Automatentheorie und Formale Sprachen

– LV 4110 –

Endliche Automaten

- Kennenlernen der Begriffe: Automat, endlicher Automat, Modell, Zustandsmodell, Graph, Zustandsgraph, Sprache
- Definition der Begriffe: Eingabealphabet, Systemzustände, Start- und Endzustand, Zustandsüberführungsfunktion
- Verwendung von Beschreibungsformen: Zustandsgraph, Funktionstafel, Eingabefolgen, Sprache
- Deterministische endliche Automaten: Komponenten, Eigenschaften, Erweiterung, Sprache, Backus-Naur-Form, Beispiele
- Nicht-deterministische endliche Automaten: Definition, Sprache, Teilmengenkonstruktion und Überführung
- Äquivalenz und Minimierung von Automaten: Minimalautomat, Äquivalenzrelation und -klassen, Algorithmusbeschreibung, Beispiele

I. Endliche Automaten

Was sind Automaten?

Automaten sind selbständig (automatisch) arbeitende Maschinen, die auf gewisse Eingabesignale ihrer Umwelt in einer bestimmten Weise reagieren.

Beispiele:

- Fahrkartenautomat
- Waschmaschine
- Rechenanlage
- Fernsprechanlage
- usw.

Aufbau- und Funktionsbeschreibung:

- Beschreibung kann verbal, z. B. in Umgangssprache erfolgen.
- Für komplexere Prozesse werden abstrakte Modelle benötigt (→ mehrere Abstraktionsebenen).
- Das einfachste Modell stellt der sog. endliche Automat (**EA**) dar.
- Ein adäquates Mittel zur Funktionsbeschreibung des **EA** sind Zustandsgraphen.
- Daneben existieren noch weitere Beschreibungsformen, wie z. B. Funktionstafeln, Formale Grammatiken etc.

Relationen und Relationseigenschaften:

$A_1, A_2, \ldots, A_n$ seien Mengen

$x_1, x_2, \ldots, x_n$ seien Elemente mit $x_1 \in A_1, x_2 \in A_2, \ldots, x_n \in A_n$

$\Rightarrow$ Implikation (daraus folgt)

$\Leftrightarrow$ Äquivalenz (genau dann wenn)

$\in$ Element von

$\Rightarrow$ $(x_1, x_2, \ldots, x_n)$ ist ein geordnetes Tupel von Elementen über $A_1, A_2, \ldots, A_n$

Kartesisches Produkt:

$$A_1 \times A_2 \times \ldots \times A_n := \{ (x_1, x_2, \ldots, x_n) \mid x_1 \in A_1, x_2 \in A_2, \ldots, x_n \in A_n \}$$

n-stellige Relation:

Satz: Jede Teilmenge **R** der Mengen A_1 x A_2 x ... x A_n heißt eine n-stellige Relation über den Mengen A_1, A_2, ..., A_n.

kurz: $\mathbf{R} \subseteq A_1 \times A_2 \times \ldots \times A_n$

Binäre Relation:

$\Rightarrow$ n = 2 (auch: 2-stellige Relation)

Anmerkung:

Im folgenden sind vor allem binäre Relationen der Art $\mathbf{R} \subseteq \mathbf{M} \times \mathbf{M}$ von Interesse $\Rightarrow$ **R** heißt dann „Relation auf **M**" oder „Relation in **M**".

Anmerkung (Fortsetzung):

$\Leftrightarrow (x, y) \in \mathbf{R} \quad \Leftrightarrow$ sog. Infixnotation für $x\ \mathbf{R}\ y$

heißt: „x steht in der Relation y“

Definition (symmetrisch, reflexiv, transitiv):

Eine binäre Relation **R** auf einer Menge **M** heißt:

symmetrisch $\Leftrightarrow \quad \forall x, y \in \mathbf{M}: \quad (x\ \mathbf{R}\ y \Rightarrow y\ \mathbf{R}\ x)$

reflexiv $\Leftrightarrow \quad \forall x \in \mathbf{M}: \quad x\ \mathbf{R}\ x$

transitiv $\Leftrightarrow \quad \forall x, y, z \in \mathbf{M}: \quad (x\ \mathbf{R}\ y \wedge y\ \mathbf{R}\ z \Rightarrow x\ \mathbf{R}\ z)$

Satz: Eine Relation R auf einer Menge M heißt **Äquivalenzrelation**, wenn sie symmetrisch, reflexiv und transitiv ist.

<u>Beispiel:</u>

Wir betrachten die Teilbarkeitsrelation „|“ über den ganzen Zahlen **Z**.
Diese Relation ist für $\forall$ m, n $\in$ **Z** definiert durch:

$$(\, m \mid n \;\Leftrightarrow\; \exists\, k \in \mathbf{Z} : n = k \cdot m \,)$$

$m \mid n$ heißt: m teilt n.

<u>Zahlenbeispiel:</u>

$7 \mid 21$, denn: $21 = 3 \cdot 7$

<u>Satz:</u> **Die Teilbarkeitsrelation „|“ ist zwar reflexiv und transitiv, aber <u>nicht</u> symmetrisch ⇒ ist demnach <u>keine</u> Äquivalenzrelation!**

I. Endliche Automaten

1. Sprachgebrauch und Motivation
   - **1.1 Automaten und Zustandsüberführungsfunktion**
   - 1.2 Sprache eines Automaten
2. Deterministische endliche Automaten
   - 2.1 Erweiterung der Überführungsfunktion
   - 2.2 Sprache eines deterministischen Automaten
3. Nicht-deterministische endliche Automaten
   - 3.1 Sprache eines nicht-deterministischen Automaten
   - 3.2 Teilmengenkonstruktion
4. Äquivalenz und Minimierung von Automaten
   - 4.1 Äquivalente und reduzierte Automaten
   - 4.2 Bildung von Äquivalenzklassen

Zustandsgraphen

Ein Zustandsgraph besteht aus Knoten und Kanten:

S_i = Knote $\mathbf{S}_i$ kennzeichnet den Systemzustand $\mathbf{S}_i$

$\xrightarrow{e}$ = Kante $\mathbf{e}$ drückt den Zustandsübergang unter Einwirkung der Eingabe $\mathbf{e}$ aus.

Kennzeichnung des Start- und Endzustandes:

$\rightarrow S_0$ = Startzustand $\mathbf{S}_0$

S_f = Endzustand $\mathbf{S}_f$

An einem Graphen lässt sich sehr leicht nachverfolgen, welche Systemzustände bei der Verarbeitung der Eingabezeichen angenommen werden.

Automat als informationsverarbeitende Maschine:

Eingabe: dient zur Versorgung des Automaten mit Eingabedaten e_i aus den sog. Eingabealphabet $\mathbf{\Sigma}$ ($e_i \in \mathbf{\Sigma}$)

Interne Zustände: werden als Reaktion auf die Eingabe durchlaufen ($\mathbf{S} = \{s_0, s_1, \ldots, s_n\}$)

Ausgabe: sind die vom Automaten i.d.R. produzierten Ausgabedaten ($a_i \in \mathbf{A}$) mit $\mathbf{A}$ = Ausgabealphabet

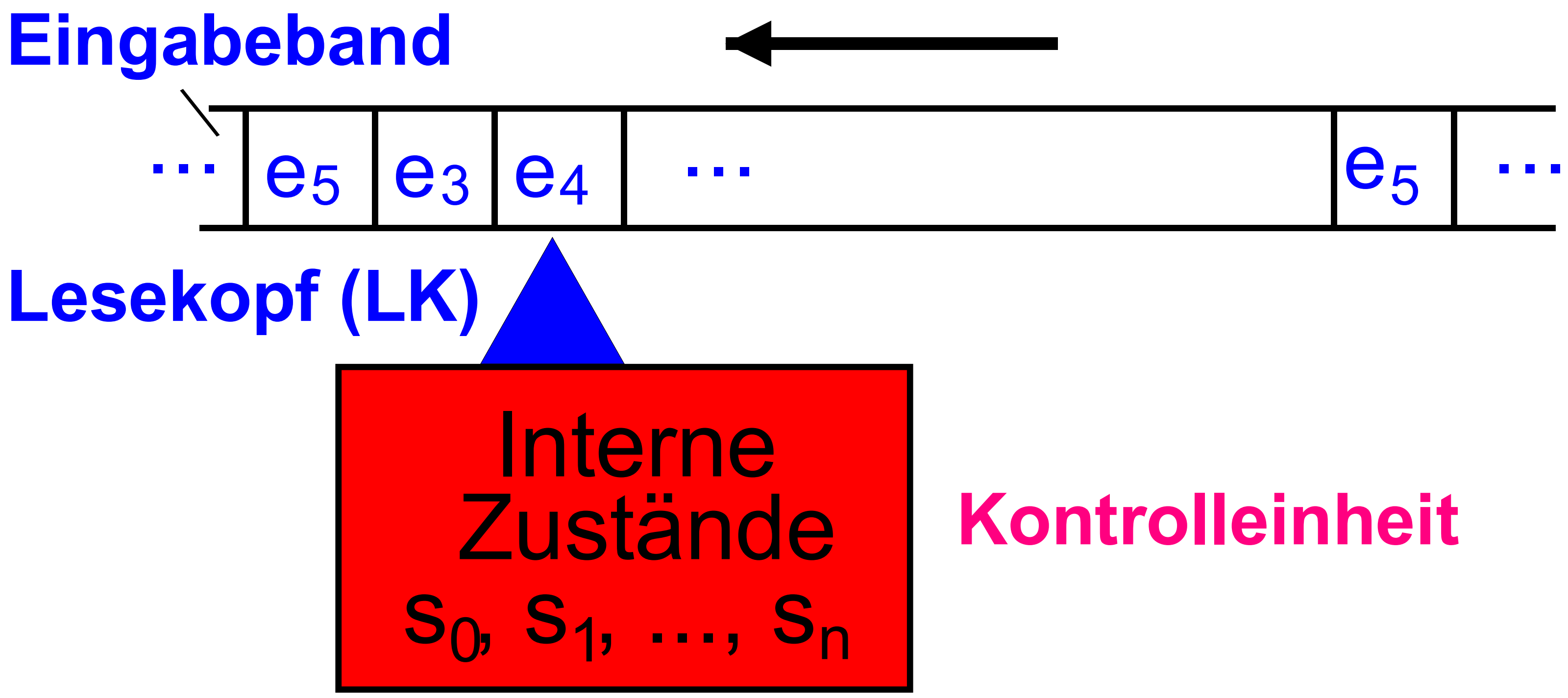
Charakteristisch für einen Automaten ist, dass der Folgezustand neben den Eingabezeichen auch vom momentanen inneren Systemzustand abhängig ist.

