Theoretische Informatik I

Einheit 2

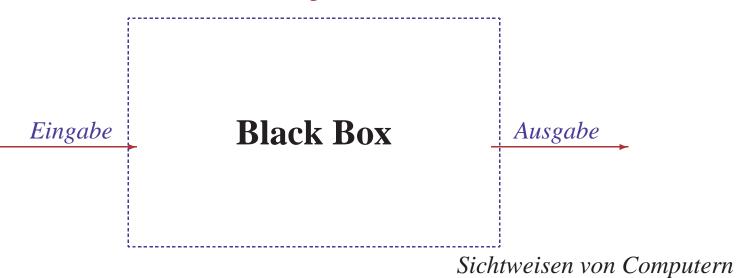


Endliche Automaten & Reguläre Sprachen

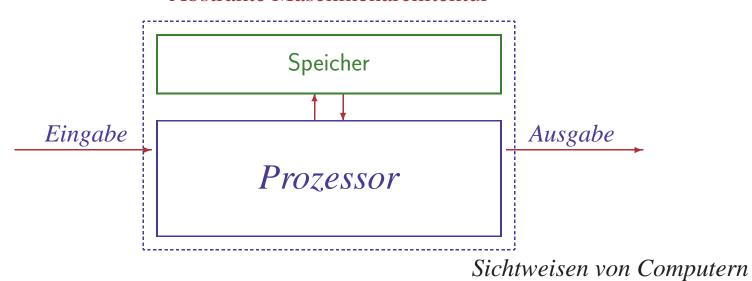


- 1. Deterministische endliche Automaten
- 2. Nichtdeterministische Automaten
- 3. Reguläre Ausdrücke
- 4. Grammatiken
- 5. Eigenschaften regulärer Sprachen

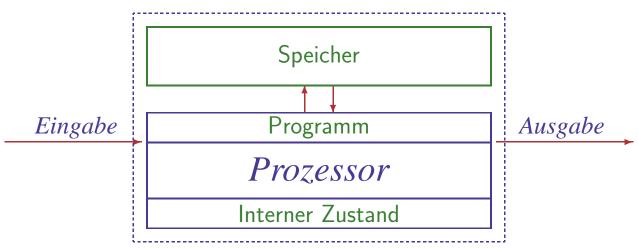
Betrachtung von außen



Abstrakte Maschinenarchitektur

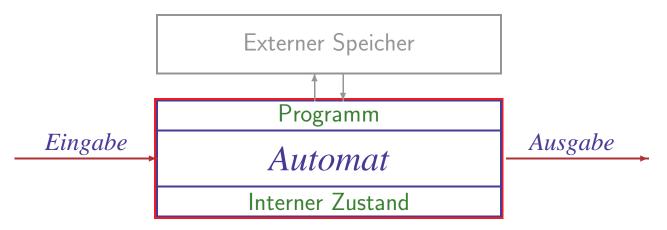


Details des Berechnungsmoduls



Sichtweisen von Computern

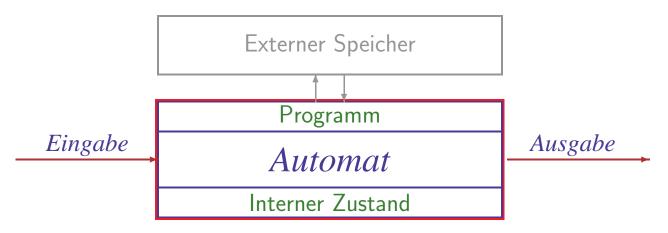
Aus der Sicht des Berechnungsmoduls



Sichtweisen von Computern

Automaten stehen im Kern jeder Berechnung

- Schnelle, direkte Verarbeitung von Eingaben
- Keine interne Speicherung von Daten
- Speicher sind Teil der Umgebung



Sichtweisen von Computern

• Automaten stehen im Kern jeder Berechnung

- Schnelle, direkte Verarbeitung von Eingaben
- Keine interne Speicherung von Daten
- Speicher sind Teil der Umgebung

• Endliche Automaten sind leicht zu analysieren

- Jede Berechnung endet nach einer festen Anzahl von Schritten
- Keine Schleifen oder Seiteneffekte

Verwendungszwecke für endliche Automaten

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

Steuerungsautomaten

Alle Formen rein Hardware-gesteuerter automatischer Maschinen
 Waschmaschinen, Autos, Unterhaltungselektronik, Ampelanlagen, Computerprozessoren

• Entwurf und Überprüfung digitaler Schaltungen

- Entwicklungswerkzeuge & Testsoftware beschreiben endliches Verhalten

• Lexikalische Analyse in Compilern

- Schnelle Identifizierung von Bezeichnern, Schlüsselwörtern, ...

• Textsuche in umfangreichen Dokumenten

– Z.B. Suche nach Webseiten mithilfe von Schlüsselwörtern

Software mit endlichen Alternativen

- Kommunikationsprotokolle, Protokolle zum sicheren Datenaustausch ...

AUTOMATEN BESCHREIBEN SPRACHEN

• Generierte Sprache

- Menge aller möglichen Ausgaben des Automaten

• Erkannte Sprache

- Menge aller Eingaben, die zur Ausgabe "ja" führen
- Alternativ: letzter Zustand des Automaten muß ein "Endzustand" sein

• Sprachen endlicher Automaten sind einfach

- Nur sehr einfach strukturierte Sprachen können beschrieben werden
- Durch endliche Automaten beschreibbare Sprachen heißen regulär

• Automaten: erkennen von Wörtern

– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben





- Zustände: aus, ein

• Automaten: erkennen von Wörtern

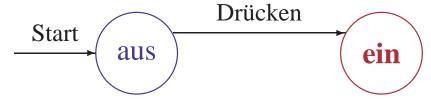
– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben



Zustände: aus, ein
 Startzustand: aus

• Automaten: erkennen von Wörtern

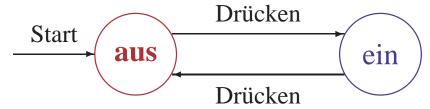
– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben



- Zustände: aus, ein Startzustand: aus
- Eingabesymbol: Drücken eines Schalters

• Automaten: erkennen von Wörtern

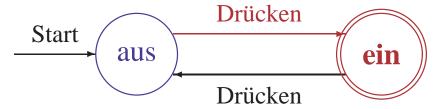
– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben



- Zustände: aus, ein
 Startzustand: aus
- Eingabesymbol: Drücken eines Schalters

• Automaten: erkennen von Wörtern

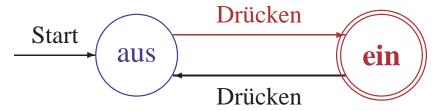
– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben



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

• Automaten: erkennen von Wörtern

– z.B. Wechselschalter: Verarbeitung von "Drück"-Eingaben



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

• Mathematische Mengennotation

 $-z.B.: \{w \in \{Dr\ddot{u}cken\}^* | \exists i \in \mathbb{N}. |w| = 2i+1\}, kurz \{Dr\ddot{u}cken^{2i+1} | i \in \mathbb{N}\}$

• Reguläre Ausdrücke: algebraische Strukturen

– z.B.: (DrückenDrücken)*Drücken

• Grammatiken: Vorschriften für Spracherzeugung

- $-z.B.: S \rightarrow Drücken, S \rightarrow SDrückenDrücken$
- Erzeugt nur ungerade Anzahl von Drücken-Symbolen

