)$

$B_1(w): (q_0, wv) \vdash^* (q_1, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $u_1(w)$

$B_2(w): (q_0, wv) \vdash^* (q_2, v) \Leftrightarrow$  es gilt  $u_0(w)$  und  $g_1(w)$

$B_3(w): (q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$  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, ...



# KORREKTHEITSBEWEIF MIT KONFIGURATIONEN

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

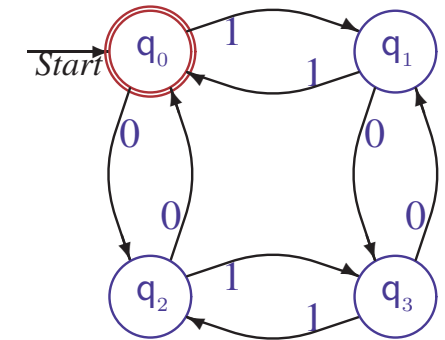
$B_0(w): (q_0, wv) \vdash^* (q_0, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $g_1(w)$

$B_1(w): (q_0, wv) \vdash^* (q_1, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $u_1(w)$

$B_2(w): (q_0, wv) \vdash^* (q_2, v) \Leftrightarrow$  es gilt  $u_0(w)$  und  $g_1(w)$

$B_3(w): (q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$  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:** Sei  $w = \epsilon$

– Per Definition ist  $(q_0, \epsilon v) \vdash^* (q_0, \epsilon v)$ ,  $g_0(\epsilon)$  und  $g_1(\epsilon)$ , also  $B_0(\epsilon)..B_3(\epsilon)$  ✓

# KORREKTHEITSBEWEIF MIT KONFIGURATIONEN

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

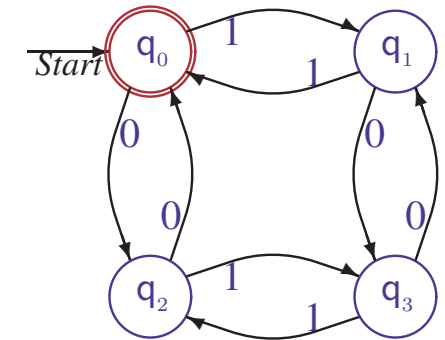
$B_0(w): (q_0, wv) \vdash^* (q_0, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $g_1(w)$

$B_1(w): (q_0, wv) \vdash^* (q_1, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $u_1(w)$

$B_2(w): (q_0, wv) \vdash^* (q_2, v) \Leftrightarrow$  es gilt  $u_0(w)$  und  $g_1(w)$

$B_3(w): (q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$  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:** Sei  $w = \epsilon$

– Per Definition ist  $(q_0, \epsilon v) \vdash^* (q_0, \epsilon v)$ ,  $g_0(\epsilon)$  und  $g_1(\epsilon)$ , also  $B_0(\epsilon)..B_3(\epsilon)$  ✓

- **Annahme:**  $B_0(w')..B_3(w')$  sei bewiesen für ein  $w' \in \{0, 1\}^*$

# KORREKTHEITSBEWWEIS MIT KONFIGURATIONEN

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

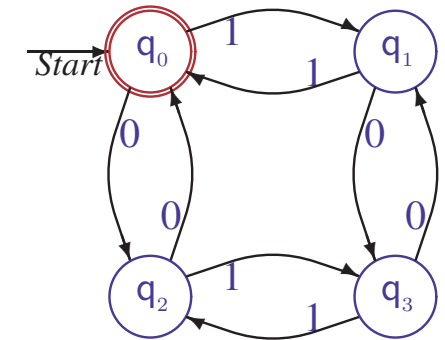
$B_0(w): (q_0, wv) \vdash^* (q_0, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $g_1(w)$

$B_1(w): (q_0, wv) \vdash^* (q_1, v) \Leftrightarrow$  es gilt  $g_0(w)$  und  $u_1(w)$

$B_2(w): (q_0, wv) \vdash^* (q_2, v) \Leftrightarrow$  es gilt  $u_0(w)$  und  $g_1(w)$

$B_3(w): (q_0, wv) \vdash^* (q_3, v) \Leftrightarrow$  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: Sei  $w = \epsilon$**

– Per Definition ist  $(q_0, \epsilon v) \vdash^* (q_0, \epsilon v)$ ,  $g_0(\epsilon)$  und  $g_1(\epsilon)$ , also  $B_0(\epsilon)..B_3(\epsilon)$  ✓

- **Annahme:  $B_0(w')..B_3(w')$  sei bewiesen für ein  $w' \in \{0, 1\}^*$**

- **Schrittfall: Sei  $w = w'a$  für ein  $a \in \{0, 1\}$**

$B_0(w), \Rightarrow$  : Es gelte  $(q_0, wv) \vdash^* (q_0, v)$ .