Die Zustands-Überführungsfunktion δ muss daher in folgender Form angeschrieben werden:

$$\delta : S \times \Sigma \rightarrow S$$

x := kartesische Produkt

Anschauliche Vorstellung eines endlichen Automaten:



Arbeitsweise eines endlichen Automaten:

BEGIN

Bringe EA *in Zustand* s_0; (* Anfangszustand *)

Setze LK *über linkes Zeichen des Eingabewortes*;

WHILE *Zeichen unter* LK *vorhanden* **DO**

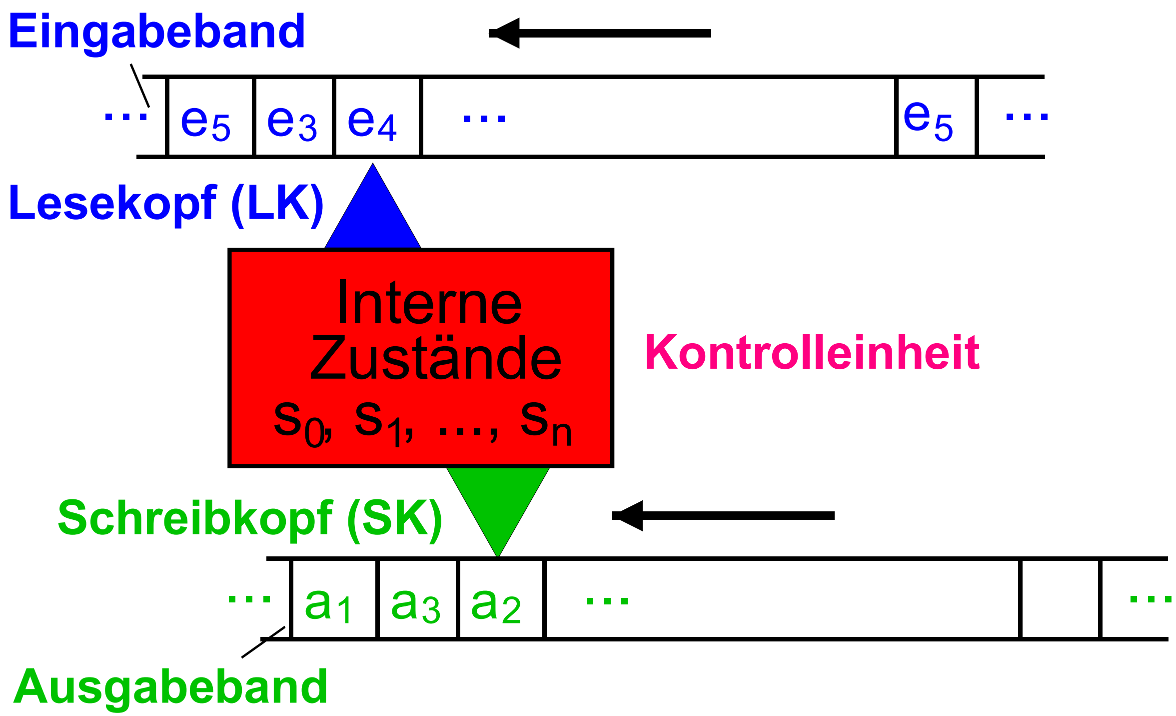
Gehe in Folgezustand gemäß $\boldsymbol{\delta : S \times \Sigma \to S}$;

Bewege LK *um ein Feld nach rechts*;

END (* WHILE *)

END

Vorstellung eines endlichen Automaten mit Ausgabe:



Arbeitsweise eines endlichen Automaten mit Ausgabe:

BEGIN

Bringe EA *in Zustand* s_0; (* Anfangszustand *)

Setze LK *über linkes Zeichen des Eingabewortes*;

WHILE *Zeichen unter* LK *vorhanden* **DO**

Schreibe das gewünschte Ausgabezeichen $a_i \in \mathbf{A}$

Gehe in Folgezustand gemäß $\boldsymbol{\delta : S \times \Sigma \rightarrow S}$;

Bewege LK *um ein Feld nach rechts*;

END (* WHILE *)

END

Definition:

Die Menge aller Eingabefolgen, die von einem Automaten **A** akzeptiert werden, nennt man die Sprache **T** des Automaten.

kurz:

$$T(A)$$

Beispiel:

mit $\mathbf{\Sigma = \{a, b, c\}}$

$$\Rightarrow T(A) = \mathbf{abcabcabc} \ldots$$
$$= (\mathbf{abc})^*$$

Beispiel

Spezifikation:

- Einwurfsmöglichkeiten sind **1** €-, **2** €- und **5** €-Münzen.
- In jeder Situation kann der Automat durch Drücken des Rückgabeknopfes (**R**) in den Anfangszustand versetzt werden. (Bereits eingeworfene Münzen werden dann zurückgegeben.)
- Falls **5** € in den Automaten eingeworfen wurden, wird durch Ziehen einer Schublade (**Z**) die Ware zur Entnahme freigegeben.
- Mit der Entnahme (**E**) der Ware wird der Anfangszustand wieder hergestellt.
- Der geleistete Münzeinwurf wird durch die Anzeige (**A**) angezeigt.

Funktionsweise:

Die Funktionsweise des Warenautomaten kann mit Hilfe der **Eingabezeichen**, einer Reihe von **Zuständen** (states) und der **Ausgabe** beschrieben werden.

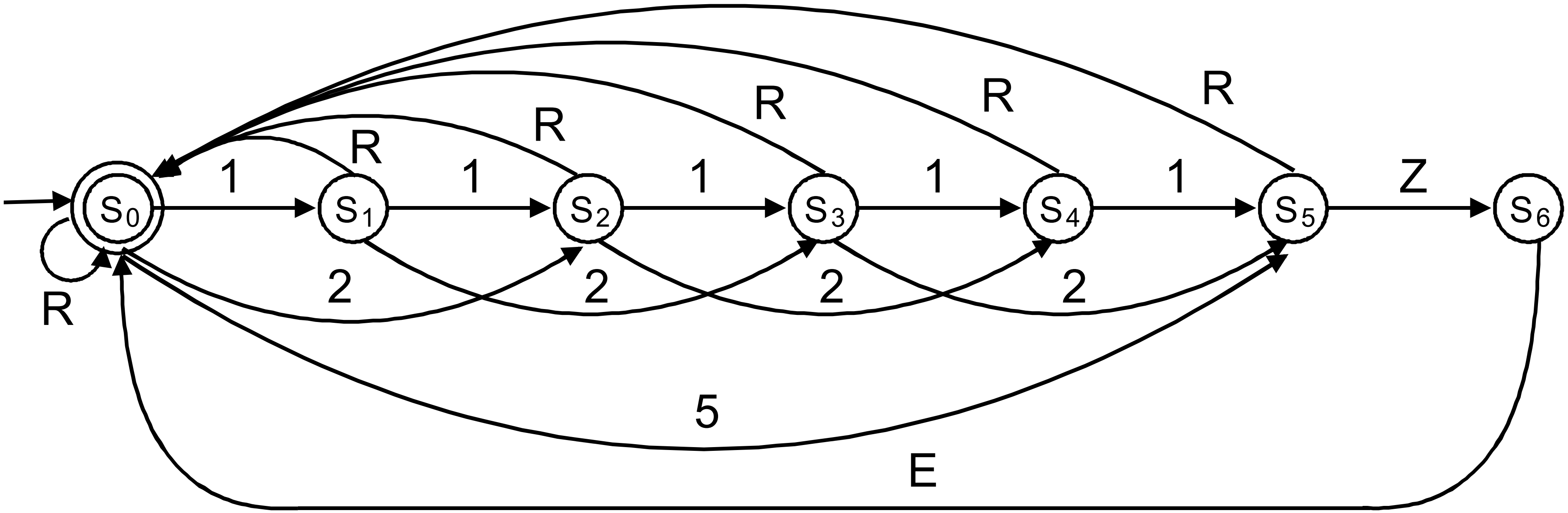
- Eingabezeichen $\boldsymbol{\Sigma} = \{ 1, 2, 5, R, Z, E \}$
- Zustände $\mathbf{S} = \{ S_0, S_1, \ldots, S_6 \}$
- Ausgabe bzw. Endzustand (hier: identisch mit Anfangszustand)

Die Zustandsübergänge bzw. Zustandswechsel lassen sich durch die sogenannte **Überführungsfunktion δ** zum Ausdruck bringen.