Theoretische Informatik I

Einheit 2.1



Deterministische Endliche Automaten



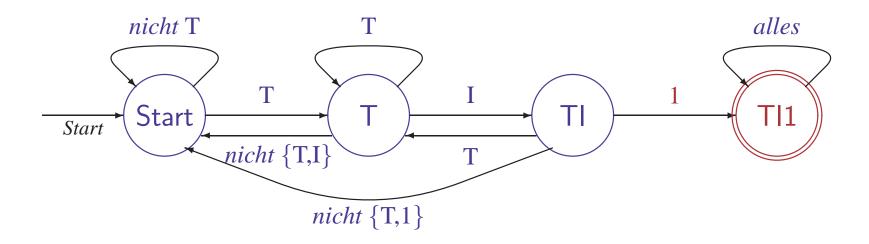
- 1. Arbeitsweise
- 2. Akzeptierte Sprache
- 3. Entwurf und Analyse
- 4. Automaten mit Ausgabe

Erkennung von Wörtern mit Automaten



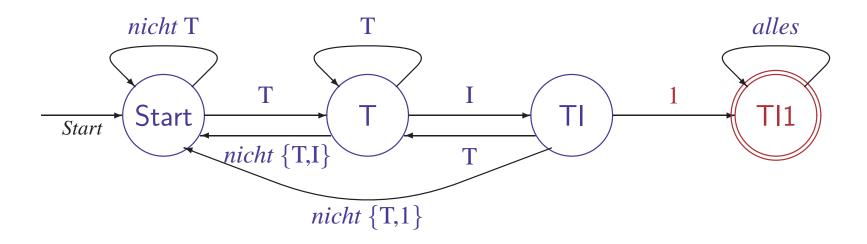
- Endliche Anzahl von Zuständen
- Ein Startzustand

Erkennung von Wörtern mit Automaten



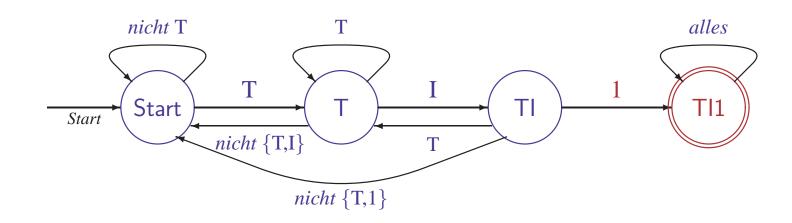
- Endliche Anzahl von Zuständen
- Ein Startzustand
- Regeln für Zustandsübergänge

Erkennung von Wörtern mit Automaten



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

Endliche Automaten – mathematisch präzisiert



Ein Deterministischer Endlicher Automat (DEA)

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

- Q nichtleere endliche **Zustandsmenge**
- $\bullet \Sigma$ (endliches) **Eingabealphabet**
- $\delta: Q \times \Sigma \to Q$ Zustandsüberführungsfunktion
- $q_0 \in Q$ Startzustand

• $F \subseteq Q$ Menge von akzeptierenden Zuständen

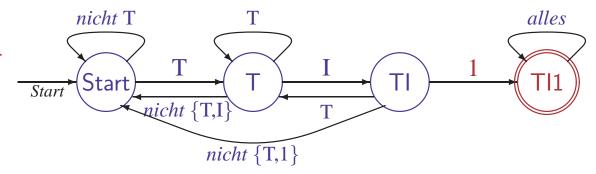
(Anfangszustand)

(Endzustände)

(Finale Zustände)

Beschreibung von Endlichen Automaten

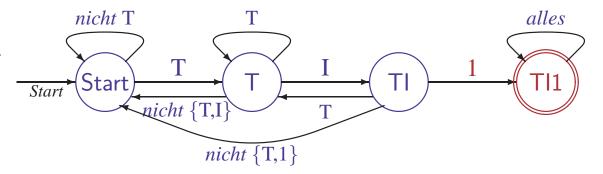
• Übergangsdiagramm



- 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

Beschreibung von Endlichen Automaten

• Übergangsdiagramm

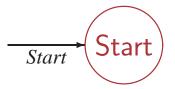


- Jeder Zustand in Q wird durch einen Knoten (Kreise) dargestellt
- Ist $\delta(q, a) = p$, so verläuft eine Kante von q nach p mit Beschriftung a (mehrere Beschriftungen derselben Kante möglich)
- $-q_0$ wird durch einen mit *Start* beschrifteten Pfeil angezeigt
- Endzustände in F werden durch doppelte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

• Übergangstabelle

- Tabellarische Darstellung der Funktion δ
- Kennzeichnung von q_0 durch einen Pfeil
- Kennzeichnung von F durch Sterne
- $-\Sigma$ und Q meist implizit durch Tabelle bestimmt

					sonst
\rightarrow	S	Т	S	S	S
	T	T	Ι	S	S
	Ι	T T T	S	1	S
*	1	1	1	1	1









• Anfangssituation

– Automat befindet sich im Startzustand $q_{\scriptscriptstyle 0}$

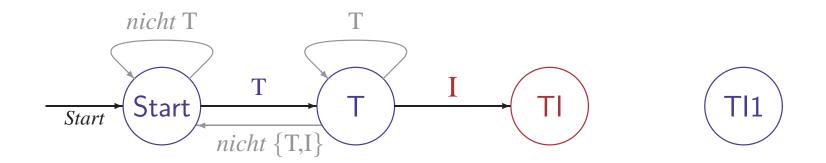


Anfangssituation

– Automat befindet sich im Startzustand q_0

Arbeitschritt

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

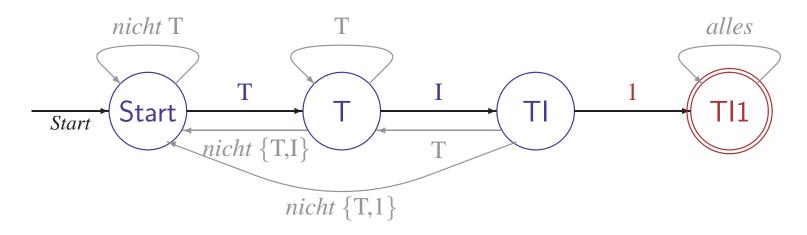


Anfangssituation

– Automat befindet sich im Startzustand q_0

• Arbeitschritt

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

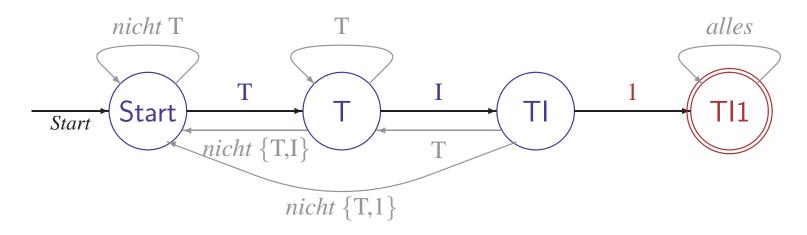


Anfangssituation

– Automat befindet sich im Startzustand q_0

• Arbeitschritt

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



Anfangssituation

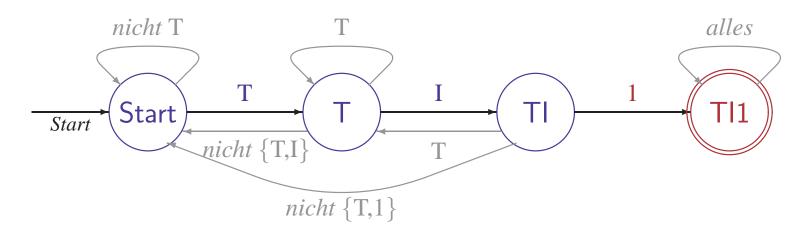
– Automat befindet sich im Startzustand q_0