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

Falls  $a = 0$ , dann ist  $p = q_2$  und wegen  $B_2(w')$  folgt  $u_0(w')$  und  $g_1(w')$ .

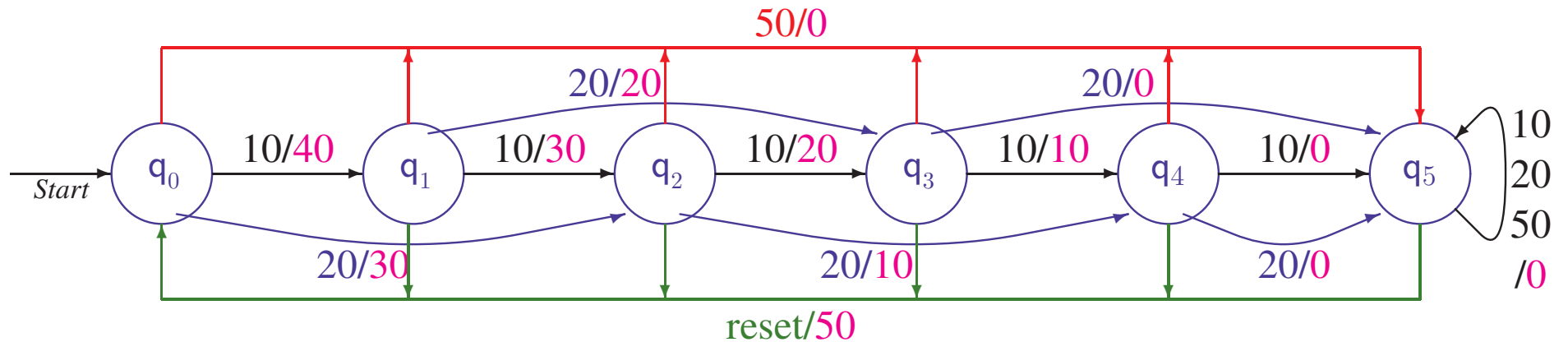
Falls  $a = 1$ , dann ist  $p = q_1$  und wegen  $B_1(w')$  folgt  $g_0(w')$  und  $u_1(w')$ .

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

Gegenrichtung durch Umkehrung des Arguments.  $B_1(w), B_2(w), B_3(w)$  analog.

# ENDLICHE AUTOMATEN MIT AUSGABEFUNKTION

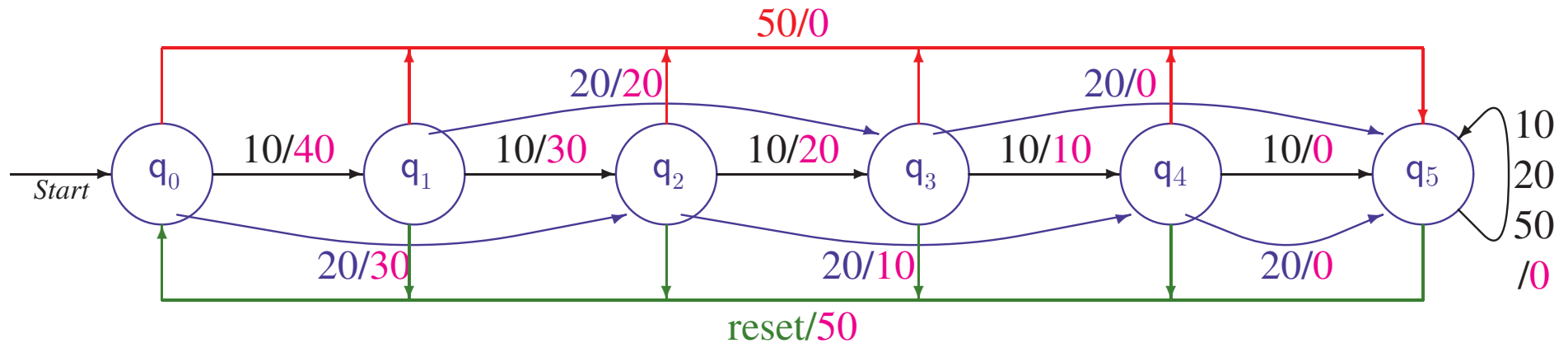
- 50c Kaffeeautomat mit Restbetragsanzeige