- Überführungsfunktion $\boldsymbol{\delta} : \mathbf{S} \times \boldsymbol{\Sigma} \rightarrow \mathbf{S}$

Beispiel

Graphische Darstellung:



- Im Zustand S_i (i = 0 bis 5) genau i € eingeworfen
- Im Zustand S_6 ist Warenentnahme möglich

Beispiel

Tabellarische Darstellung δ:

$S \downarrow$ $\Sigma \rightarrow$	**1**	**2**	**5**	**R**	**Z**	**E**
S_0	S_1	S_2	S_5	S_0		
S_1	S_2	S_3	–	S_0	–	–
S_2	S_3	S_4	–	S_0	–	–
S_3	S_4	S_5	–	S_0	–	–
S_4	S_5	–	–	S_0	–	–
S_5	–	–	–	S_0	S_6	–
S_6	–	–	–	–	–	S_0

<u>Sprache des Automaten:</u>

- Eine besondere Rolle spielt der Zustand S_6, da in diesem Zustand die gewünschte Warenentnahme möglich ist.
- Eingabefolgen, die in den Zustand S_6 führen, sind z. B. **122Z**, **5Z**, aber auch **1R1R...5Z**.
- Es gibt <u>unendlich</u> viele solcher Eingabefolgen.

Alle Eingabefolgen, die in den Zustand S_6 führen sind Worte, gebildet auch Zeichen der Zeichenmenge $\mathbf{\Sigma} = \{ 1, 2, 5, R, Z, E \}$. Sie stellen eine **formale Sprache** über $\mathbf{\Sigma}$ dar, die durch besondere Bildungsregeln gekennzeichnet ist.

Deterministische endliche Automaten

Das 5-Tupel $A = (\mathbf{S}, s_0, \mathbf{F}, \Sigma, \delta)$ bezeichnet einen deterministischen endlichen Automaten, wenn für die einzelnen Komponenten gilt:

S *endliche* Menge der möglichen Zustände des Automaten (Bezeichnung der Elemente mit s, s', s_i usw.)

s_0 Anfangszustand des Automaten, $s_0 \in \mathbf{S}$

F Menge der Endzustände des Automaten, $\mathbf{F} \subseteq \mathbf{S}$

Σ *endliche* Menge der Eingabezeichen (Bezeichnung der Elemente mit a, b, a_i usw.)

δ (*deterministische*) Zustandsüberführungsfunktion, die gewissen Paaren (s,a) des kartesischen Produkts $\mathbf{S} \times \Sigma$ einen Folgezustand s' aus **S** zuordnet. Man schreibt auch $\delta\colon \mathbf{S} \times \Sigma \to \mathbf{S}$ und $\delta(s,a) = s'$.

Anmerkung:

Ob die Funktion δ vollständig ($\rightarrow$ **totale Funktion)** ist, d.h. für alle Paare (s,a) ein Nachfolgezustand vorhanden ist, spielt manchmal eine Rolle. Man kann dies immer durch Einführung eines ggf. noch nicht vorhandenen Zustands „Fehler“ erreichen, der als Folgezustand für alle Paare (s,a) auftritt, für die δ(s,a) nicht definiert ist (ansonsten: $\rightarrow$ **partiell definierte Übergangsfunktion)**.

I. Endliche Automaten

Erweiterung der Übergangsfunktion δ auf Worte aus Σ^*:

Sei Σ^* die Menge aller endlichen Folgen bzw. *Worte* $w = a_1a_2...a_n$, mit $a_i \in \Sigma$ für alle i. Durch iterative Anwendung der (vollständigen) Übergangsfunktion – ausgehend von einem beliebigen $s \in$ **S** – erhält man eine Folge von Zuständen s_i mit $\delta(s,a_1) = s_1$, $\delta(s_1,a_2) = s_2$ ... $\delta(s_{n-1},a_n) = s_n$

Interpretation:

Wird im Zustand s das Symbol a_1 vorgelegt, so geht der Automat in den Zustand $\delta(s,a_1) = s_1$ und liest das nächste Symbol a_2, und geht danach in den nächsten Zustand $\delta(s_1,a_2) = s_2$ usw.

Erweiterung der Übergangsfunktion δ auf Worte aus Σ^*:

Wir definieren $\delta(s,w) := s_n$, dann gilt für die erweiterte Funktion δ:

δ: **S** x Σ^* -> **S** und $\delta(s,w) = \delta(s,a_1 w_{-1}) = \delta(\delta\,(s,a_1),w_{-1})$, wobei w_{-1} den Rest des Wortes w nach Entfernung von a_1 bezeichnet.

Vereinbarungsgemäß sei auch das *leere Wort* ε in Σ^* und wir setzen $\delta(s, \varepsilon) = s$ für alle s.

Interpretation:

Lesen der leere Eingabefolge erzeugt keine Zustandsänderung.

I. Endliche Automaten

1. Sprachgebrauch und Motivation
   - 1.1 Automaten und Zustandsüberführungsfunktion
   - 1.2 Sprache eines Automaten
2. Deterministische endliche Automaten
   - 2.1 Erweiterung der Überführungsfunktion
   - **2.2 Sprache eines deterministischen Automaten**
3. Nicht-deterministische endliche Automaten
   - 3.1 Sprache eines nicht-deterministischen Automaten
   - 3.2 Teilmengenkonstruktion
4. Äquivalenz und Minimierung von Automaten
   - 4.1 Äquivalente und reduzierte Automaten
   - 4.2 Bildung von Äquivalenzklassen

Sprache T(A) eines deterministischen Automaten:

Man sagt ein Wort $w \in \Sigma^*$ werde vom Automaten A *akzeptiert*, wenn $\delta(s_0,w) \in \mathbf{F}$ ist, d.h. wenn die beschriebene iterative Anwendung der Übergangsfunktion ausgehend vom Anfangszustand s_0 auf einen Endzustand $s_f \in \mathbf{F}$ führt.

Definition:

Unter der Sprache T(A) eines deterministischen Automaten versteht man die Menge aller vom Automaten akzeptierten Worte, d.h. :

$T(A) = \{ w \in \Sigma^* \mid \delta(s_0,w) \in \mathbf{F} \}$. Es ist ferner: $\varepsilon \in T(A) \Leftrightarrow s_0 \in \mathbf{F}$.

<u>Bedeutung der Symbole der BNF</u>:

< ... >	spitze Klammern grenzen sogenannte **syntaktische Variable** ein
::=	ist zu lesen als „ist“
|	ist zu lesen als „oder“
{ ... }	geschweifte Klammern zeigen die **Wiederholung** des Klammerinhalts an; <u>Anmerkung</u>: Inhalt **...** kann auch weggelassen werden.

Die BNF-Regeln für eine korrekte Zahlendarstellung:

<number> ::= <unsigned number> | <sign> <unsigned number>

<unsigned number> ::= <unsigned integer> | <unsigned real>

<unsigned integer> ::= <digit> {digit}

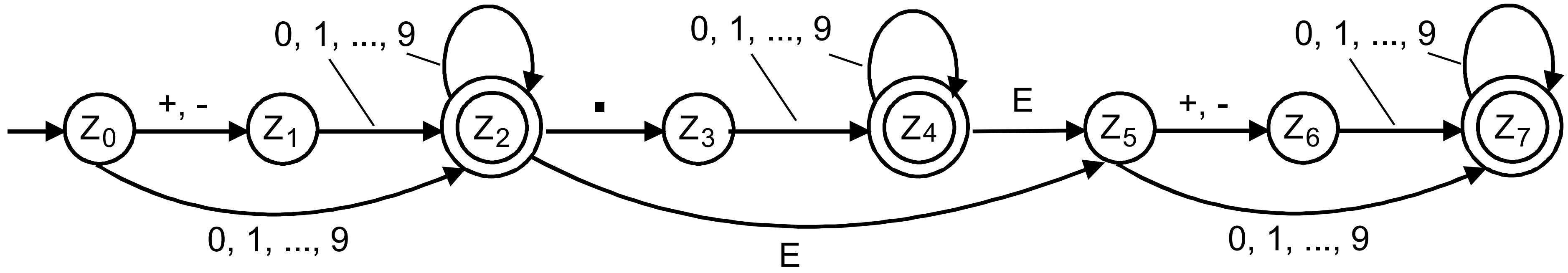
<unsigned real> ::= <unsigned integer> . <digit> {digit} |
<unsigned integer> . <digit> {digit} **E**
<scale factor> | <unsigned integer> **E**
<scale factor>

<scale factor> ::= <unsigned integer> | <sign> <unsigned integer>

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<sign> ::= + | -

Zustandsgraph des Automaten:



Beispiel

Komponenten des Automaten:

Dabei bestehen die einzelnen Komponenten **S**, **F** und Σ aus:

S = { s0,s1,s2,s3,s4,s5,s6,s7},

F = { s2,s4,s7} (Doppelkreise im Graphen!) und

Σ = { 0,1,2,3,4,5,6,7,8,9,+,-,.,E }

Funktionstafel des Automaten:

Andererseits kann der Automat auch durch eine Tabelle für die Werte der Übergangsfunktion δ repräsentiert werden:

s ↓ Σ →	0, … ,9	+, -	.	E
S_0	$\mathbf{S_2}$	S_1	–	–
S_1	$\mathbf{S_2}$	–	–	–
S_2	$\mathbf{S_2}$	–	S_3	S_5
S_3	$\mathbf{S_4}$	–	–	–
S_4	$\mathbf{S_4}$	–	–	S_5
S_5	$\mathbf{S_7}$	S_6	–	–
S_6	$\mathbf{S_7}$	–	–	–
S_7	$\mathbf{S_7}$	–	–	–

Striche bedeuten, dass δ für die entsprechende (s,a)-Kombination nicht definiert ist, bzw. als “Fehlerzustand“ aufzufassen ist, wenn man die Funktion vervollständigt.

Funktionserweiterung des deterministischen EA:

Bisher:

Gemäß der bisherigen Definition der Zustandsüberführungsfunktion δ erhielten wir höchstens einen Folgezustand → **eindeutig**.

Jetzt:

Lassen wir eine Menge **T** möglicher Folgezustände zu → Zustands-überführungsrelation → **nicht eindeutig**.