Arbeitschritt

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

• Terminierung

– Eingabewort $w = a_1..a_n$ ist komplett gelesen, Automat im Zustand q_n



Anfangssituation

– Automat befindet sich im Startzustand q_0

Arbeitschritt

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

Terminierung

- Eingabewort $w = a_1..a_n$ ist komplett gelesen, Automat im Zustand q_n

• Ergebnis

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

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

• Erweiterte Überführungsfunktion $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$

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

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

• Erweiterte Überführungsfunktion $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$

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

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

• Von A akzeptierte Sprache

– Menge der Eingabewörter w, für die $\hat{\delta}(q_0,w)$ akzeptierender Zustand ist

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

– Auch: die von A erkannte Sprache

• Erweiterte Überführungsfunktion $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$

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

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

• Von A akzeptierte Sprache

– Menge der Eingabewörter w, für die $\hat{\delta}(q_0,w)$ akzeptierender Zustand ist

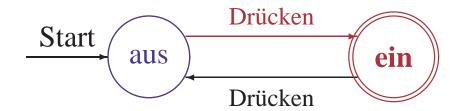
$$oldsymbol{L(A)} = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

– Auch: die von A erkannte Sprache

• Reguläre Sprache

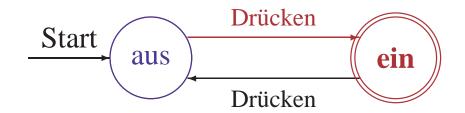
- Sprache, die von einem DEA A akzeptiert wird

Analyse der Sprache des Wechselschalters



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

Analyse der Sprache des Wechselschalters



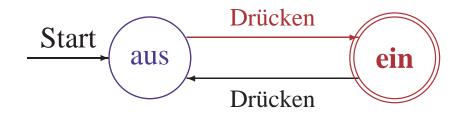
- Sprache: Eingaben, für die Automat eingeschaltet ist
 - Teilmenge der Wörter über dem Alphabet $\Sigma = \{Drücken\}$
- Automat A ist ein Wechselschalter

Nach n-fachem Drücken ist A aus(ein)geschaltet, wenn n (un)gerade ist Zwei Eigenschaften sind zu zeigen:

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

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

Analyse der Sprache des Wechselschalters



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

Nach n-fachem Drücken ist A aus(ein)geschaltet, wenn n (un)gerade ist Zwei Eigenschaften sind zu zeigen:

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

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

ullet Formale Beschreibung der Sprache von A

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

DETAILLIERTER INDUKTIONSBEWEIS



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

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

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

Induktions an fang n=0:

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

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

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

Induktionsschritt: Es sei n = m+1.

```
S_1(n): Es ist n = m+1 gerade
```

 $\Leftrightarrow m$ ist ungerade

 $\Leftrightarrow \hat{\delta}(\text{aus, Drücken}^m) = \text{ein} \quad (\text{Induktionsannahme } S_2(m))$

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

 $S_2(n)$: Es ist n = m+1 ungerade

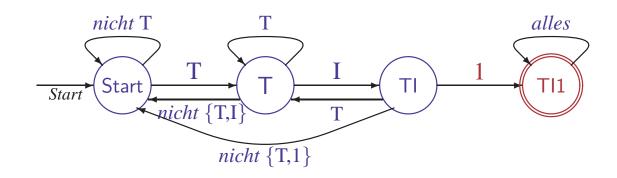
 $\Leftrightarrow m$ ist gerade

 $\Leftrightarrow \hat{\delta}(\text{aus, Drücken}^m) = \text{aus} \quad (\text{Induktionsannahme } S_1(m))$

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

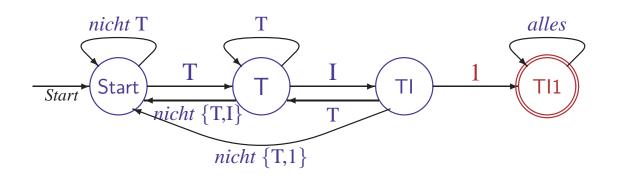
Aufgrund des Induktionsprinzips gilt $S_1(n)$ und $S_2(n)$ für alle $n \in \mathbb{N}$

Analyse der Sprache des "TI1"-Automaten



• Sprache: Eingaben, die den Zustand Tl1 erreichen

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



• Sprache: Eingaben, die den Zustand Tl1 erreichen

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

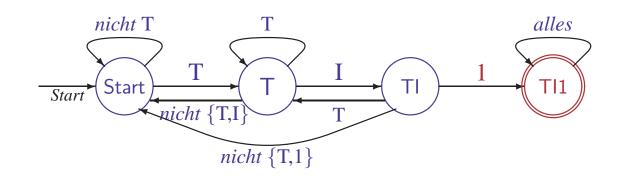
• Beweis zeigt, welche Wörter welchen Zustand erreichen

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

$$-S_3(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \iff \exists u \in \Sigma^*. \ w = u \mathsf{TI}$

$$-S_2(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \iff \exists u \in \Sigma^*. \ w = u\mathsf{T}$

$$-S_1(w)$$
: $\hat{\delta}(\text{Start}, w)$ =Start in allen anderen Fällen



• Sprache: Eingaben, die den Zustand Tl1 erreichen

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

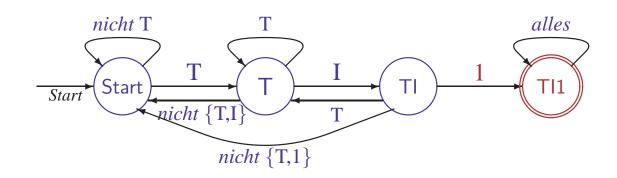
• Beweis zeigt, welche Wörter welchen Zustand erreichen

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

$$-S_3(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \iff \exists u \in \Sigma^*. \ w = u \mathsf{TI}$

$$-S_2(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \iff \exists u \in \Sigma^*. \ w = u\mathsf{T}$

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



• Sprache: Eingaben, die den Zustand Tl1 erreichen

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

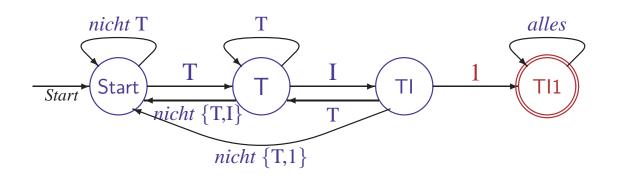
• Beweis zeigt, welche Wörter welchen Zustand erreichen

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

$$-S_3(w)$$
: $\hat{\delta}(\text{Start}, w) = \text{TI} \Leftrightarrow \exists u \in \Sigma^*. \ w = u \text{TI} \text{ fehlt da nicht etwas ?}$

$$-S_2(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \iff \exists u \in \Sigma^*. \ w = u\mathsf{T}$

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



• Sprache: Eingaben, die den Zustand Tl1 erreichen

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

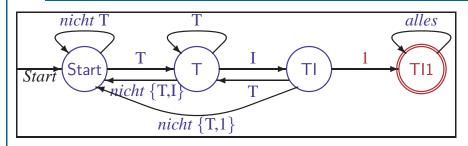
• Beweis zeigt, welche Wörter welchen Zustand erreichen

Beweise drei Eigenschaften durch simultane Induktion: (nächste Folie)

$$-S_3(w)$$
: $\hat{\delta}(\mathsf{Start}, w) = \mathsf{TI}$ $\Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \land \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$

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

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



Beweise durch simultane (strukturelle) Induktion:

$$S_3(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_2(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{T} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_1(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land w \neq u \mathsf{TI} \land w \neq u \mathsf{T}$$

