Endliche Automaten

- Einfaches Modellierungswekzeug (z.B. UML-Statecharts)
- Verarbeiten Wörter/Ereignisfolgen
- Erkennen Sprachen
- Erlauben schnelle Spracherkennung
- Anwendungsbereiche: Objektorientierte Modellierung, Model-Checking, Lexikalische Analyse, XML-Parser, Kontrollanweisungen, Spezifikation von Kommunikationsabläufen, Beschreibung von Rechnersystemen und deren Systemprogrammierung, . . .

Funktionsweise

- Taktweises Arbeiten
- Eingabe: ein Wort w
- Lesen von w Zeichen für Zeichen von links nach rechts
- In jedem Takt wird ein Zeichen gelesen.
- In jedem Takt befindet sich der endliche Automat in einem seiner endlich vielen $Zustände$.
- Das Eingabewort w wird akzeptiert, falls sich der Automat nach dem Lesen von w in einem $Endzustand$ befindet.

Komponenten

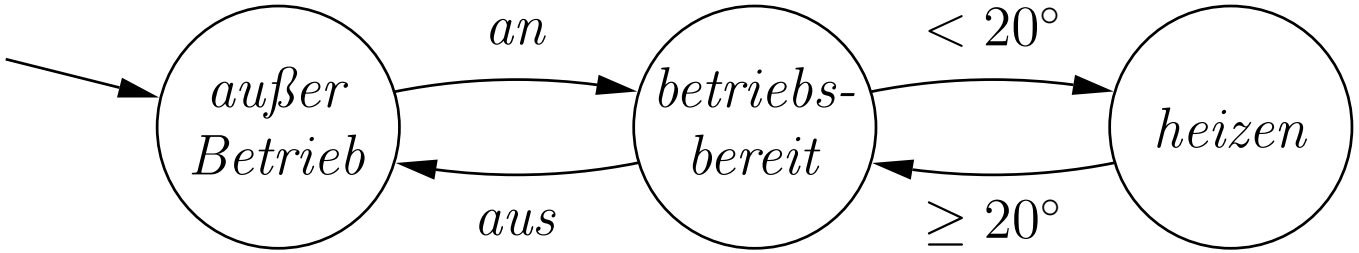
- Zustände

- Eingabealphabet (Menge von potenziellen Ereignissen)

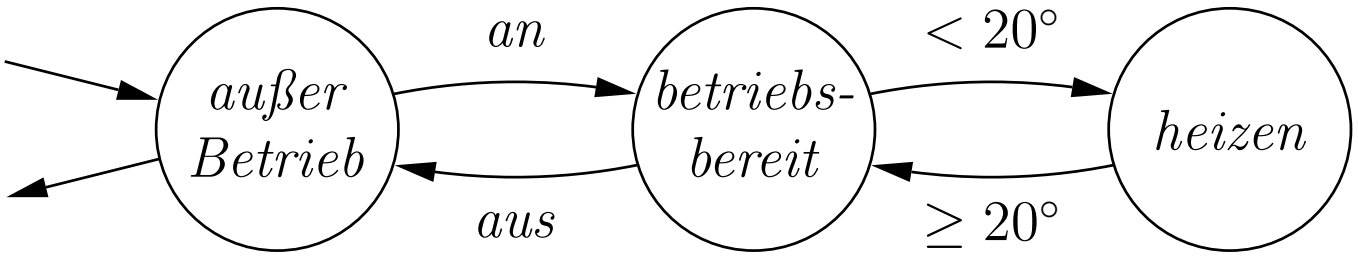
$$\{an, aus, < 20°, \geq 20°\}$$

- Zustandsüberführungen

- Startzustand



- Endzustände



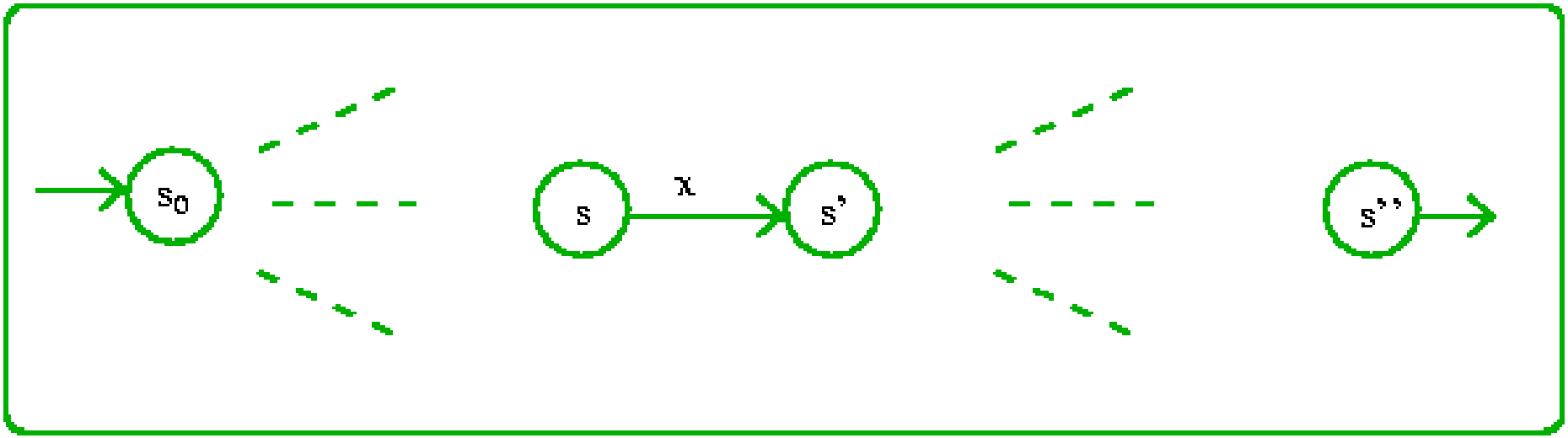
Endlicher Automat

Ein endlicher Automat ist ein System $A = (Z, I, d, s_0, F)$ mit

- Z: endliche Menge von Zuständen,
- I: endliches Eingabealphabet,
- $d \subseteq Z \times I \times Z$: Zustandsüberführung,
- $s_0 \in Z$: Startzustand,
- $F \subseteq Z$: Endzustände.