– Münzeinwurf führt zu Zustandsänderung und erzeugt Ausgabe

# ENDLICHE AUTOMATEN MIT AUSGABEFUNKTION

## ● 50c Kaffeeautomat mit Restbetragsanzeige



– Münzeinwurf führt zu Zustandsänderung und erzeugt Ausgabe

## ● Formalisierungen von Automaten mit Ausgabe

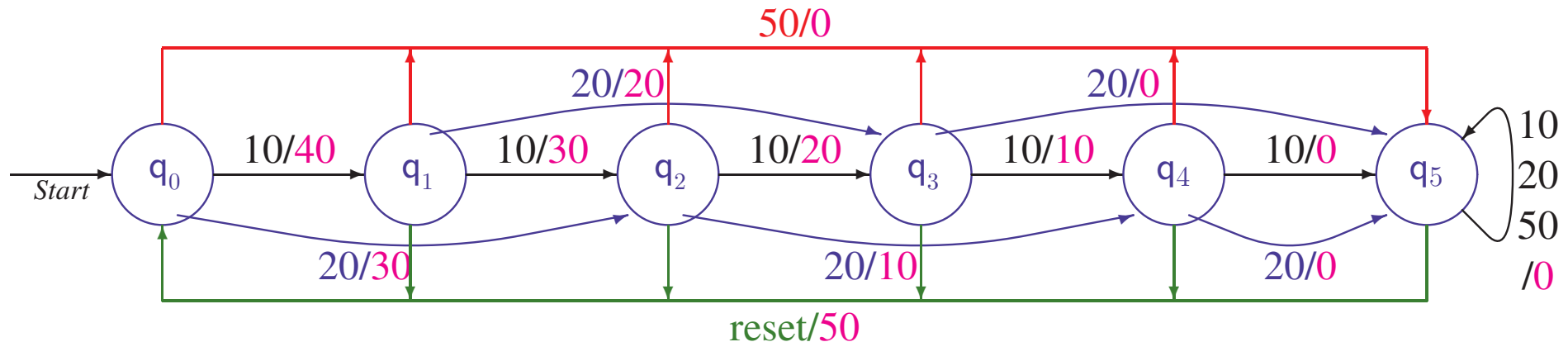
– **Mealy-Automaten**: Ausgabefunktion abhängig von Eingabe & Zustand

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



# 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**

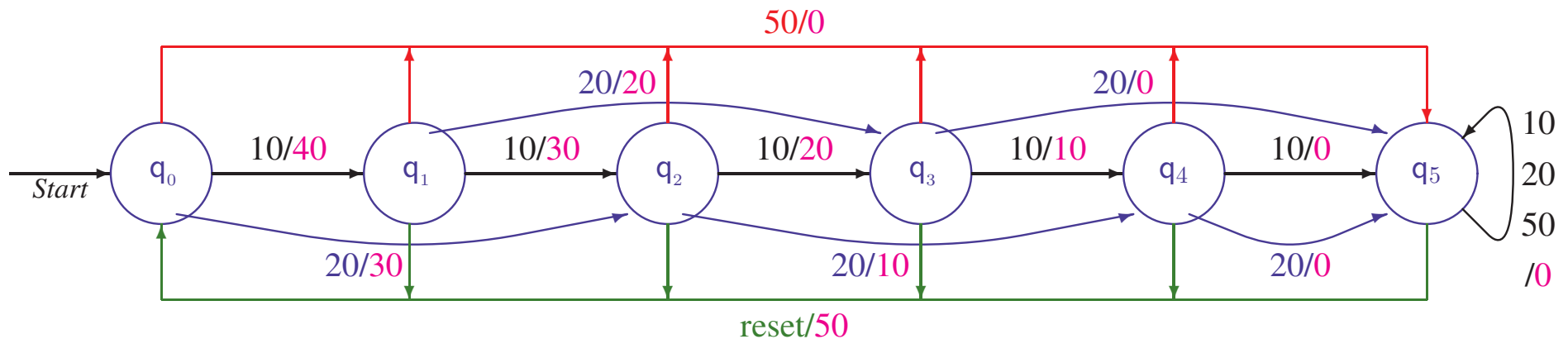
- Mealy- und Moore-Automaten sind äquivalent