Induktions an fang $w=\epsilon$:

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

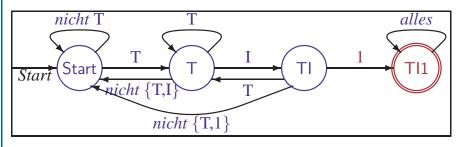
v

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

 $\sqrt{}$

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

1/



Beweise durch simultane (strukturelle) Induktion:

$$S_3(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_2(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{T} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_1(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land w \neq u \mathsf{TI} \land w \neq u \mathsf{T}$$

Induktions an fang $w=\epsilon$:

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

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

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

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

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

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

$$\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a \neq \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a \not\in \{\mathsf{T}, \mathsf{I}\}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ a \not\in \{\mathsf{T}, \mathsf{1}\})$$

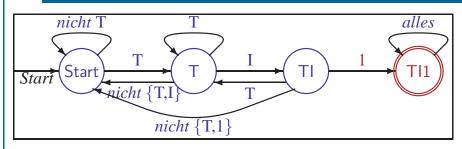
$$\Leftrightarrow \forall u, v \in \Sigma^*. \ w' \neq u \mathsf{TI1} v \land w' \neq u \mathsf{TI} \land w' \neq u \mathsf{T} \land a \neq \mathsf{T})$$

$$\forall \exists u \in \Sigma^*. \ w = u \mathsf{T} \land \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI1} v \land a \notin \{\mathsf{T}, \mathsf{I}\})$$
 $S_2(w')$

$$\forall \exists u \in \Sigma^*. \ w = u \mathsf{TI} \land \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land a \notin \{\mathsf{T}, \mathsf{1}\})$$
 $S_3(w')$

$$\Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \texttt{TI1} v \land w \neq u \texttt{TI} \land w \neq u \texttt{T}$$

 $S_1(w')$



Beweise durch simultane (strukturelle) Induktion:

$$S_3(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_2(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{T} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_1(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land w \neq u \mathsf{TI} \land w \neq u \mathsf{T}$$

Induktions an fang $w=\epsilon$:

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

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

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

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

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

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

$$\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a \neq \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a \not\in \{\mathsf{T}, \mathsf{I}\}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ a \not\in \{\mathsf{T}, \mathsf{1}\})$$

$$\Leftrightarrow \forall u, v \in \Sigma^*. \ w' \neq u \texttt{TI1} v \land w' \neq u \texttt{TI} \land w' \neq u \texttt{T} \land a \neq \texttt{T})$$

$$S_1(w')$$

$$\vee \exists u \in \Sigma^*. \ w=uT \land \forall u, v \in \Sigma^*. \ w\neq uTI1v \land a \notin \{T, I\})$$

$$S_2(w')$$

$$\vee \exists u \in \Sigma^*. \ w=u \text{TI} \wedge \forall u, v \in \Sigma^*. \ w\neq u \text{TI1} v \wedge a \notin \{\text{T}, \text{1}\})$$

$$S_3(w')$$

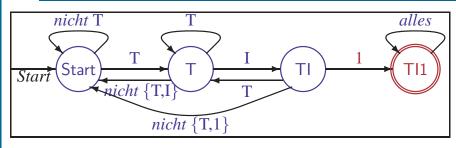
$$\Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \texttt{TI1} v \land w \neq u \texttt{TI} \land w \neq u \texttt{T}$$

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

$$\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ aa = \mathsf{T})$$

$$\Leftrightarrow \exists u \in \Sigma^*. \ w=u\mathtt{T} \land \forall u, v \in \Sigma^*. \ w\neq u\mathtt{TI1}v$$

mit
$$S_1(w'), S_2(w'), S_3(w')$$



Beweise durch simultane (strukturelle) Induktion:

$$S_3(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_2(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{T} \ \land \ \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_1(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land w \neq u \mathsf{TI} \land w \neq u \mathsf{T}$$

Induktions an fang $w=\epsilon$:

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

•

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

v

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

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

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

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

 $\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a \neq \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a \not\in \{\mathsf{T}, \mathsf{I}\}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ a \not\in \{\mathsf{T}, \mathsf{1}\})$

 $\Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \texttt{TI1} v \land w \neq u \texttt{TI} \land w \neq u \texttt{T}$

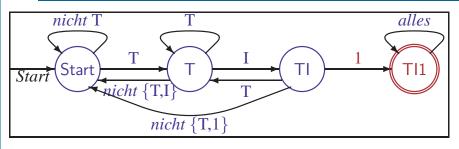
mit $S_1(w'), S_2(w'), S_3(w')$

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

 $\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ aa = \mathsf{T})$

 $\Leftrightarrow \exists u \in \Sigma^*. \ w=u \mathsf{T} \land \forall u, v \in \Sigma^*. \ w\neq u \mathsf{TI1}v$

mit $S_1(w'), S_2(w'), S_3(w')$



Beweise durch simultane (strukturelle) Induktion:

$$S_3(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{TI} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{TI} \land \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_2(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{T} \quad \Leftrightarrow \exists u \in \Sigma^*. \ w = u \mathsf{T} \land \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v$$

$$S_1(w) \colon \hat{\delta}(\mathsf{Start}, w) = \mathsf{Start} \Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \mathsf{TI} \mathsf{1} v \land w \neq u \mathsf{TI} \land w \neq u \mathsf{T}$$

Induktions an fang $w=\epsilon$:

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

 \checkmark

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

 $\sqrt{}$

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

 $\sqrt{}$

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

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

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

$$\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a \neq \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a \not\in \{\mathsf{T}, \mathsf{I}\}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ a \not\in \{\mathsf{T}, \mathsf{1}\})$$

$$\Leftrightarrow \forall u, v \in \Sigma^*. \ w \neq u \texttt{TI1} v \land w \neq u \texttt{TI} \land w \neq u \texttt{T}$$

mit
$$S_1(w'), S_2(w'), S_3(w')$$

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

$$\Leftrightarrow (\hat{\delta}(\mathsf{Start}, w') = \mathsf{Start} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{T} \ \land \ a = \mathsf{T}) \ \lor \ (\hat{\delta}(\mathsf{Start}, w') = \mathsf{TI} \ \land \ aa = \mathsf{T})$$

$$\Leftrightarrow \exists u \in \Sigma^*. \ w=u \mathsf{T} \land \forall u, v \in \Sigma^*. \ w\neq u \mathsf{TI1}v$$

mit
$$S_1(w'), S_2(w'), S_3(w')$$

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

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

$$\Leftrightarrow \exists u \in \Sigma^*. \ w = u \text{TI} \land \forall u, v \in \Sigma^*. \ w \neq u \text{TI1}v$$

$$\mathsf{mit}\ S_2(w')$$

 $\sqrt{}$

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

Entwurf und Analyse endlicher Automaten

Erkenne, ob eine Binärzahl durch 3 teilbar ist

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

Entwurf und Analyse endlicher Automaten

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 $n \mod 3 = 0$ (Divisionsrest)
 - Wir wissen $2i+a \mod 3 = 2(i \mod 3) + a \mod 3$

Entwurf und Analyse endlicher Automaten

Erkenne, ob eine Binärzahl durch 3 teilbar ist

• Binärzahl wird von links nach rechts gelesen

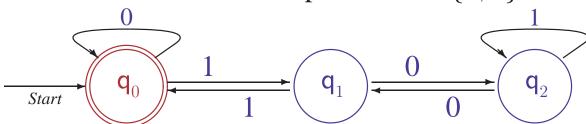
– Eingabewort $w=w_0..w_n$ entspricht der Zahl $n=r_b(w)=\sum_{j=0}^n w_j\cdot 2^{n-j}$