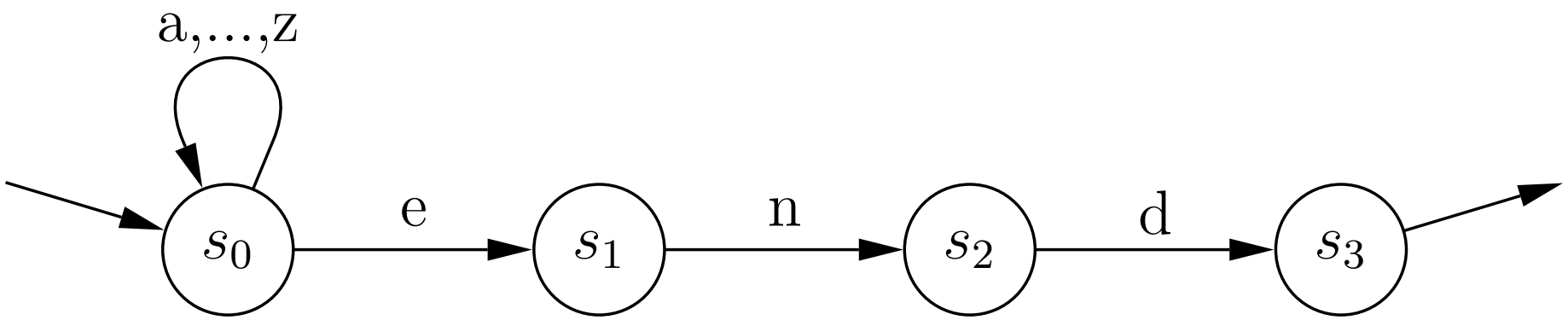
Graphische Darstellung

Gegeben: $A = (Z, I, d, s_0, F)$

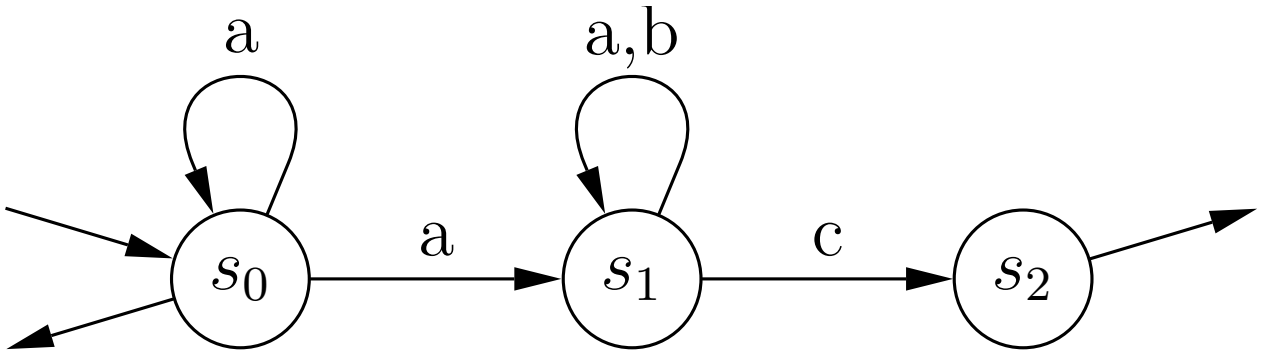


$$s' \in d(s, x), \quad s'' \in F$$

Beispiel 1



Beispiel 2



Akzeptiert u.a. das Wort $aabc$:

- $(s_0, aabc) \vdash\!\!\!- (s_0, abc) \vdash\!\!\!- (s_1, bc) \vdash\!\!\!- (s_1, c) \vdash\!\!\!- (s_2, \lambda)$
- $(s_0, aabc) \vdash\!\!\!- (s_1, abc) \vdash\!\!\!- (s_1, bc) \vdash\!\!\!- (s_1, c) \vdash\!\!\!- (s_2, \lambda)$

Fortgesetzte Zustandsüberführung

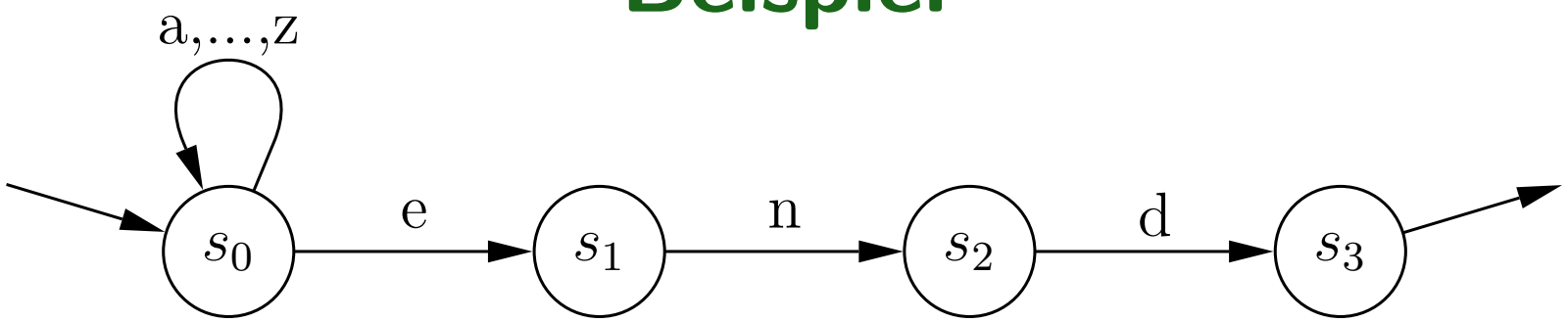
Die fortgesetzte Zustandsüberführung verarbeitet Wörter statt Zeichen.