Interpretation:

Ein Element (s, **a**, s‘) aus T ist also zu interpretieren als: „Im Zustand s ist bei Eingabe von **a** ein Übergang in den Zustand s‘ möglich“

Damit kommen wir zum

Nicht-deterministischen endlichen Automaten

kurz: **NFA**

Folge:

Beim Zustandsgraphen kann dasselbe Eingabesymbol an mehreren Kanten stehen, die aus einem Zustand herausführen.

Somit gilt es den nicht-deterministischen endlichen Automaten bzgl. dieser Erweiterung zu definieren!

<u>Definition</u>:

A = ($\mathbf{S}$, $\mathbf{S}_0$, $\mathbf{F}$, Σ, δ) bezeichnet einen nicht-deterministischen endlichen Automaten, wenn **S**, **F** und Σ die gleiche Bedeutung wie bei deterministischen Automaten haben und für die anderen Komponenten gilt:

$\mathbf{S}_0$ Menge der Anfangszustände des Automaten, von denen es mehr als einen geben kann.

δ (nicht deterministische) Übergangs-Relation, die gewissen Paaren (s,a) des kartesischen Produkts $\mathbf{S} \times \Sigma$ i.a. mehrere mögliche Folgezustände, die man in einer Menge $\mathbf{T} \subseteq \mathbf{S}$ zusammenfassen kann, zuordnet. Man schreibt auch $\delta(s,a) = \mathbf{T}$ und δ: $\mathbf{S} \times \Sigma \rightarrow \mathbf{P(S)}$, wobei **P(S)** die **Potenzmenge** von **S** ist.

I. Endliche Automaten

1. Sprachgebrauch und Motivation
   - 1.1 Automaten und Zustandsüberführungsfunktion
   - 1.2 Sprache eines Automaten
2. Deterministische endliche Automaten
   - 2.1 Erweiterung der Überführungsfunktion
   - 2.2 Sprache eines deterministischen Automaten
3. Nicht-deterministische endliche Automaten
   - **3.1 Sprache eines nicht-deterministischen Automaten**
   - 3.2 Teilmengenkonstruktion
4. Äquivalenz und Minimierung von Automaten
   - 4.1 Äquivalente und reduzierte Automaten
   - 4.2 Bildung von Äquivalenzklassen

Sprache:

Wir betrachten wieder ein beliebiges Wort $w = a_1a_2...a_n$ aus Σ^*. Ausgehend von S_0 erzeugen wir iterativ die Mengen:

$\delta(S_0, a_1) = T_1$;

$\delta(T_1, a_2) = T_2$;

...

$\delta(T_{n-1}, a_n) = T_n$

und erhalten mit T_n die Menge aller möglichen Folgezustände, die mit der Übergangsrelation von den Anfangszuständen ausgehend durch Eingabe des Wortes w erreicht werden können.

Definition:

Ist unter den Zuständen $\mathbf{T}_n$ ein Endzustand, dann soll definitionsgemäß das Wort **w** akzeptiert werden.

Ein Wort $\mathbf{w} \in \Sigma^*$ wird von A akzeptiert, wenn

$$\delta(S_0, \mathbf{w}) \cap \mathbf{F} \neq 0 \text{ (leere Menge)}$$

ist.

Als Sprache des nicht-deterministischen Automaten erhalten wir:

$$T(A) = \{ \mathbf{w} \in \Sigma^* \mid \delta(S_0, \mathbf{w}) \cap \mathbf{F} \neq 0 \}$$

Für den Fall des leeren Wortes ε wird hier implizit definiert:

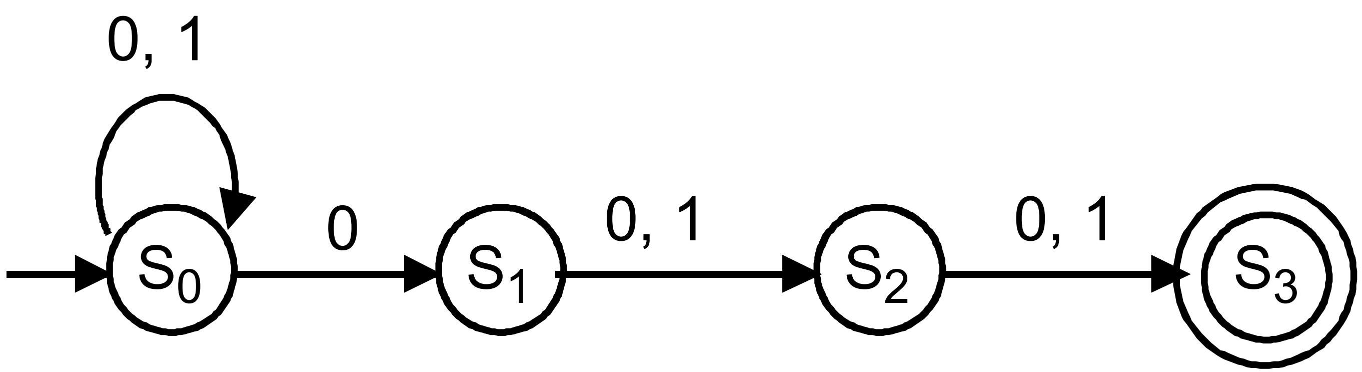
$$\varepsilon \in T(A) \Longleftrightarrow S_0 \cap \mathbf{F} \neq 0$$

<u>Beispiel</u>:

Wir betrachten die Menge aller Worte über dem Alphabet {0, 1}, die als **<u>drittletztes</u>** Symbol eine Null haben, d. h. alle Worte der Gestalt:

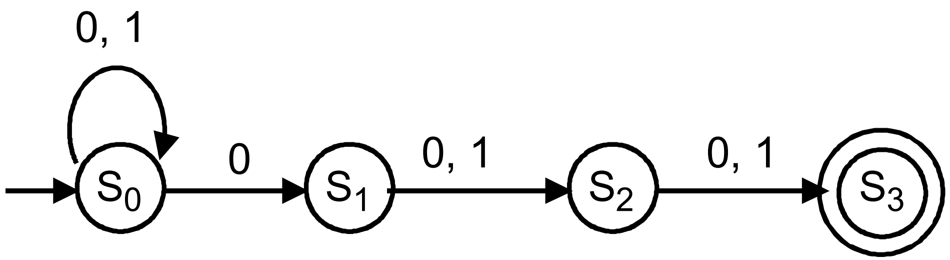
$$T(A) = \{\ \mathbf{w} \in \Sigma^* \mid w = u0v \text{ mit } u = \{0,1\}^* \text{ und } v = \{00, 01, 10, 11\}\ \}$$

<u>Zustandsgraph</u>:



Nicht-deterministische endliche Automaten

Nicht-Determinismus:

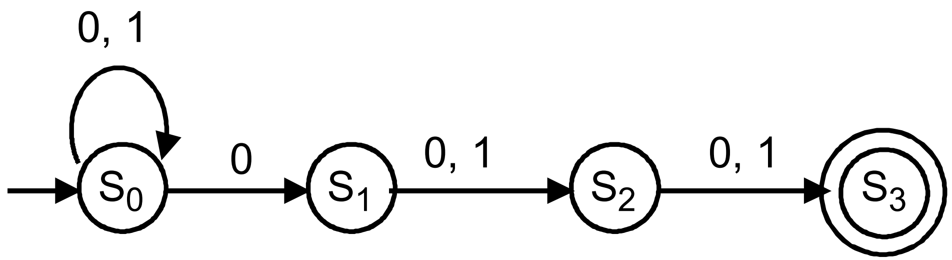


Der Nicht-Determinismus besteht darin, dass vom Zustand S0 zwei Pfeile mit dem Eingabezeichen 0 ausgehen.

Satz:
Man kann auch einen deterministischen Automaten angeben, der dieselbe Sprache akzeptiert, aber wesentlich mehr Zustände besitzt.
Nicht-deterministischer Automat mit k+1 Zustände ➔ deterministischer Automat mit 2^k Zustände.

Nicht-deterministische endliche Automaten

Komponenten des Automaten:



S = $\{S_0, S_1, S_2, S_3\}$
S0 = $\{S_0\}$
F = $\{S_3\}$
Σ = $\{0, 1\}$

Untersuchung der Worte: $w_1 = 10\mathbf{1}01 \notin T(A)$ und $w_2 = 11\mathbf{0}01 \in T(A)$

I. Endliche Automaten

Satz:

Zu jedem nicht deterministischen endlichen Automaten gibt es einen deterministischen, der die gleiche Sprache akzeptiert, d. h.

$$T(A) = T(A')$$

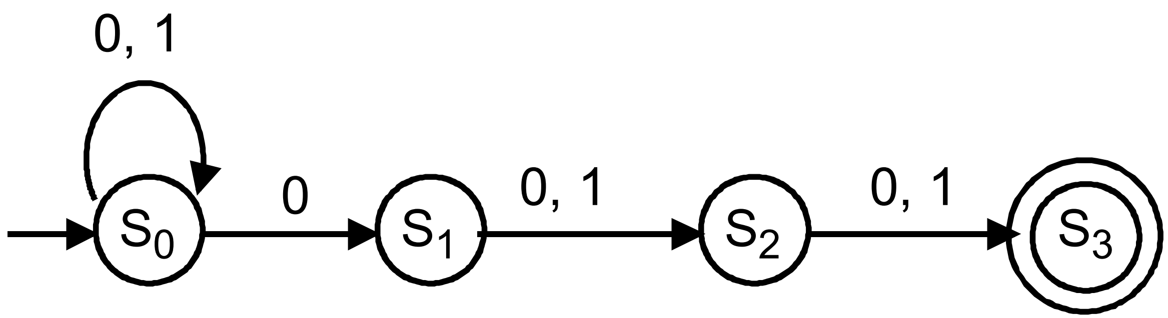
Lösungsverfahren:

Teilmengenkonstruktion!