• Konstruktion des Automaten aus induktiver Beweisidee

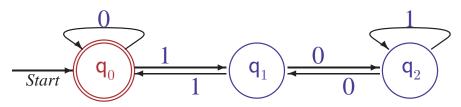
- Das Eingabewort $w = \epsilon$ entspricht der Zahl n = 0
- Entspricht v der Zahl i, dann entspricht w = va der Zahl n = 2i + a
- Die Zahl n ist durch 3 teilbar wenn $n \mod 3 = 0$ (Divisionsrest)
- Wir wissen $2i+a \mod 3 = 2(i \mod 3) + a \mod 3$

• Drei Zustände sind erforderlich

- Zustand $q_i \triangleq \text{bisher gelesene Binärzahl hat Divisionsrest } i \pmod{3}$
- Zustandsübergänge erhalten "Bedeutung": $\delta(q_i, a) = q_{2i+a \mod 3}$
- Resultierender DEA über Alphabet $\Sigma = \{0, 1\}$

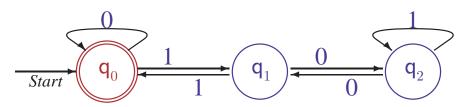


ZEIGE
$$L(A) = \{ w \mid r_b(w) \mod 3 = 0 \}$$



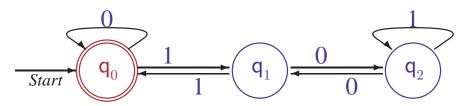
- ullet Zeige durch simultane strukturelle Induktion über w
 - $-S_j(w)$: $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \mod 3 = j$ für $j \in \{0, 1, 2\}$

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



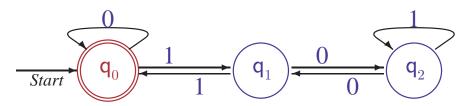
- ullet Zeige durch simultane strukturelle Induktion über w
 - $-S_j(w)$: $\hat{\delta}(q_0, w) = q_j \Leftrightarrow r_b(w) \mod 3 = j$ für $j \in \{0, 1, 2\}$
- Induktions an fang $w=\epsilon$:
 - Es ist $\hat{\delta}(q_0, \epsilon) = q_0$ und $r_b(\epsilon) \mod 3 = 0$
 - Damit gilt $S_0(w)$ und auch $S_1(w)$ und $S_2(w)$ (beide Seiten von \Leftrightarrow sind falsch) \checkmark

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



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

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



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

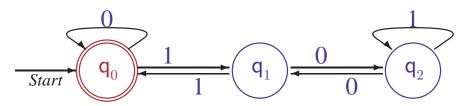
$$\hat{\delta}(q_0, w) = q_j$$

$$\Leftrightarrow \exists i.\hat{\delta}(q_0, w') = q_i \land \delta(q_i, a) = q_j$$

$$\Leftrightarrow \exists i.r_b(w') \bmod 3 = i \land j = 2i + a \bmod 3$$

$$\Leftrightarrow r_b(w) \mod 3 = 2r_b(w') + a \mod 3 = 2(r_b(w') \mod 3) + a \mod 3 = j$$

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



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

$$\hat{\delta}(q_0, w) = q_j$$

$$\Leftrightarrow \exists i.\hat{\delta}(q_0, w') = q_i \land \delta(q_i, a) = q_j$$

$$\Leftrightarrow \exists i.r_b(w') \bmod 3 = i \land j = 2i + a \bmod 3$$

$$\Leftrightarrow r_b(w) \mod 3 = 2r_b(w') + a \mod 3 = 2(r_b(w') \mod 3) + a \mod 3 = j$$

ullet Damit gilt $S_j(w)$ für alle und es folgt $w\in \Sigma^*$ und $j\in \{0,1,2\}$

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

ALTERNATIVE BESCHREIBUNG DER ARBEITSWEISE VON DEAS

• Konfiguration: 'Gesamtzustand' von Automaten

- Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
- Formal dargestellt als Tupel $\mathbf{K} = (\mathbf{q}, \mathbf{w}) \in Q \times \Sigma^*$

ALTERNATIVE BESCHREIBUNG DER ARBEITSWEISE VON DEAS

• Konfiguration: 'Gesamtzustand' von Automaten

- Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
- Formal dargestellt als Tupel $\mathbf{K} = (\mathbf{q}, \mathbf{w}) \in Q \times \Sigma^*$

• Konfigurationsübergangsrelation \vdash^*

- Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
- $-(q,aw) \vdash (p,w)$, falls $\delta(q,a) = p$
- $-\mathbf{K_1} \vdash^* \mathbf{K_2}$, falls $K_1 = K_2$ oder es gibt eine Konfiguration K mit $K_1 \vdash K$ und $K \vdash^* K_2$

ALTERNATIVE BESCHREIBUNG DER ARBEITSWEISE VON DEAS

• Konfiguration: 'Gesamtzustand' von Automaten

- Mehr als $q \in Q$: auch die noch unverarbeitete Eingabe zählt
- Formal dargestellt als Tupel $\mathbf{K} = (\mathbf{q}, \mathbf{w}) \in Q \times \Sigma^*$

• Konfigurationsübergangsrelation \vdash^*

- Wechsel zwischen Konfigurationen durch Abarbeitung von Wörtern
- $-(q,aw) \vdash (p,w)$, falls $\delta(q,a) = p$
- $-\mathbf{K_1} \vdash^* \mathbf{K_2}$, 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 ⊢ zu akzeptierendem Zustand führt

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

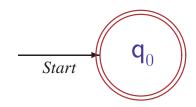
Für DEAs weniger intuitiv, aber leichter zu verallgemeinern

Codiere Anzahl der gelesener 0/1 im Zustand

 $q_0 = (gerade, gerade)$ $q_1 = (gerade, ungerade)$

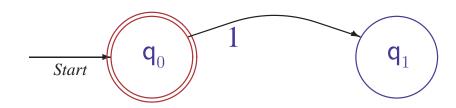
Codiere Anzahl der gelesener 0/1 im Zustand

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



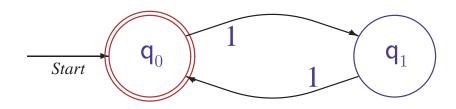
Codiere Anzahl der gelesener 0/1 im Zustand

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



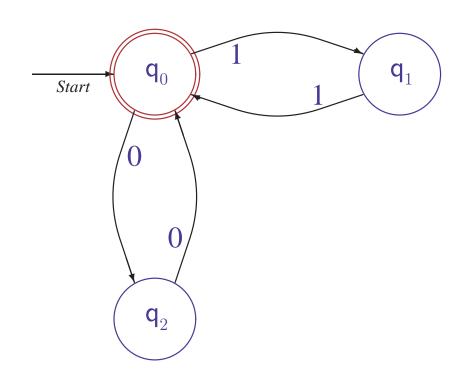
Codiere Anzahl der gelesener 0/1 im Zustand

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



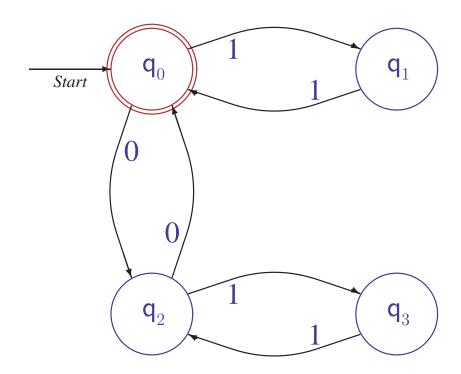
Codiere Anzahl der gelesener 0/1 im Zustand