Gegeben: $A = (Z, I, d, s_0, F)$

Für alle $s, s', s'' \in Z$, $x \in I$, $w \in I^*$:

- $d^*(s, \lambda) = \{s\}$,
- $d^*(s, wx) = \bigcup_{s' \in d^*(s,w)} d(s', x)$.

Beispiel



$$d^*(s_0, end) = \bigcup_{s' \in d^*(s_0,en)} d(s', d) =_{1.} \bigcup_{s' \in \{s_0,s_2\}} d(s', d)$$
$$= d(s_0, d) \cup d(s_2, d) = \{s_0\} \cup \{s_3\} = \{s_0, s_3\}$$

1. $$d^*(s_0, en) = \bigcup_{s' \in d^*(s_0,e)} d(s', n) =_{2.} \bigcup_{s' \in \{s_0,s_1\}} d(s', n)$$
$$= d(s_0, n) \cup d(s_1, n) = \{s_0\} \cup \{s_2\} = \{s_0, s_2\}$$

2. $$d^*(s_0, e) = \bigcup_{s' \in d^*(s_0,\lambda)} d(s', e) = \bigcup_{s' \in \{s_0\}} d(s', e)$$
$$= d(s_0, e) = \{s_0, s_1\}$$

Erkannte Sprache

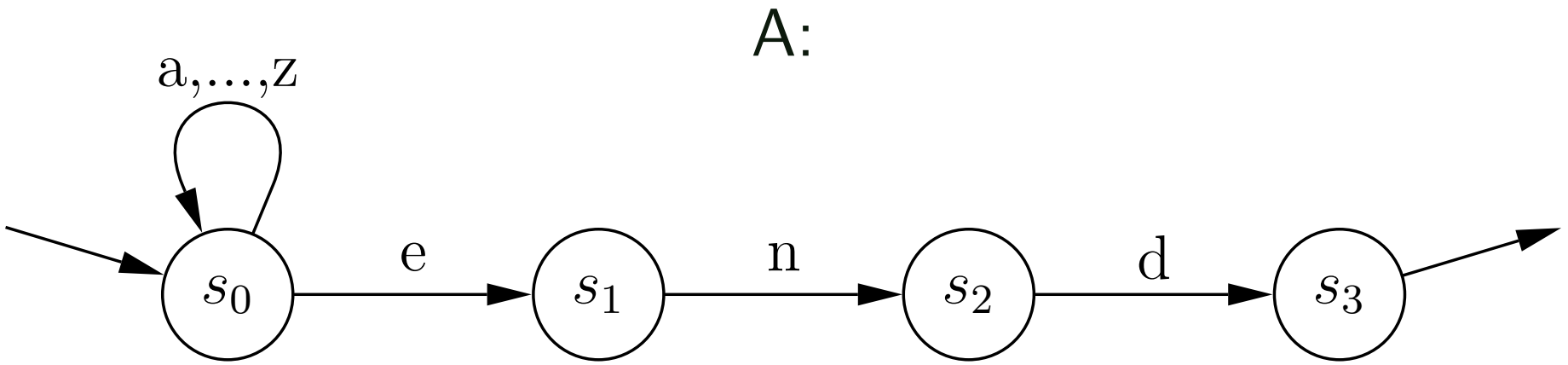
Die erkannte Sprache besteht aus allen Wörtern, die der Automat ausgehend vom Startzustand lesen kann, so dass nach dem Lesen ein Endzustand erreicht wird.

Erkannte Sprache

Gegeben: $A = (Z, I, d, s_0, F)$

$$L(A) = \{w \in I^* \mid d^*(s_0, w) \cap F \neq \emptyset\}$$

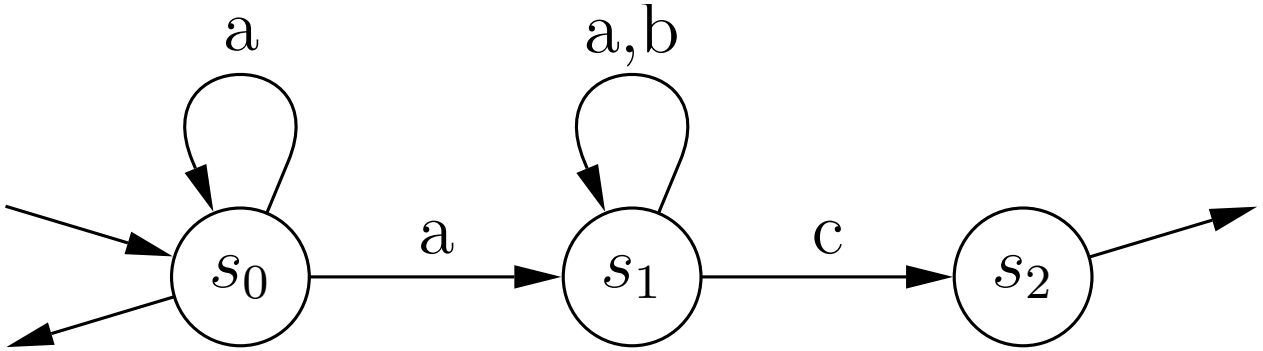
Beispiele



$$L(A) = \{wend \mid w \in \{a, \ldots, z\}^*\}$$

Beispiele

A:



$$L(A) = \{a^n \mid n \in \mathbb{N}\} \cup \{a^n wc \mid n \geq 1, w \in \{a,b\}^*\} = \{a\}^* \cup (\{a\}\{a\}^*\{a,b\}^*\{c\})$$

Teilwortsuche mit endlichen Automaten

- Eingabe: $u, v \in A^*$
- Ausgabe: Alle Stellen in v, an denen u vorkommt.

Idee: Konstruiere einen endlichen Automaten, der alle Anfangswörter (Präfixe) von v erkennt, die auf u enden.