- 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**
- $\Sigma$  (endliches) **Eingabealphabet**
- $\Delta$  (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  $q_0$
- **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

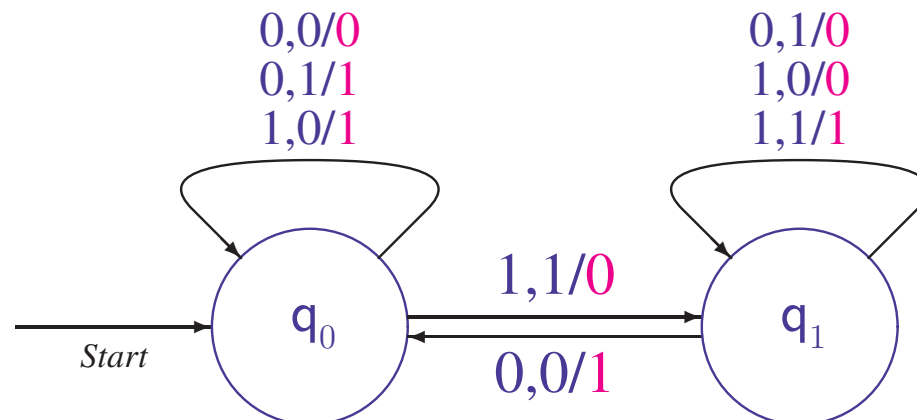
- **Addition von Bitpaaren von rechts nach links**
  - Eingabealphabet  $\Sigma = \{0, 1\} \times \{0, 1\}$
  - Ausgabealphabet  $\Delta = \{0, 1\}$

# MEALY-AUTOMAT FÜR (INVERSE) BINÄRADDITION

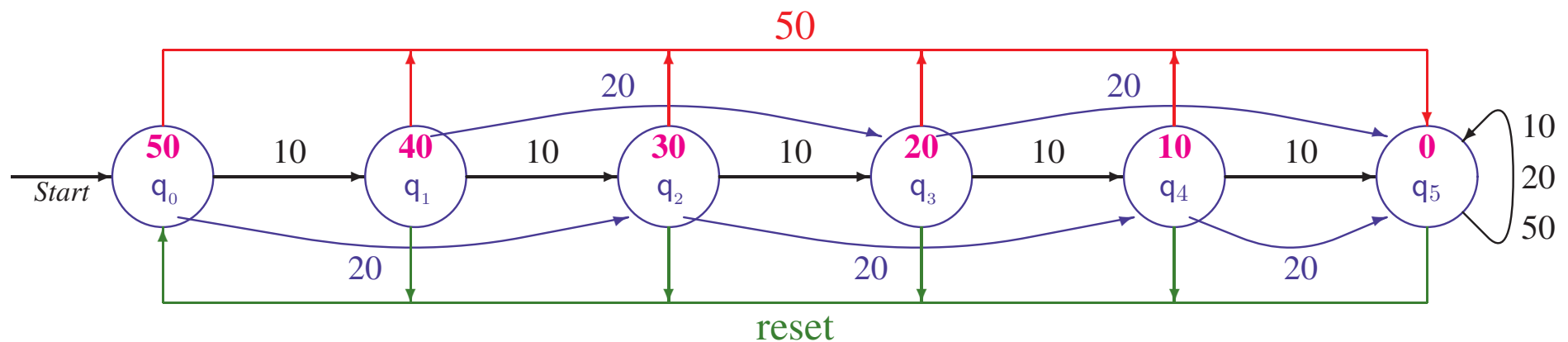
- **Addition von Bitpaaren von rechts nach links**
  - Eingabealphabet  $\Sigma = \{0, 1\} \times \{0, 1\}$
  - Ausgabealphabet  $\Delta = \{0, 1\}$
- **Zwei Zustände sind erforderlich**
  - Zustand  $q_0$ :  $A$  kann Addition zweier Bits direkt ausführen
  - Zustand  $q_1$ :  $A$  hat bei Addition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



# MOORE-AUTOMATEN – MATHEMATISCH PRÄZISIERT



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

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



- **Anfangssituation:** Automat im Startzustand  $q_0$ , Ausgabe  $x_0 = \lambda(q_0)$
  - **Arbeitsschritt**
    - 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  $x_0x_1..x_n$
-