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



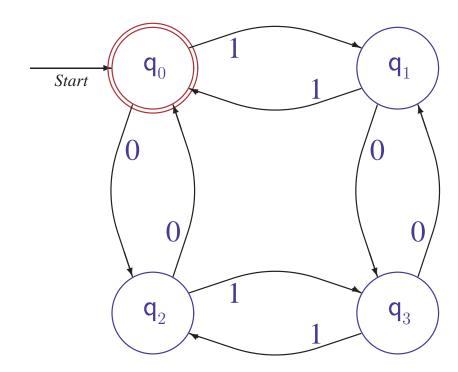
Codiere Anzahl der gelesener 0/1 im Zustand

 $q_0 = (gerade, gerade)$ $q_1 = (gerade, ungerade)$



Codiere Anzahl der gelesener 0/1 im Zustand

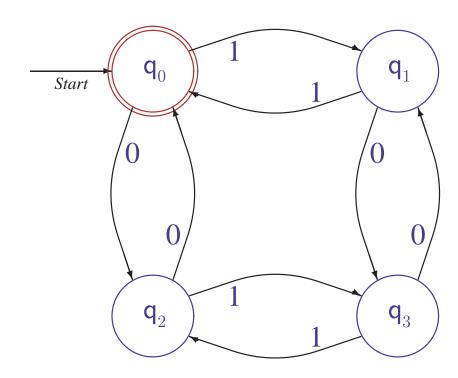
 $q_0 = (gerade, gerade)$ $q_1 = (gerade, ungerade)$



Codiere Anzahl der gelesener 0/1 im Zustand

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

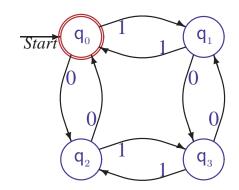
 $q_2 = (ungerade, gerade)$ $q_3 = (ungerade, ungerade)$



Korrektheit: gegenseitige strukturelle Induktion

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

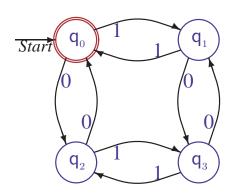
- (1) $(q_0, w v) \stackrel{*}{\vdash} (q_0, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } g_1(w)$
- (2) $(q_0, w v) \stackrel{*}{\vdash} (q_1, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } u_1(w)$
- (3) $(q_0, w v) \stackrel{*}{\vdash} (q_2, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } g_1(w)$
- (4) $(q_0, w v) \stackrel{*}{\vdash} (q_3, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } u_1(w)$



 $g_0(w) = w$ hat gerade Anzahl von Nullen, $u_0(w) = w$ hat ungerade Anzahl von Nullen, ...

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

- (1) $(q_0, w v) \stackrel{*}{\vdash} (q_0, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } g_1(w)$
- (2) $(q_0, w v) \stackrel{*}{\vdash} (q_1, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } u_1(w)$
- (3) $(q_0, w v) \stackrel{*}{\vdash} (q_2, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } g_1(w)$
- (4) $(q_0, w v) \stackrel{*}{\vdash} (q_3, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } u_1(w)$



 $g_0(w) = w$ hat gerade Anzahl von Nullen, $u_0(w) = w$ hat ungerade Anzahl von Nullen, ...

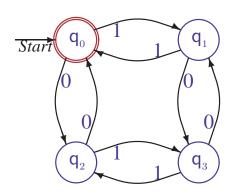
• Basisfall $w = \epsilon$:

- Per Definition gilt $(q_0, v) \stackrel{*}{\vdash} (q_0, v)$ und $g_0(w)$ und $g_1(w)$



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

- (1) $(q_0, w v) \stackrel{*}{\vdash} (q_0, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } g_1(w)$
- (2) $(q_0, w v) \stackrel{*}{\vdash} (q_1, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } u_1(w)$
- (3) $(q_0, w v) \stackrel{*}{\vdash} (q_2, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } g_1(w)$
- (4) $(q_0, w v) \stackrel{*}{\vdash} (q_3, v) \Leftrightarrow \text{es gilt } u_0(w) \text{ und } u_1(w)$



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

• Basisfall $w = \epsilon$:

- Per Definition gilt $(q_0, v) \stackrel{*}{\vdash} (q_0, v)$ und $g_0(w)$ und $g_1(w)$
- ullet Schrittfall w=ua für ein $u\in \Sigma^*, a\in \Sigma$:
 - (1) Es gelte $(q_0, w v) \vdash^* (q_0, v)$.

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

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

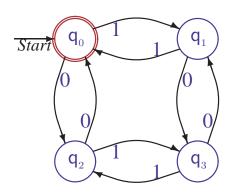
Falls a = 1, dann ist $p = q_1$ und nach (2) folgt $g_0(u)$ und $u_1(u)$.

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

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

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

- (1) $(q_0, w v) \stackrel{*}{\vdash} (q_0, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } g_1(w)$
- (2) $(q_0, w v) \stackrel{*}{\vdash} (q_1, v) \Leftrightarrow \text{ es gilt } g_0(w) \text{ und } u_1(w)$
- (3) $(q_0, w v) \stackrel{*}{\vdash} (q_2, v) \Leftrightarrow \text{ es gilt } u_0(w) \text{ und } g_1(w)$
- (4) $(q_0, w v) \stackrel{*}{\vdash} (q_3, v) \Leftrightarrow \text{es gilt } u_0(w) \text{ und } u_1(w)$



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

• Basisfall $w = \epsilon$:

- Per Definition gilt $(q_0, v) \stackrel{*}{\vdash} (q_0, v)$ und $g_0(w)$ und $g_1(w)$
- ullet Schrittfall w=ua für ein $u\in \Sigma^*, a\in \Sigma$:
 - (1) Es gelte $(q_0, w v) \vdash^* (q_0, v)$.

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

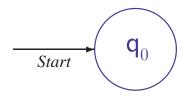
Falls a = 0, dann ist $p = q_2$ und nach (3) folgt $u_0(u)$ und $g_1(u)$.

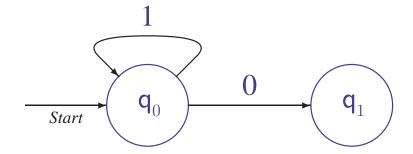
Falls a = 1, dann ist $p = q_1$ und nach (2) folgt $g_0(u)$ und $u_1(u)$.

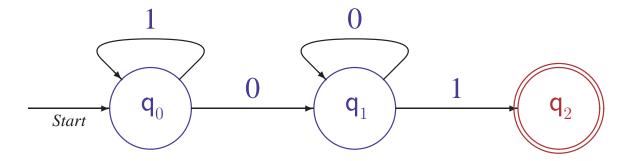
Für w = ua folgt somit $g_0(w)$ und $g_1(w)$.

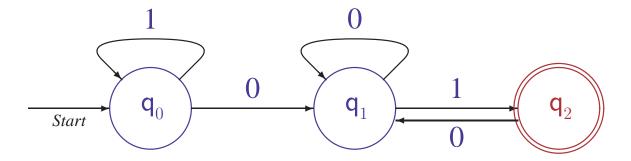
Gegenrichtung durch Umkehrung des Arguments. (2), (3), (4) analog.