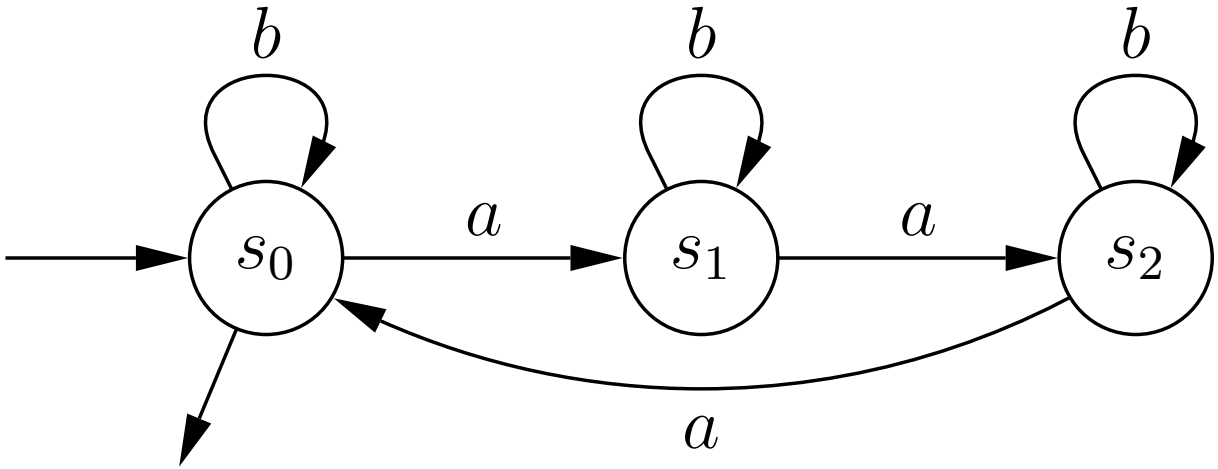
Deterministische endliche Automaten

Definition

Ein deterministischer endlicher Automat (DEA) ist ein System $A = (Z, I, d, s_0, F)$ mit

- Z: endliche Menge von Zuständen,
- I: endliches Eingabealphabet,
- d: $Z \times I \to Z$: Abbildung
- $s_0 \in Z$: Startzustand,
- $F \subseteq Z$: Endzustände.

Beispiel



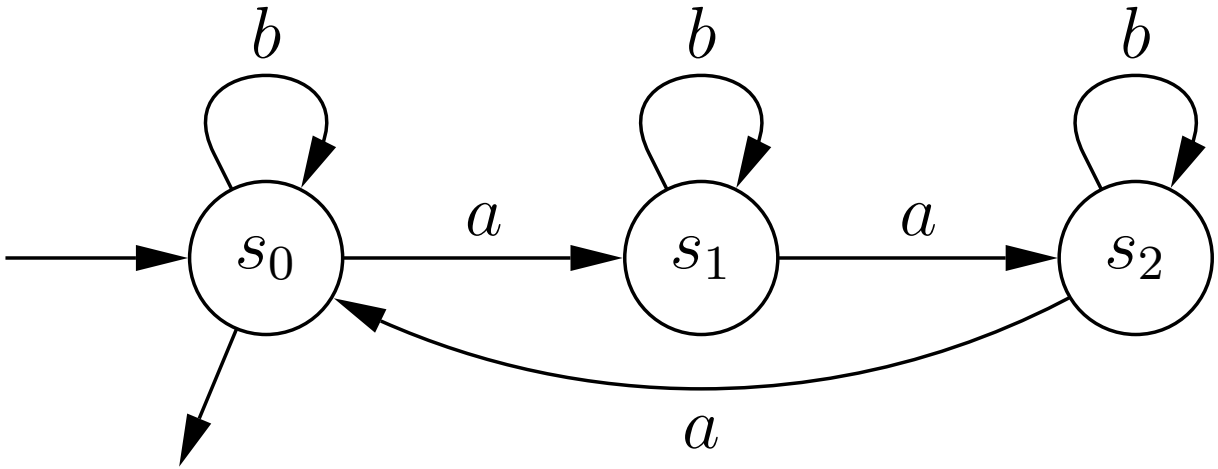
Fortgesetzte Zustandsüberführung

$A = (Z, I, d, s_0, F)$: DEA

Für alle $s, s', s'' \in Z$, $x \in I$, $w \in I^*$:

- $d^*(s, \lambda) = s$;
- $d^*(s, wx) = d(d^*(s, w), x)$.

Beispiel



$$
\begin{aligned}
d^*(s_1, aab) = \quad & d(d^*(s_1, aa), b) = d(d(d^*(s_1, a), a), b) = \\
& d(d(d(d^*(s_1, \lambda), a), a), b) = \\
& d(d(d(s_1, a), a), b) = \\
& d(d(s_2, a), b) = d(s_0, b) = s_0
\end{aligned}
$$

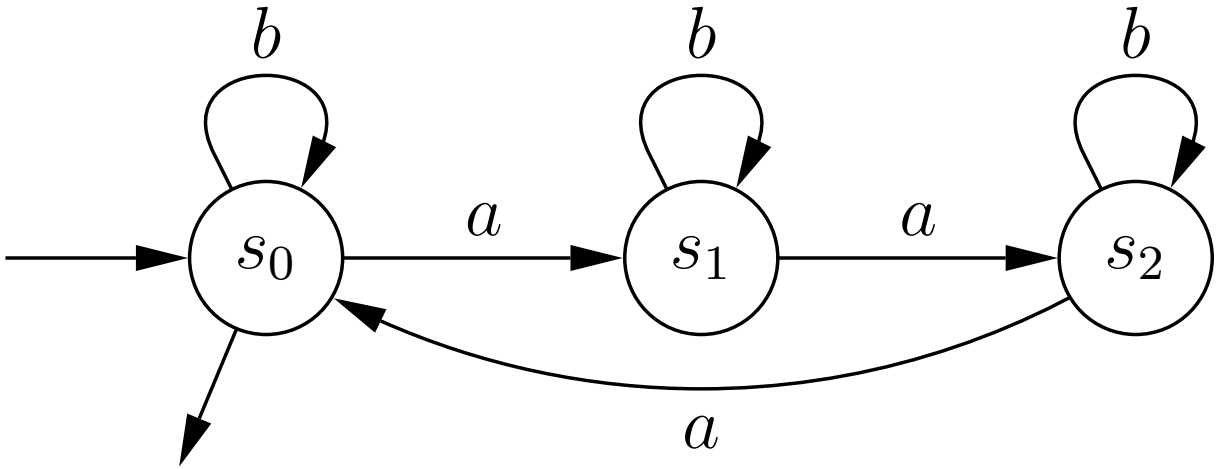
Erkannte Sprache

Erkannte Sprache

Gegeben: $A = (Z, I, d, s_0, F)$

$$L(A) = \{w \in I^* \mid d^*(s_0, w) \in F\}$$

Beispiel



Erkannte Sprache:

$$\{w \in \{a,b\}^* \mid count(a,w) \bmod 3 = 0\}$$

Verarbeitung von Wörtern in iterativer Darstellung

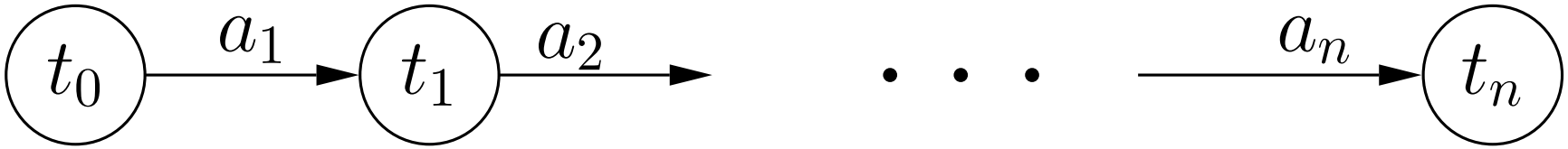
Sei

- $A = (Z, I, d, s_0, F)$
- $w = a_1 \cdots a_n$ mit $a_i \in I$ für $i = 1, \ldots, n$
  $(n = 0$ impl. $w = \lambda)$
- $s, s' \in Z$

Dann

$$s' \in d^*(s, w) \iff \begin{array}{l}\text{Es ex. } t_0, \ldots, t_n \in Z, \text{ so dass} \\ t_i \in d(t_{i-1}, a_i)\ (i = 1, \ldots, n).\end{array}$$

Verarbeitung im Zustandsgraph



Wortproblem

Gegeben: eine Sprache $L \subseteq I^*$
(z.B. als endlicher Automat).

Eingabe: Ein Wort $w \in I^*$.

Ausgabe: Ja, falls $w \in L$
Nein, sonst.

Satz (schnelle Worterkennung)
Für von endlichen Automaten erkannte Sprachen ist das Wortproblem in linearer Zeit lösbar.

Potenzautomat

Gegeben: $A = (Z, I, d, s_0, F)$

$$\mathcal{P}(A) = (\mathcal{P}(Z), I, D, \{s_0\}, F_{\mathcal{P}})$$

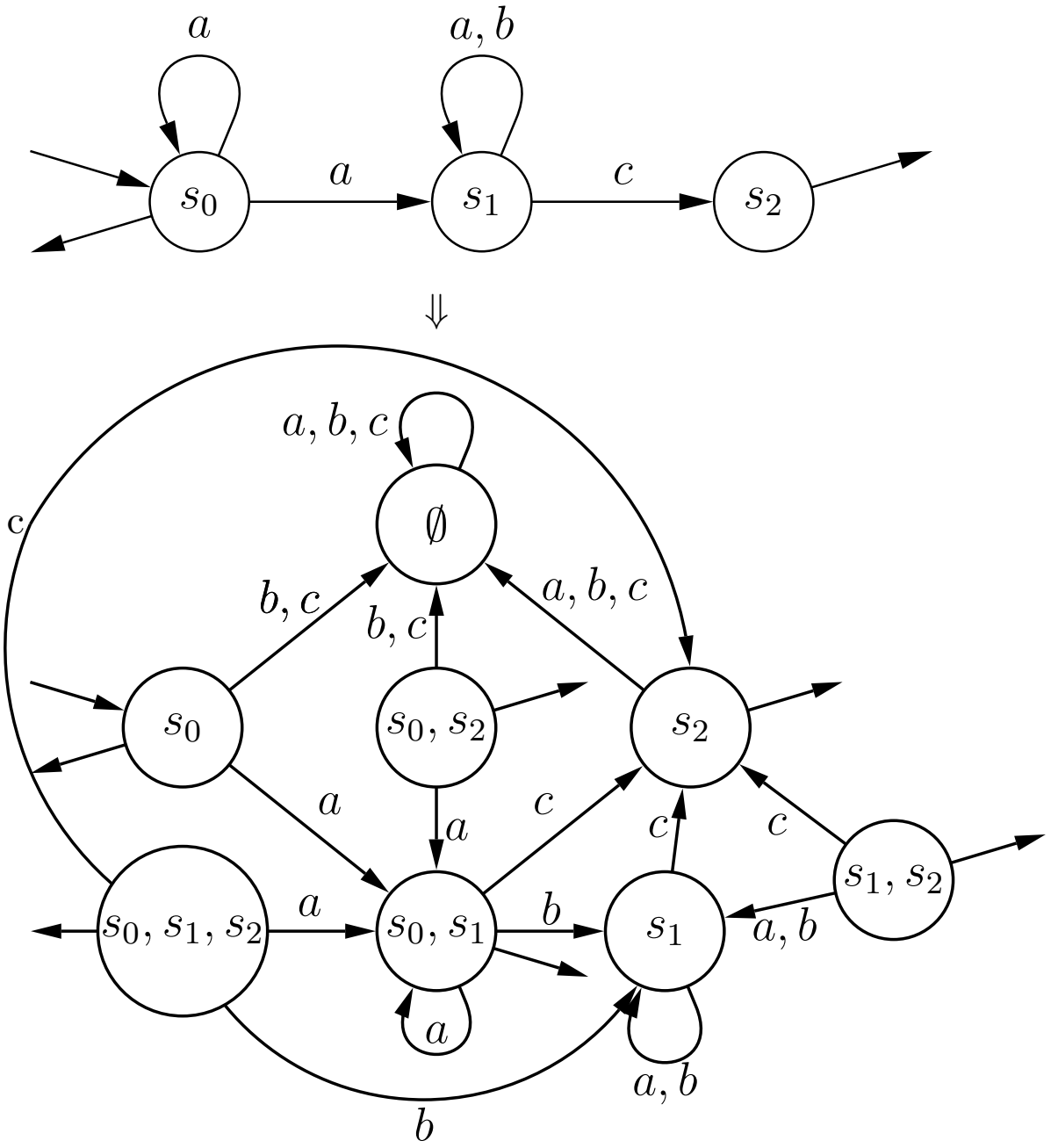
- $\mathcal{P}(Z) = \{S \mid S \subseteq Z\}$: *Potenzmenge* von Z;
- $D\colon \mathcal{P}(Z) \times I \to \mathcal{P}(Z)$ mit