Der Beweis zeigt, daß bei der Konstruktion des deterministischen Automaten u.U. mit großen Zustandsmengen gerechnet werden muß, da eine Menge M mit k Elementen 2^k Untermengen besitzt ($\rightarrow$ sog. Potenzmenge P(M)).

Teilmengenkonstruktion

Nicht-deterministischer endlicher Automat:

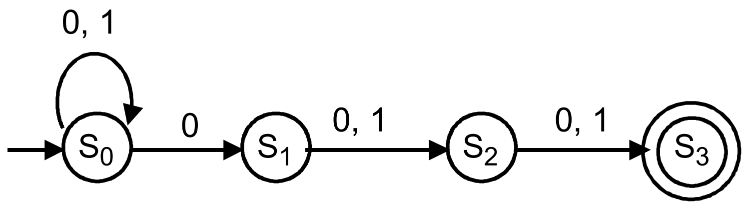


	0	1
S_0	S_0, S_1	S_0
S_1	S_2	S_2
S_2	S_3	S_3

Konstruktion von Teilmengen:

Man erhält die Teilmengen durch sukzessives Nachverfolgen aller möglichen Pfade im ursprünglichen Graphen, beginnend mit der Menge der Anfangszustände.

Nicht-deterministischer endlicher Automat:

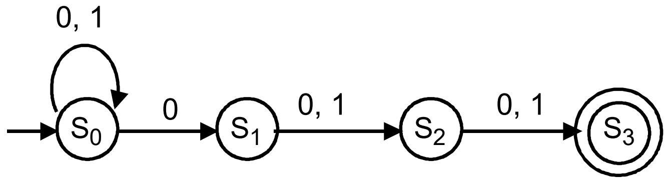


	0	**1**
S_0	S_0, S_1	S_0
S_1	S_2	S_2
S_2	S_3	S_3

Konstruktion von Teilmengen:

Dazu trägt man die Menge der Anfangszustandsmenge in eine Tabelle ein und berechnet hierfür die Zustandsmengen bei Eingabe von **0** und **1**. Neu auftretende Mengen werden unter der Rubrik Zustandsmenge eingetragen und deren Tabelleneinträge für **0** und **1** ermittelt. Dies wird solange fortgesetzt, bis keine neuen Zustandsmengen mehr erscheinen.

Nicht-deterministischer endlicher Automat:

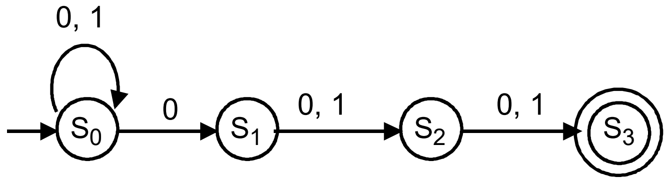


	0	1
s_0	s_0, s_1	s_0
s_1	s_2	s_2
s_2	s_3	s_3

Deterministischer Automat:

	0	1
{s0}	{s0,s1}	{s0}
{s0,s1}	{s0,s1,s2}	{s0,s2}
{s0,s1,s2}	{s0,s1,s2,s3}	{s0,s2,s3}
{s0,s2}	{s0,s1,s3}	{s0,s3}
{s0,s1,s2,s3}	{s0,s1,s2,s3}	{s0,s2,s3}
{s0,s2,s3}	{s0,s1,s3}	{s0,s3}
{s0,s1,s3}	{s0,s1,s2}	{s0,s2}
{s0,s3}	{s0,s1}	{s0}

Nicht-deterministischer endlicher Automat:

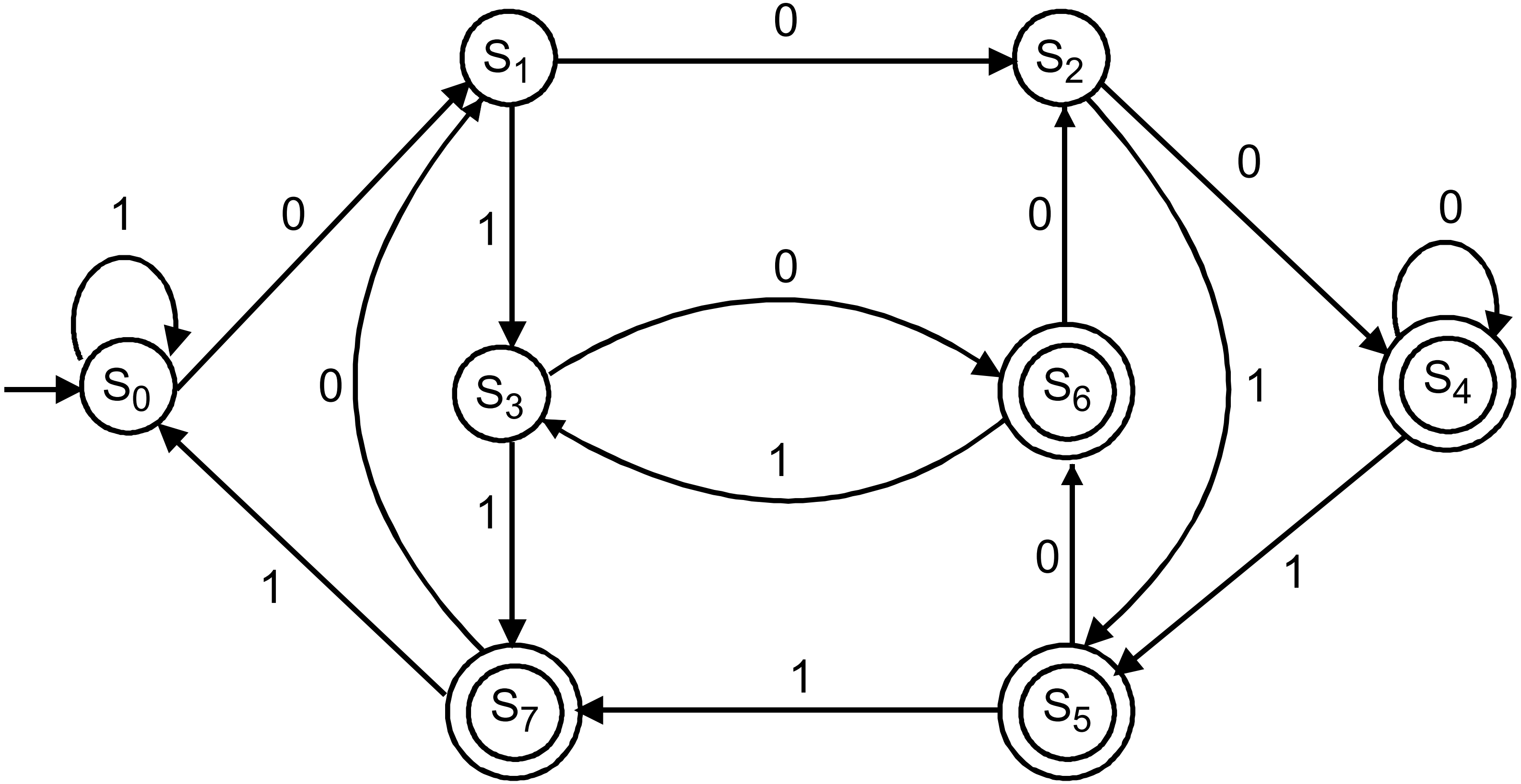


	0	1
S_0	S_0, S_1	S_0
S_1	S_2	S_2
S_2	S_3	S_3

Deterministischer Automat:

	0	1
(0) {s0}	(1) {s0,s1}	(0) {s0}
(1) {s0,s1}	(2) {s0,s1,s2}	(3) {s0,s2}
(2) {s0,s1,s2}	(4) {s0,s1,s2,s3}	(5) {s0,s2,s3}
(3) {s0,s2}	(6) {s0,s1,s3}	(7) {s0,s3}
(4) {s0,s1,s2,s3}	(4) {s0,s1,s2,s3}	(5) {s0,s2,s3}
(5) {s0,s2,s3}	(6) {s0,s1,s3}	(7) {s0,s3}
(6) {s0,s1,s3}	(2) {s0,s1,s2}	(3) {s0,s2}
(7) {s0,s3}	(1) {s0,s1}	(0) {s0}

<u>Deterministischer endlicher Automat</u>:



Ziel:

Reduktion der Anzahl der Zustände eines Automaten ohne dabei die akzeptierte Sprache des Automaten zu verändern.

➔ Zusammenfassung sog. äquivalenter Zustände auf einen **Minimalautomaten**

Definition:

Zwei Zustände s und s´ eines endlichen deterministischen Automaten heißen *äquivalent* (s =~ s´), wenn die Menge der Worte, die in einen Endzustand führen, für beide identisch ist , d.h. $\delta(s,w) \in F \Leftrightarrow \delta(s',w) \in F$ für alle $w \in \Sigma^*$ gilt. Der Automat heißt *reduziert*, wenn keine zueinander äquivalenten Zustände existieren und jeder Zustand von s_0 aus erreichbar ist.

Eigenschaften:

Die Äquivalenzrelation hat die üblichen Eigenschaften der

- Reflexivität,
- Symmetrie und
- Transitivität.

Alle zueinander äquivalenten Zustände können in einer *Äquivalenz-klasse* [s] = { s´ | s´ =~ s }, die durch einen Vertreter s repräsentiert werden kann, zusammengefaßt werden.

Satz:

Zu jedem deterministischen endlichen Automaten gibt es einen reduzierten, der die gleiche Sprache akzeptiert.