 $\bullet \; \textbf{Es folgt} \;\; w \in L(A) \; \Leftrightarrow \; (q_0,w) \vdash^* (q_0,\epsilon) \; \Leftrightarrow \; g_0(w) \land g_1(w) \; \Leftrightarrow \; w \in L$



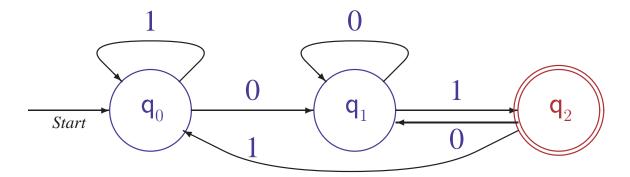






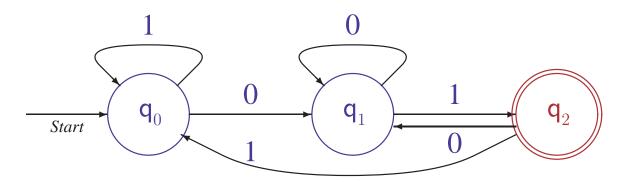
Weitere Beispiele endlicher Automaten

• Erkenne Strings, die mit 01 enden



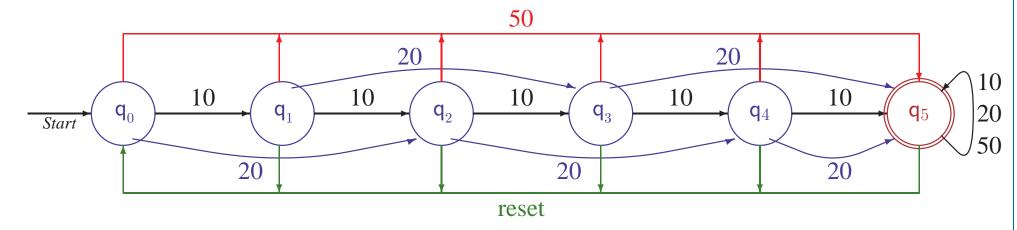
Weitere Beispiele endlicher Automaten

• Erkenne Strings, die mit 01 enden



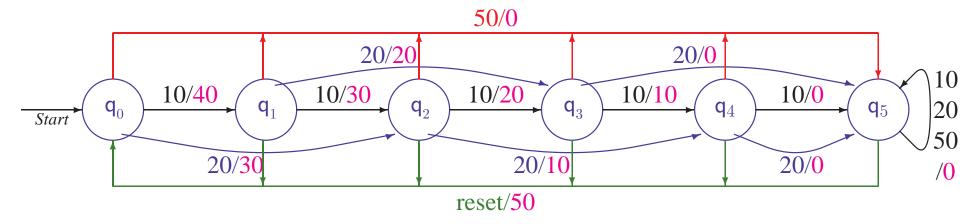
• 50c Kaffeeautomat

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



ENDLICHE AUTOMATEN MIT AUSGABEFUNKTION

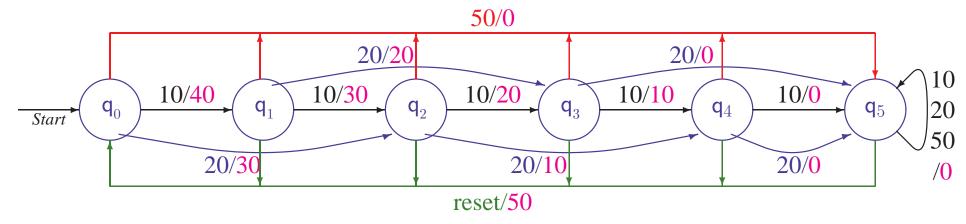
• 50c Kaffeeautomat mit Restbetragsanzeige



Münzeinwurf führt zu Zustandsänderung und erzeugt Ausgabe

ENDLICHE AUTOMATEN MIT AUSGABEFUNKTION

• 50c Kaffeeautomat mit Restbetragsanzeige



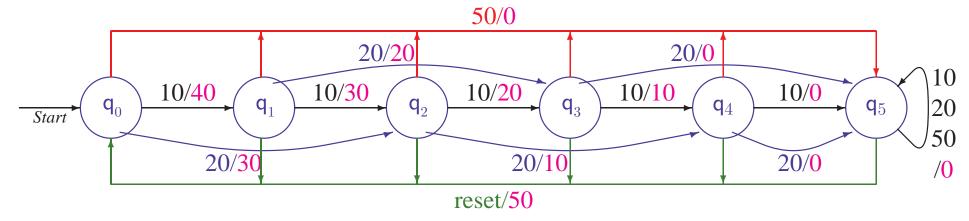
- Münzeinwurf führt zu Zustandsänderung und erzeugt Ausgabe

• 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

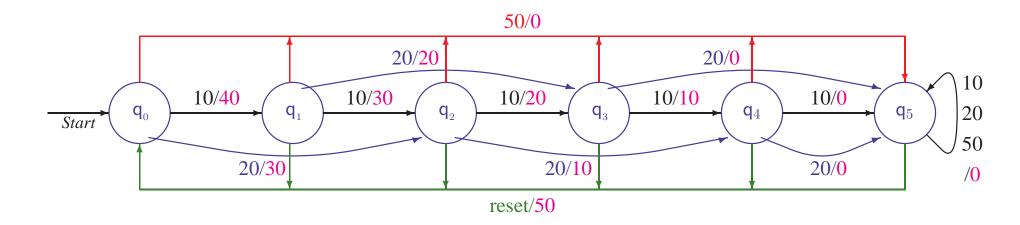
• Automaten mit Ausgabe sind keine echte Erweiterung

- 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**
- $\bullet \Sigma$ (endliches) **Eingabealphabet**
- $\bullet \Delta$ (endliches) **Ausgabealphabet**
- $\delta: Q \times \Sigma \to Q$ Zustandsüberführungsfunktion
- $\lambda: Q \times \Sigma \to \Delta$ Ausgabefunktion
- $q_0 \in Q$ Startzustand

Arbeitsweise von Mealy-Automaten analog zu Deas

- ullet **Anfangssituation:** Automat im Startzustand q_0
- Arbeitschritt
 - Im Zustand q lese Eingabesymbol a,
 - Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p
 - Bestimme $x = \lambda(q,a)$ und gebe dieses Symbol aus
- **Terminierung:** Eingabewort $w = a_1..a_n$ ist komplett gelesen
- Ausgabewort: Verkettung der ausgegebenen Symbole $x_1...x_n$
- Erweiterte Ausgabefunktion $\hat{\lambda}: Q \times \Sigma^* \to \Delta^*$
 - Schrittweise Erzeugung der Ausgabe mit Abarbeitung der Eingabe
 - Formal: Induktive Definition

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

ullet 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

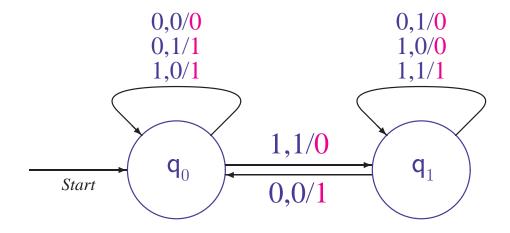
Addition von Bitpaaren von rechts nach links

- Eingabealphabet $\Sigma = \{0, 1\} \times \{0, 1\}$
- Ausgabealphabet $\Delta = \{0, 1\}$