$$D(S, x) = \bigcup_{s \in S} d(s, x)$$

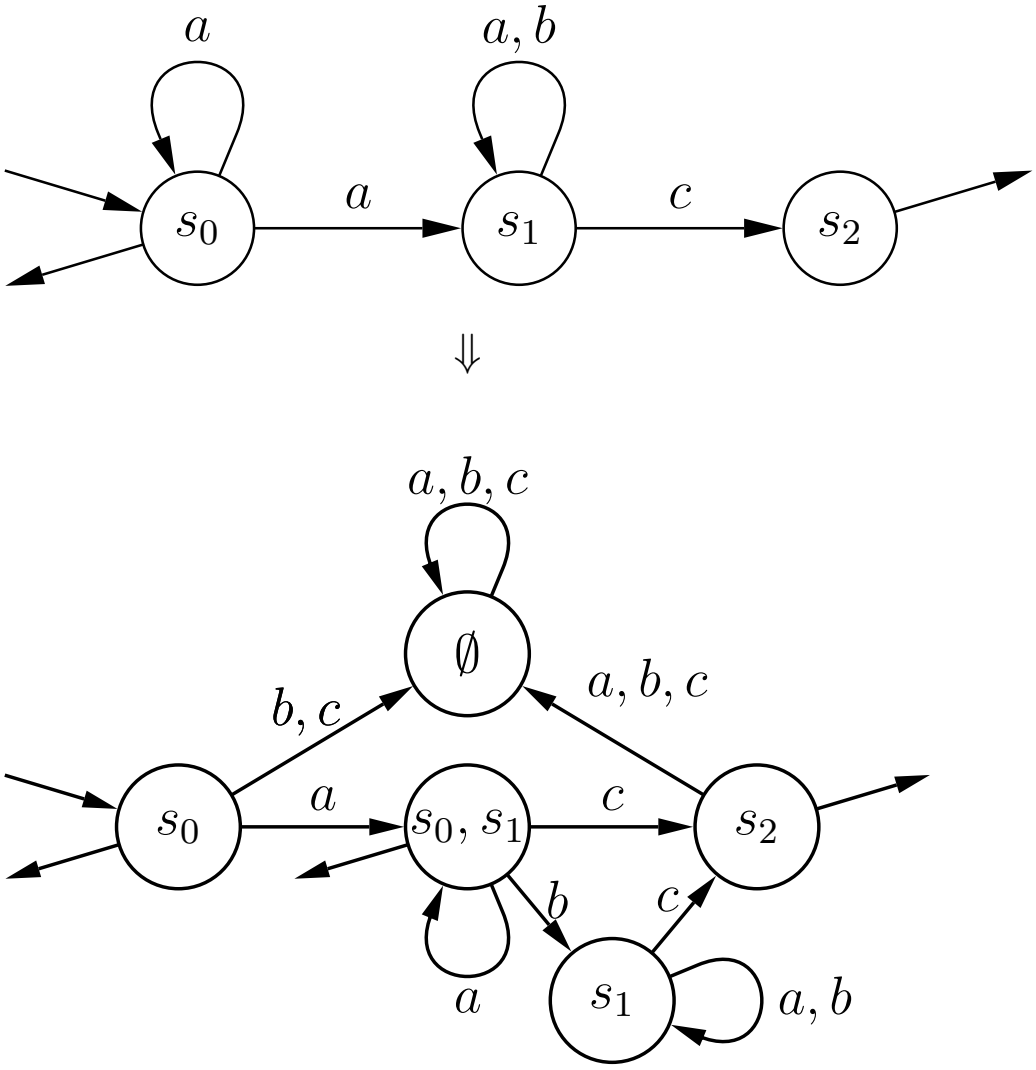
für alle $S \in \mathcal{P}(Z)$, $x \in I$;

- $F_{\mathcal{P}} = \{S \in \mathcal{P}(Z) \mid S \cap F \neq \emptyset\}$.

Beispiel:


a
a, b
a
c
s_0
s_1
s_2
⇓
a, b, c
∅
c
b, c
b, c
a, b, c
s_0
s_0, s_2
s_2
a
a
c
c
c
s_0, s_1, s_2
a
s_0, s_1
b
s_1
s_1, s_2
a, b
a
a, b
b

Ein optimierter Potenzautomat



(Alle Zustände, die von s_0 nicht erreichbar sind, wurden gestrichen.)

Äquivalenz von nichtdeterministischen und deterministischen endlichen Automaten

Satz (Äquivalenz)
Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat. Dann gilt

$$L(A) = L(\mathcal{P}(A)).$$

Schnelle Worterkennung

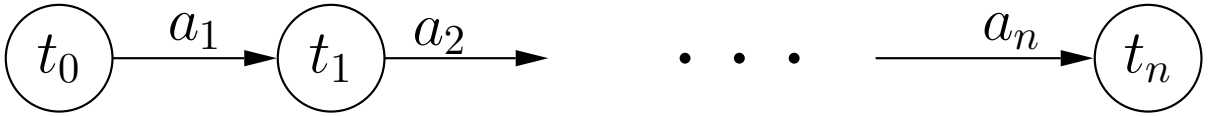
Satz

Für von endlichen Automaten erkannte Sprachen ist das Wortproblem in linearer Zeit lösbar.