<u>**Beweis**</u>:

Beweis durch Angabe eines Algorithmus zur schrittweisen Bildung der Äquivalenzklassen [s] des Automaten A = (**S**, s_0, **F**, Σ, δ) und der Definition des *reduzierten Automaten* A´ = (**S**´, $s´_0$, **F**´, Σ, $\delta´$) mit:

(1) **S**´ = { [s] | s $\in$ **S**};

(2) $s´_0 = s_0$;

(3) **F**´ = { [s] | s $\in$ **F**} und

(4) $\delta´([s],a) = [\delta(s,a)]$ für alle [s] $\in$ S´ und a $\in \Sigma$

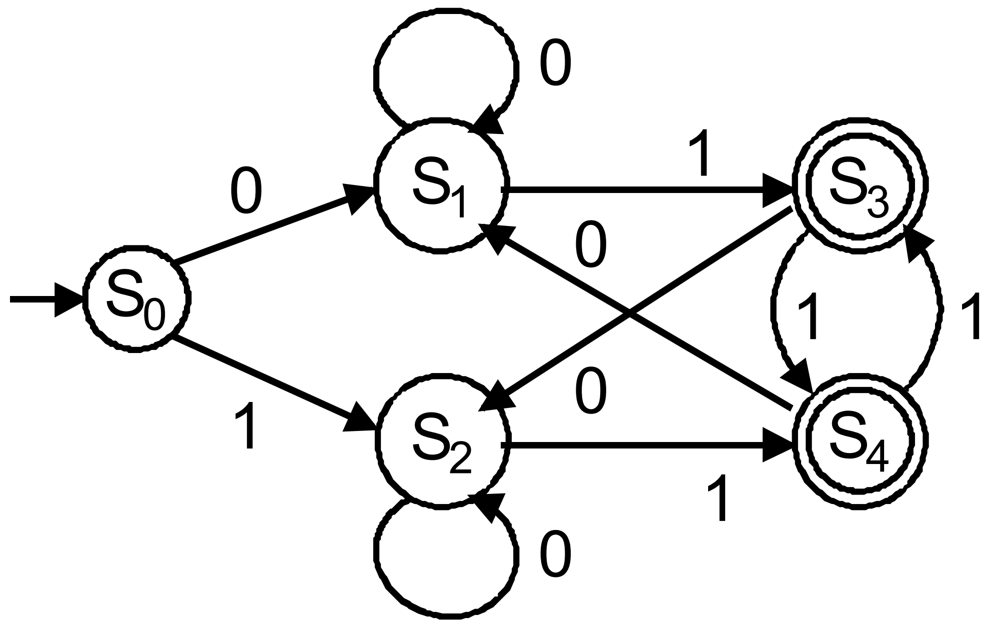
Algorithmus:

Der *Algorithmus zur Bildung der Äquivalenzklassen von A* läuft darauf hinaus die *gröbste Partition* der Menge der Zustände **S** zu finden, bei der die Menge der Folgezustände einer Partitionsmenge für jedes $a \in \Sigma$ wieder eine Partitionsmenge bilden.
Diese Partition stellt dann die Zerlegung in Äquivalenzklassen dar.

Algorithmusbeschreibung:

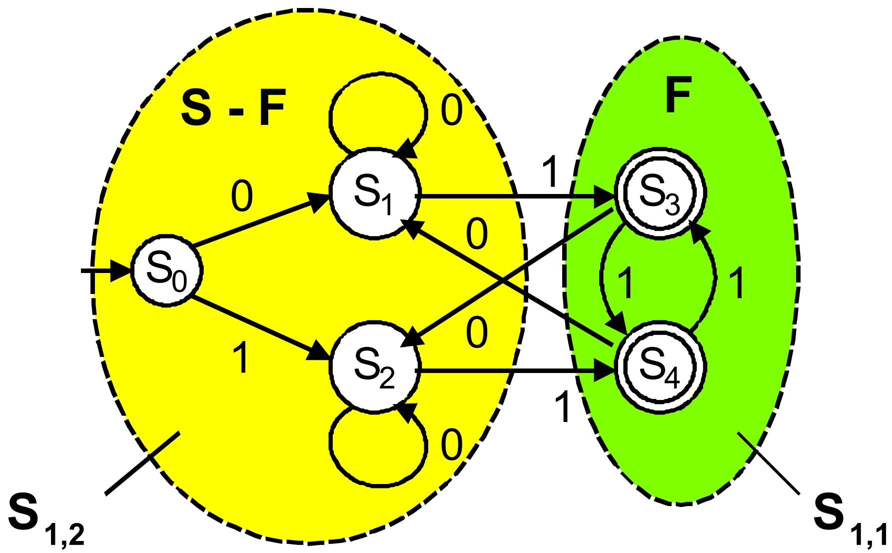
1. Teile die Menge der Zustände **S** in die disjunkten Teilmengen **F** und (**S**-**F**) auf, die offensichtlich nicht beide Elemente der gleichen Äquivalenzklasse enthalten können, da $\delta(s, \varepsilon) \in$ **F** für alle s $\in$ **F** gilt, aber nicht für s $\in$ (**S**-**F**).
2. Die Partitionsmengen werden nun für jedes **a** $\in \Sigma$ untersucht. Liegen für ein bestimmtes **a** die Bilder δ(s, **a**) der Zustände einer Partitionsmenge nicht alle in der gleichen Bildmenge, dann muss die Urbildmenge erneut aufgeteilt werden, d. h. die Partition wird verfeinert und Schritt 2 wiederholt.
3. Der Prozeß endet, wenn keine Verfeinerung der Partition mehr notwendig ist.

Zustandsgraph:



Aufgabe: Gesucht sei der entsprechende **Minimalautomat**.

1. Schritt:



Wir haben zwei disjunkte Partitionsmengen $\mathbf{S_{1,1}}$ und $\mathbf{S_{1,2}}$ mit den Zuständen:

$$S_{1,1} = \{S_3, S_4\} \quad \text{und} \quad S_{1,2} = \{S_0, S_1, S_2\}$$

2. Schritt:

Partitionsmengen werden für jedes **a** $\in \Sigma = \{0, 1\}$ untersucht.

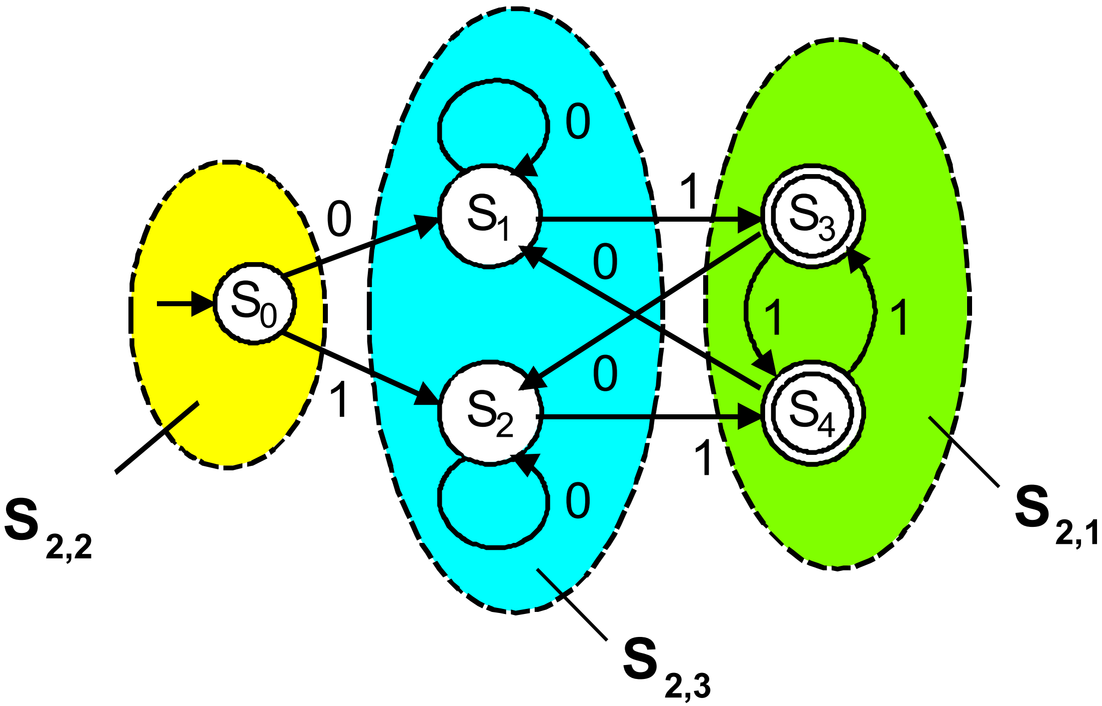
Partition		$S_{1,1}$			$S_{1,2}$		
$\Sigma\downarrow$		S_3	S_4		S_0	S_1	S_2
0		$S_{1,2}$			$S_{1,2}$		
1		$S_{1,1}$			$S_{1,2}$	$S_{1,1}$	

nicht in der gleichen Bildmenge!

$\Rightarrow$ $S_{1,2}$ wird weiter aufgeteilt in: $\{S_0\}$ und $\{S_1, S_2\}$

3. Schritt:

Verfeinerung von $S_{1,2}$



Wiederholung von Schritt 2:

Partitionsmengen werden für jedes **a** $\in \Sigma = \{0, 1\}$ untersucht.