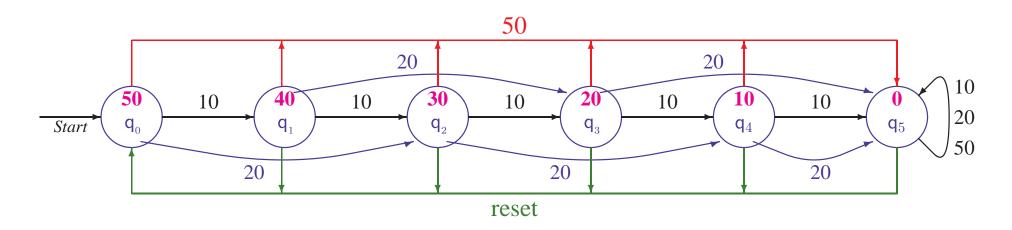
• Zwei Zustände sind erforderlich

- 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**
- $\bullet \Sigma$ (endliches) **Eingabealphabet**
- $\bullet \Delta$ (endliches) **Ausgabealphabet**
- $\delta: Q \times \Sigma \to Q$ Zustandsüberführungsfunktion
- $\lambda:Q \to \Delta$ Ausgabefunktion
- $q_0 \in Q$ Startzustand

Arbeitsweise von Moore-Automaten analog zu Deas

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

Arbeitsweise von Moore-Automaten analog zu Deas

- **Anfangssituation:** Automat im Startzustand q_0 , Ausgabe $x_0 = \lambda(q_0)$
- Arbeitschritt
 - Im Zustand q lese Eingabesymbol a,
 - Bestimme $\delta(q,a)=p$ und wechsele in neuen Zustand p
 - Bestimme $x = \lambda(p)$ und gebe dieses Symbol aus
- **Terminierung:** Eingabewort $w = a_1..a_n$ ist komplett gelesen
- Ausgabewort: Verkettung der ausgegebenen Symbole $x_0x_1...x_n$
- Erweiterte Ausgabefunktion $\hat{\lambda}: Q \times \Sigma^* \to \Delta^*$
 - Schrittweise Erzeugung der Ausgabe mit Abarbeitung der Eingabe
 - Formal: Induktive Definition

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

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

• Eingabe einer Bitfolge von links nach rechts

- Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl \boldsymbol{n}
- Ausgabe ist jeweils "n mod 3"
- Eingabealphabet $\Sigma = \{0, 1\}$, Ausgabealphabet $\Delta = \{0, 1, 2\}$

• Eingabe einer Bitfolge von links nach rechts

- Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl \boldsymbol{n}
- Ausgabe ist jeweils "n mod 3"
- Eingabealphabet $\Sigma = \{0, 1\}$, Ausgabealphabet $\Delta = \{0, 1, 2\}$

• Drei Zustände sind erforderlich

- Zustand q_0 : Bisheriger Divisionsrest ist 0 (Ausgabe 0)
- Zustand q_1 : Bisheriger Divisionsrest ist 1 (Ausgabe 1)
- Zustand q_2 : Bisheriger Divisionsrest ist 2 (Ausgabe 2)

• Eingabe einer Bitfolge von links nach rechts

- Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl \boldsymbol{n}
- Ausgabe ist jeweils "n mod 3"
- Eingabealphabet $\Sigma = \{0, 1\}$, Ausgabealphabet $\Delta = \{0, 1, 2\}$

Drei Zustände sind erforderlich

- Zustand q_0 : Bisheriger Divisionsrest ist 0 (Ausgabe 0)
- Zustand q_1 : Bisheriger Divisionsrest ist 1 (Ausgabe 1)
- Zustand q_2 : Bisheriger Divisionsrest ist 2 (Ausgabe 2)
- Zustandsüberführungsregel $\delta(q_i, j) = q_{2*i+j \mod 3}$

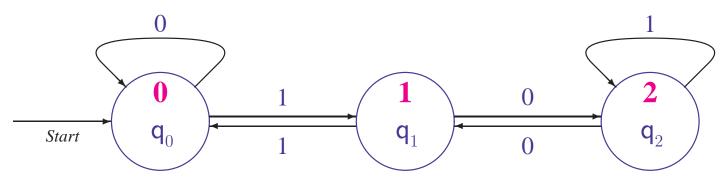
• Eingabe einer Bitfolge von links nach rechts

- Bisher eingegebene Bitfolge ist Binärdarstellung einer Zahl n
- Ausgabe ist jeweils "n mod 3"
- Eingabealphabet $\Sigma = \{0, 1\}$, Ausgabealphabet $\Delta = \{0, 1, 2\}$

• Drei Zustände sind erforderlich

- Zustand q_0 : Bisheriger Divisionsrest ist 0 (Ausgabe 0)
- Zustand q_1 : Bisheriger Divisionsrest ist 1 (Ausgabe 1)
- Zustand q_2 : Bisheriger Divisionsrest ist 2 (Ausgabe 2)
- Zustandsüberführungsregel $\delta(q_i, j) = q_{2*i+j \mod 3}$

Zugehöriger Moore-Automat



Moore-Automaten sind äquivalent zu DEAs

Gegenseitige Simulation ist möglich

ullet 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

- $\operatorname{graph}(f) = \{(w, v) \mid f(w) = v\}$
- $-\operatorname{graph}^*(f) = \{(w_1, v_0, v_1)..(w_n, v_n) \mid f(w_1..w_n) = v_0..v_n\}$
- DEAs können Graphen berechneter Funktionen akzeptieren

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

Beweis der Äquivalenz (Skizze)

- ullet L regulär $\Leftrightarrow \chi_L$ "Moore-berechenbar"
 - $-\operatorname{Zu} A = (Q, \Sigma, \delta, q_0, F) \text{ konstruiere } M = (Q, \Sigma, \{0,1\}, \delta, \lambda, q_0)$ $\operatorname{mit} \lambda(q) = \begin{cases} 1 & \text{falls } q \in F, \\ 0 & \text{sonst} \end{cases}$
 - 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)

ullet L regulär $\Leftrightarrow \chi_L$ "Moore-berechenbar"

- $-\operatorname{Zu} A = (Q, \Sigma, \delta, q_0, F) \text{ konstruiere } M = (Q, \Sigma, \{0,1\}, \delta, \lambda, q_0)$ $\operatorname{mit} \lambda(q) = \begin{cases} 1 & \text{falls } q \in F, \\ 0 & \text{sonst} \end{cases}$
- Dann ist $w\in L(A)$ genau dann, wenn $f_M(w)=v1$ für ein $v\in\{0,1\}^*$ $\chi_L(w)$ ist das letzte Ausgabesymbol von $f_M(w)$

• f Moore-berechenbar \Leftrightarrow graph*(f) regulär

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

$$\boldsymbol{\delta'(q,(a,b))} = \begin{cases} \delta(q_0,a) & \text{falls } q = q_s, b = (\lambda(q_0), \lambda(\delta(q_0,a)), \\ \delta(q,a) & \text{falls } \lambda(\delta(q,a)) = b, \\ q_f & \text{sonst} \end{cases}$$

- 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)

ullet L regulär $\Leftrightarrow \chi_L$ "Moore-berechenbar"

- $-\operatorname{Zu} A = (Q, \Sigma, \delta, q_0, F) \text{ konstruiere } M = (Q, \Sigma, \{0,1\}, \delta, \lambda, q_0)$ $\operatorname{mit} \lambda(q) = \begin{cases} 1 & \text{falls } q \in F, \\ 0 & \text{sonst} \end{cases}$
- Dann ist $w\in L(A)$ genau dann, wenn $f_M(w)=v1$ für ein $v\in\{0,1\}^*$ $\chi_L(w)$ ist das letzte Ausgabesymbol von $f_M(w)$

• f Moore-berechenbar \Leftrightarrow graph*(f) regulär

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

$$\boldsymbol{\delta'(q,(a,b))} = \begin{cases} \delta(q_0,a) & \text{falls } q = q_s, b = (\lambda(q_0), \lambda(\delta(q_0,a)), \\ \delta(q,a) & \text{falls } \lambda(\delta(q,a)) = b, \\ q_f & \text{sonst} \end{cases}$$

- 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