Beweisskizze

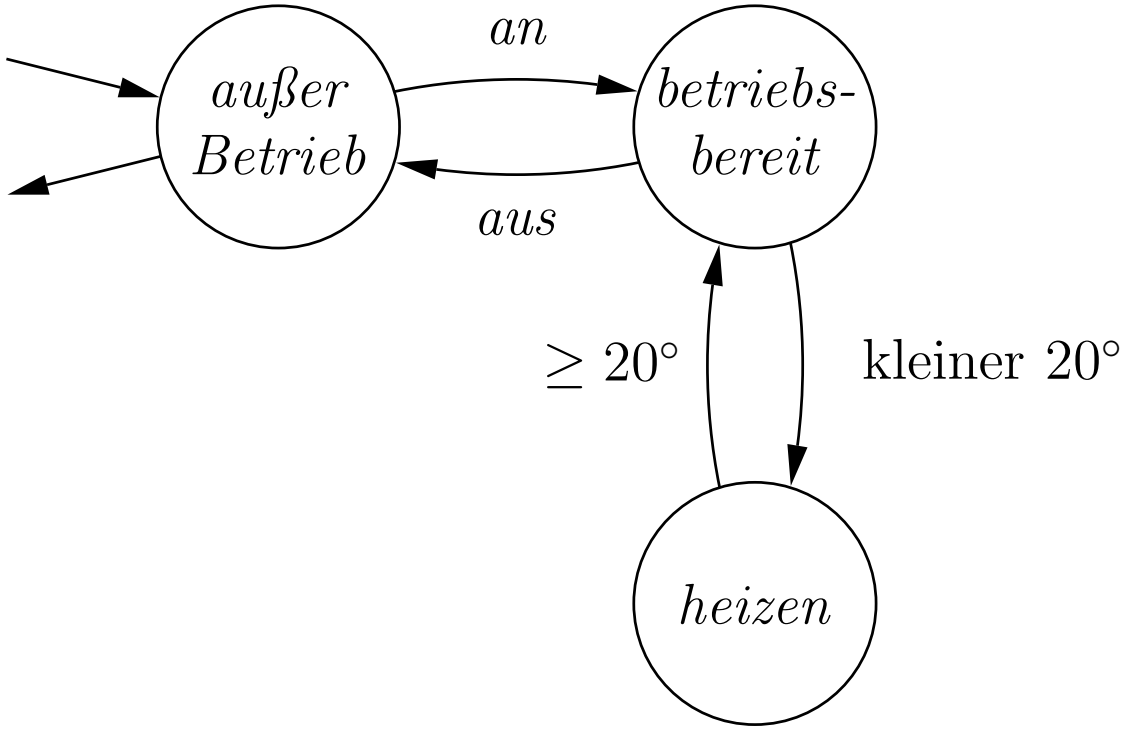
Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat.

1. Falls A nichtdeterministisch ist, $A := \mathcal{P}(A)$.
   (Dies muss höchstens einmal durchgeführt werden.)
2. Sei $w = a_1 \cdots a_n$ $(a_i \in I,\ i = 1, \ldots, n)$.
   Verarbeite w mit A:



   (Zeitverbrauch: n Schritte, da Folgezustände eindeutig sind.)
3. Ja, falls $t_n \in F$; sonst nein.

Model-Checking (beispielhaft)



$L(heating)$: Menge aller möglichen Abläufe

$L_{forbidden}$: Alle verbotenen Abläufe

Heating ist korrekt bezüglich $L_{forbidden}$, falls

$$L(heating) \cap L_{forbidden} = \emptyset.$$

- Kann man einen endlichen Automaten für den Schnitt konstruieren?
- Kann man feststellen, ob ein beliebiger, gegebener endlicher Automat die leere Sprache erkennt?

Produktautomat (Parallelschaltung zweier endlicher Automaten)

Gegeben: deterministische endliche Automaten

$$A_1 = (Z_1, I, d_1, s_{01}, F_1),\ A_2 = (Z_2, I, d_2, s_{02}, F_2)$$

Produktautomat

$$A_1 \times A_2 = (Z_1 \times Z_2, I, d, (s_{01}, s_{02}), F_1 \times F_2)$$

mit $d((s_1, s_2), x) = (d_1(s_1, x), d_2(s_2, x))$ für alle $(s_1, s_2) \in Z_1 \times Z_2$ und $x \in I$.

Produktautomat erkennt Schnitt

Satz

Seien $A_1 = (Z_1, I, d_1, s_{01}, F_1)$, $A_2 = (Z_2, I, d_2, s_{02}, F_2)$ deterministische endliche Automaten. Dann gilt