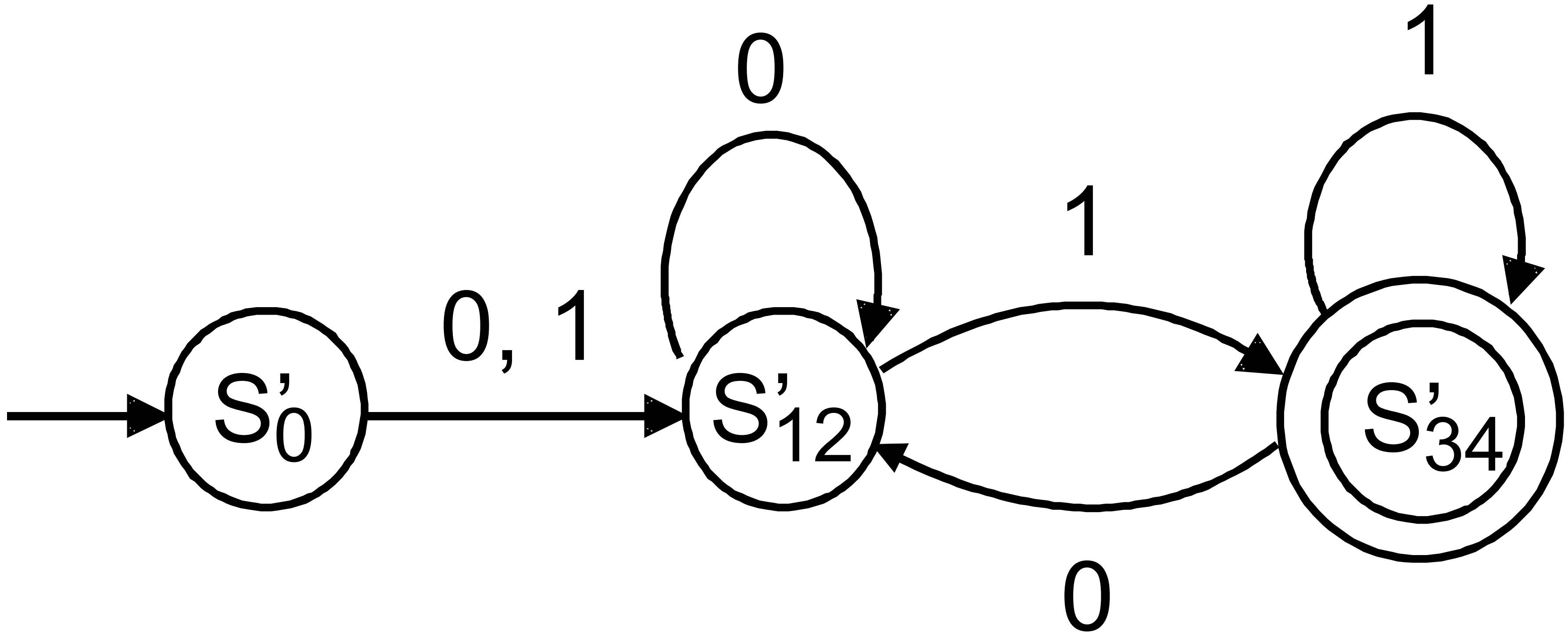
Partition		$S_{2,1}$			$S_{2,2}$		$S_{2,3}$	
$\Sigma\downarrow$		S_3	S_4		S_0		S_1	S_2
0		$S_{2,3}$			$S_{2,3}$		$S_{2,3}$	
1		$S_{2,1}$			$S_{2,3}$		$S_{2,1}$	

jeweils in der gleichen Bildmenge!

$\Rightarrow$ Damit erhalten wir den **Minimalautomaten A´** mit:

$\mathbf{S'_0} := \{S_0\}$; $\mathbf{S'_{12}} := \{S_1, S_2\}$; $\mathbf{S'_{34}} := \{S_3, S_4\}$

Ergebnis:



I. Endliche Automaten

Äquivalenz und Minimierung von Automaten:

- Zwei Zustände eines DFA heißen **äquivalent**, wenn die Menge der Worte, die in einen Endzustand führen, für beide identisch ist.
- Das Zusammenlegen von äquivalenten Zuständen eines DFA führt schließlich zum sogenannten **reduzierten** bzw. **minimalen** Automaten.
- Dazu haben wir alle zueinander äquivalente Zustände in einer sogenannten **Äquivalenzklasse [s]**, die durch einen **Vertreter s** repräsentiert werden kann, zusammengefasst.
- In diesem Zusammenhang haben wir herausgestellt, dass die **Äquivalenzrelation** die üblichen Eigenschaften **Reflexivität**, **Symmetrie** und **Transitivität** hat.

Idee eines weiteren Algorithmus:

Der betrachtete Algorithmus bestimmt die Äquivalenzklassen durch Markierung aller Zustandspaare, die **nicht äquivalent** sein können:

- Dabei sind $\mathbf{x_0}$ die Anfangsmarkierungen, die sich durch die Unterscheidung der **Endzustände** und **Nicht-Endzustände** ergeben.
- Die Markierungen x_1 und x_2 sind die Zustandspaare, die beim ersten bzw. zweiten Durchlauf als **nicht äquivalent** erkannt werden.
- Alle **nicht ausge-x-ten** Zustandspaare stellen eine Äquivalenzklasse dar und können demzufolge zusammengelegt werden.

Äquivalente Zustände

Beschreibung des Algorithmus zur Ermittlung äquivalenter Zustände:

Eingabe: DFA **A** mit den Systemzuständen **1**, **2**, ..., **n**.

Ausgabe: Erkenntnis, welche Zustände von **A** noch zu verschmelzen sind, um den Minimalautomaten zu erhalten.

Voraussetzung: Zustände, welche vom Anfangszustand aus nicht erreichbar sind, müssen zuvor entfernt worden sein.

Algorithmus:

1. Man vereinbare eine Dreiecksmatrix der Form (Indizes k, i):

$$
\begin{array}{llll}
(2, 1) & & & \\
(3, 1) & (3, 2) & & \\
\vdots & & & \\
(n, 1) & (n, 2) & \dots & (n, n-1)
\end{array}
$$

Äquivalente Zustände

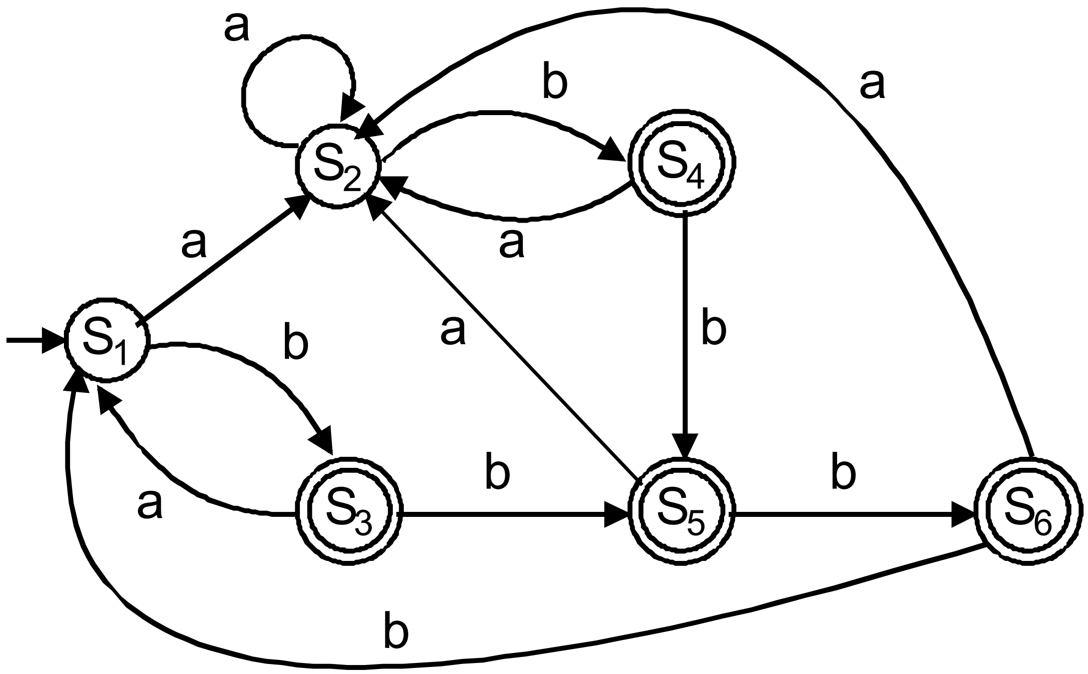
2. Initialisiere Matrixelemente mit dem Wert **1 (markiert)**, falls einer der Indizes **(k, i)** zu einem Endzustand des Automaten gehört und der andere nicht. Anderenfalls initialisiere Matrixelemente mit dem Wert **0 (unmarkiert)**.

3. Für alle mit **0 (unmarkiert)** initialisierten Matrixelemente untersuche, ob das Zustandspaar **($\delta(z_i,a)$, $\delta(z_k,a)$)** für mindestens ein Eingabezeichen $\mathbf{a} \in \Sigma$ zu einem bereits mit **1 (markiert)** initialisierten Matrixelement führt. Wenn ja, dann setze auch den Matrixwert des Elementes **(i, k)** auf **1 (markiert)**.

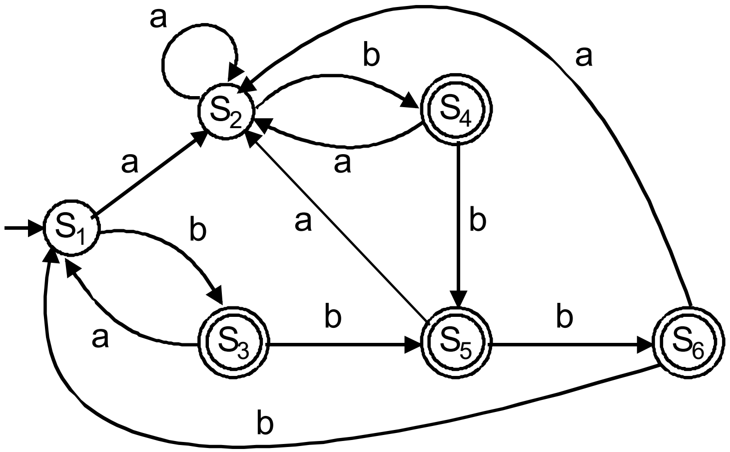
4. Wiederhole Schritt 3, bis sich ein Durchlauf ohne Änderung ergibt.