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

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

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

# MOORE-AUTOMAT FÜR DIVISIONSREST

- **Eingabe einer Bitfolge von links nach rechts**
  - Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl  $n$
  - Ausgabe ist jeweils “ $n \bmod 3$ ”
  - Eingabealphabet  $\Sigma = \{0, 1\}$ , Ausgabealphabet  $\Delta = \{0, 1, 2\}$

# MOORE-AUTOMAT FÜR DIVISIONSREST

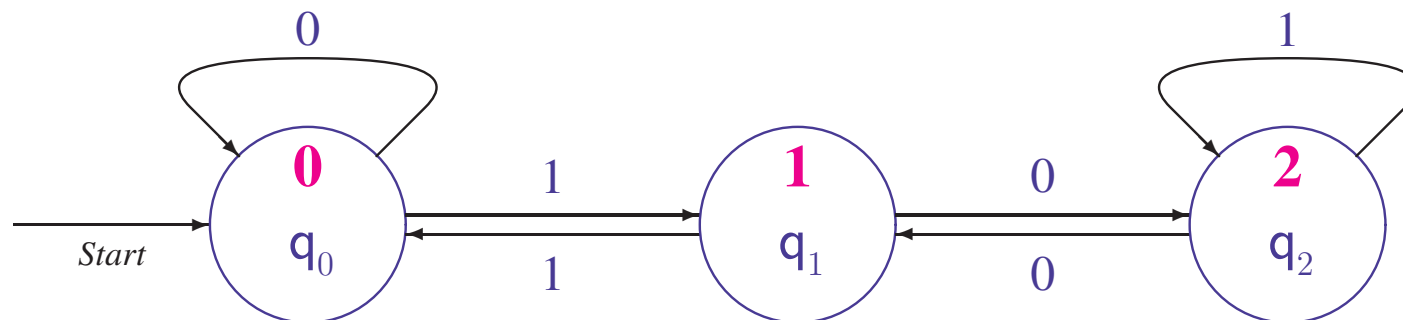
- **Eingabe einer Bitfolge von links nach rechts**
  - Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl  $n$
  - Ausgabe ist jeweils “ $n \bmod 3$ ”
  - Eingabealphabet  $\Sigma = \{0, 1\}$ , Ausgabealphabet  $\Delta = \{0, 1, 2\}$
- **Drei Zustände sind erforderlich**
  - Zustand  $q_0$ : Bisheriger Divisionsrest ist 0 (Ausgabe 0)
  - Zustand  $q_1$ : Bisheriger Divisionsrest ist 1 (Ausgabe 1)
  - Zustand  $q_2$ : Bisheriger Divisionsrest ist 2 (Ausgabe 2)

# MOORE-AUTOMAT FÜR DIVISIONSREST

- **Eingabe einer Bitfolge von links nach rechts**
  - Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl  $n$
  - Ausgabe ist jeweils “ $n \bmod 3$ ”
  - 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  $\delta(q_i, j) = q_{2*i+j \bmod 3}$

# MOORE-AUTOMAT FÜR DIVISIONSREST

- **Eingabe einer Bitfolge von links nach rechts**
  - Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl  $n$
  - Ausgabe ist jeweils “ $n \bmod 3$ ”
  - 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  $\delta(q_i, j) = q_{2*i+j \bmod 3}$
- **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  $\Leftrightarrow \chi_L$  “Moore-berechenbar”

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

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

- $\mathbf{graph}^*(f) = \{(w_1, v_0, v_1) \dots (w_n, v_n) \mid f(w_1 \dots w_n) = v_0 \dots v_n\}$

- DEAs können Graphen berechneter Funktionen akzeptieren

Satz:  $f$  Moore-berechenbar  $\Leftrightarrow \mathbf{graph}^*(f)$  reguläre Sprache

## BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **$L$  regulär  $\Leftrightarrow \chi_L$  “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  $f_M(w)$



# BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **$L$  regulär  $\Leftrightarrow \chi_L$  “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  $\Leftrightarrow \text{graph}^*(f)$  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  $(w_1, v_0, v_1)..(w_n, v_n) \in L(A)$

# BEWEIS DER ÄQUIVALENZ (SKIZZE)

- **$L$  regulär  $\Leftrightarrow \chi_L$  “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  $\Leftrightarrow \text{graph}^*(f)$  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