5. Alle jetzt noch mit **0 (unmarkiert)** besetzten Zustandspaare sind **äquivalente Zustände** und können zu einem Zustand verschmolzen werden.

Zustandsgraph:



Gesucht: Entsprechender Minimalautomat

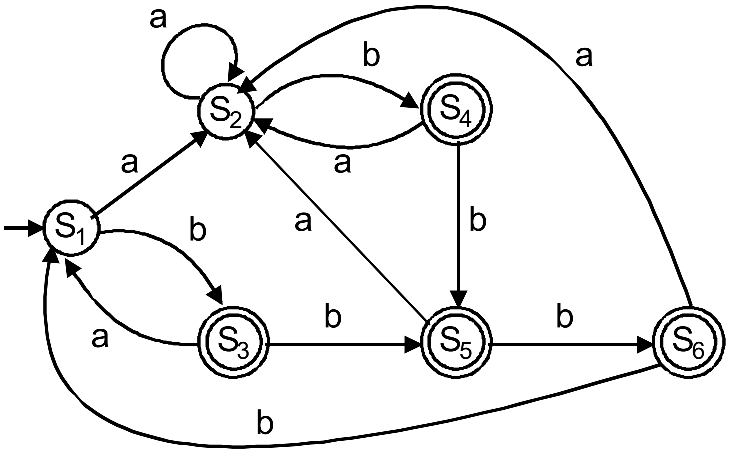


S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0			
	S_1	S_2	S_3	S_4	S_5

k	**i**		**bereits markiert?**
2	1	$(\delta(S_2, a), \delta(S_1, a)) = (S_2, S_2) \rightarrow$	nein
		$(\delta(S_2, b), \delta(S_1, b)) = (S_4, S_3) \rightarrow$	nein
4	3	$(\delta(S_4, a), \delta(S_3, a)) = (S_2, S_1) \rightarrow$	nein
		$(\delta(S_4, b), \delta(S_3, b)) = (S_5, S_5) \rightarrow$	nein
5	3	$(\delta(S_5, a), \delta(S_3, a)) = (S_2, S_1) \rightarrow$	nein
		$(\delta(S_5, b), \delta(S_3, b)) = (S_6, S_5) \rightarrow$	nein

S_2	?				
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0			
	S_1	S_2	S_3	S_4	S_5

Reduzierter Automat

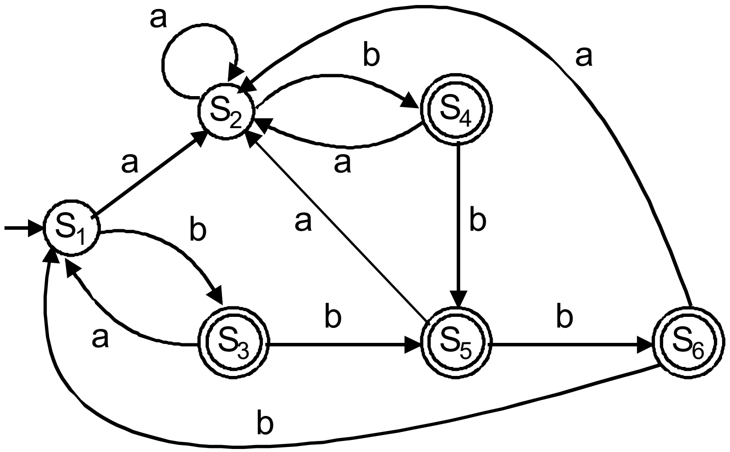


	S_1	S_2	S_3	S_4	S_5
S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0			

k	i		bereits markiert?
6	3	$(\delta(S_6, a), \delta(S_3, a)) = (S_2, S_1)$	$\rightarrow$ nein
		$(\delta(S_6, b), \delta(S_3, b)) = (S_1, S_5)$	$\rightarrow$ ja $\rightarrow x_1$
5	4	$(\delta(S_5, a), \delta(S_4, a)) = (S_2, S_2)$	$\rightarrow$ nein
		$(\delta(S_5, b), \delta(S_4, b)) = (S_6, S_5)$	$\rightarrow$ nein
6	4	$(\delta(S_6, a), \delta(S_4, a)) = (S_2, S_2)$	$\rightarrow$ nein
		$(\delta(S_6, b), \delta(S_4, b)) = (S_1, S_5)$	$\rightarrow$ ja $\rightarrow x_1$

	S_1	S_2	S_3	S_4	S_5
S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0	x_1	x_1	

Reduzierter Automat



S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0	x_1	x_1	
	S_1	S_2	S_3	S_4	S_5

k i **bereits markiert?**

6 5 $(\delta(S_6, a), \delta(S_5, a)) = (S_2, S_2) \rightarrow$ nein
$(\delta(S_6, b), \delta(S_5, b)) = (S_1, S_6) \rightarrow$ ja $\rightarrow x_1$

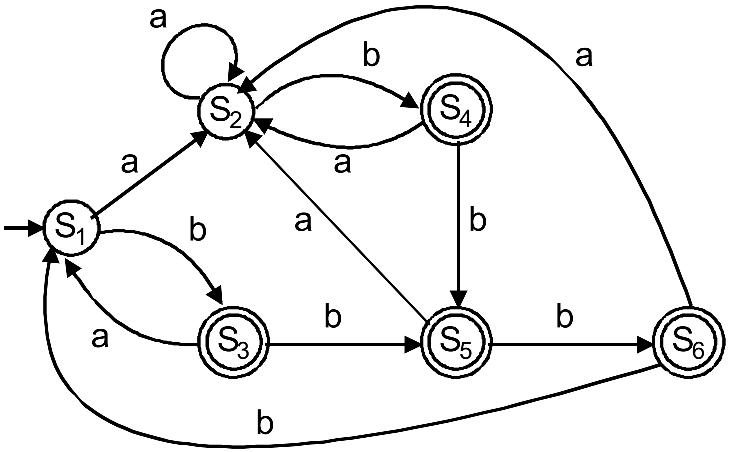
2. Durchgang

2 1 $(\delta(S_2, a), \delta(S_1, a)) = (S_2, S_2) \rightarrow$ nein
$(\delta(S_2, b), \delta(S_1, b)) = (S_4, S_3) \rightarrow$ nein

S_2	?				
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0	x_1	x_1	x_1
	S_1	S_2	S_3	S_4	S_5

Reduzierter Automat

Fortsetzung 2. Durchgang



	S_1	S_2	S_3	S_4	S_5
S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0			
S_6	x_0	x_0	x_1	x_1	x_1

k i **bereits markiert?**

4 3 $(\delta(S_4, a), \delta(S_3, a)) = (S_2, S_1) \rightarrow$ nein

$(\delta(S_4, b), \delta(S_3, b)) = (S_5, S_5) \rightarrow$ nein

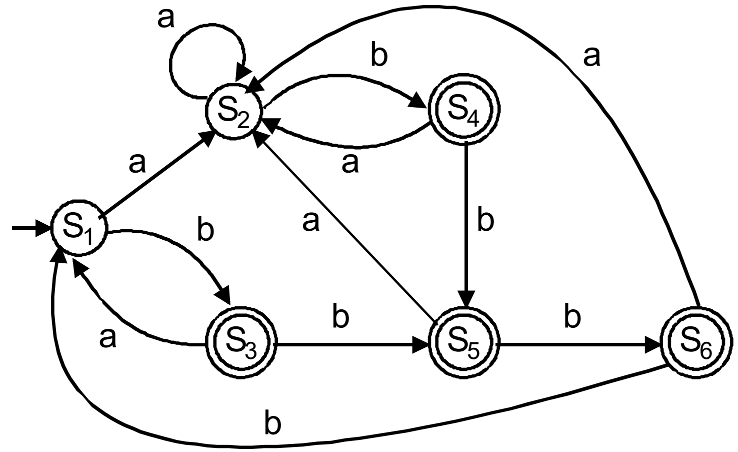
5 3 $(\delta(S_5, a), \delta(S_3, a)) = (S_2, S_1) \rightarrow$ nein

$(\delta(S_5, b), \delta(S_3, b)) = (S_6, S_5) \rightarrow$ ja $\rightarrow x_2$

5 4 $(\delta(S_5, a), \delta(S_4, a)) = (S_2, S_2) \rightarrow$ nein

$(\delta(S_5, b), \delta(S_4, b)) = (S_6, S_5) \rightarrow$ ja $\rightarrow x_2$

	S_1	S_2	S_3	S_4	S_5
S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0	x_2	x_2	
S_6	x_0	x_0	x_1	x_1	x_1



	S_1	S_2	S_3	S_4	S_5
S_2					
S_3	x_0	x_0			
S_4	x_0	x_0			
S_5	x_0	x_0	x_2	x_2	
S_6	x_0	x_0	x_1	x_1	x_1

k i bereits markiert?

3. Durchgang

2 1 $(\delta(S_2, a), \delta(S_1, a)) = (S_2, S_2) \rightarrow$ nein

$(\delta(S_2, b), \delta(S_1, b)) = (S_4, S_3) \rightarrow$ nein

4 3 $(\delta(S_4, a), \delta(S_3, a)) = (S_2, S_1) \rightarrow$ nein

$(\delta(S_4, b), \delta(S_3, b)) = (S_5, S_5) \rightarrow$ nein

	S_1	S_2	S_3	S_4	S_5
S_2	nein				
S_3	x_0	x_0			
S_4	x_0	x_0	nein		
S_5	x_0	x_0	x_2	x_2	
S_6	x_0	x_0	x_1	x_1	x_1

Ergebnis:

Die Zustände S_1 und S_2 bzw. S_3 und S_4 sind **äquivalent** und können damit zusammengelegt werden → **Minimalautomat**