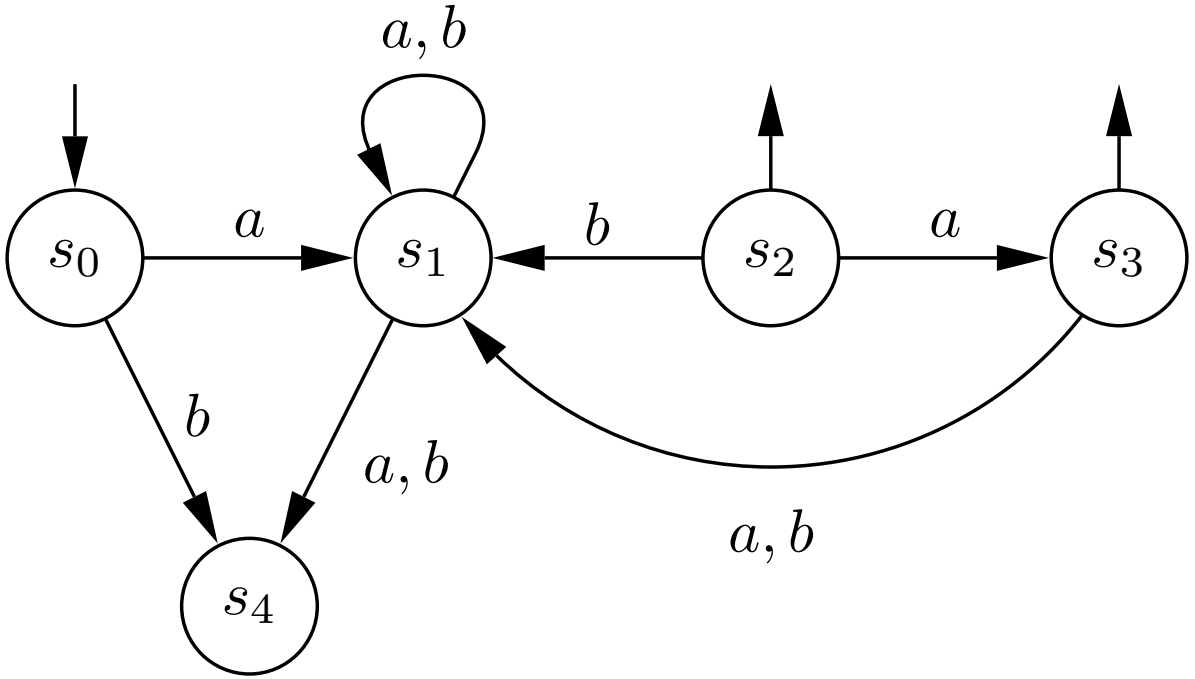
$$L(A_1 \times A_2) = L(A_1) \cap L(A_2).$$

Leerheitsproblem

- Eingabe: eine Sprache L
  (z.B. in Form eines endlichen Automaten).
- Ausgabe: Ja, falls $L = \emptyset$
  Nein sonst.

Beispiel

Eingabe:



Ausgabe: Ja

Satz (Lösbarkeit des Leerheitsproblems)
Für von endlichen Automaten erkannte Sprachen ist das Leerheitsproblem lösbar.

Gesucht:

Algorithmus, der für jeden endlichen Automaten A die folgende Funktion $leer$ berechnet:

$$leer(A) = \begin{cases} Ja, \text{ falls } L(A) = \emptyset \\ Nein \text{ sonst} \end{cases}$$

Überlegung

$L(A) = \emptyset$ genau dann, wenn es keinen Weg in A vom Startzustand zu einem Endzustand gibt.

Idee

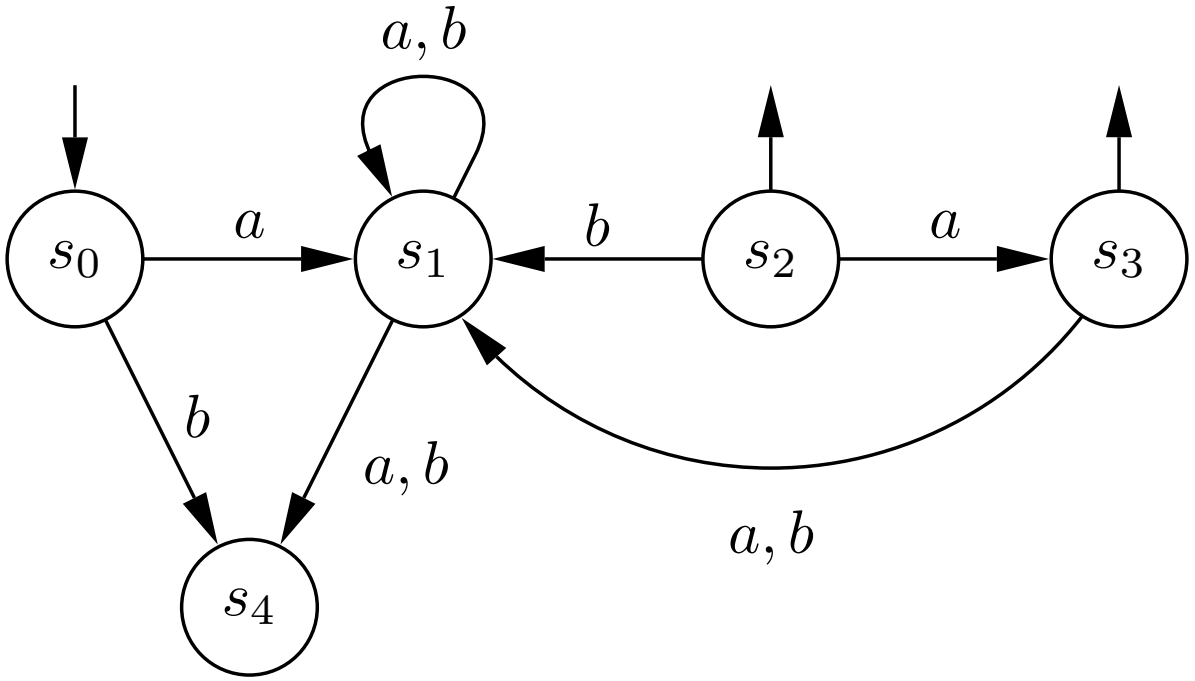
1. Sammle alle vom Startzustand erreichbaren Zustände auf.
2. $L(A) = \emptyset$ genau dann, wenn sich in der Menge der gesammelten Zustände kein Endzustand befindet.

Algorithmus (Skizze)

Gegeben: DEA $A = (Z, I, d, s_0, F)$.

1. (Aufsammeln der erreichbaren Zustände)
   (a) $R_0 := \{s_0\}$; $i := 0$; $R_1 = R_0 \cup \{d(s_0, x) \mid x \in I\}$;
   (b) while $R_{i+1} \neq R_i$ do $i := i + 1$;
   $$R_{i+1} := R_i \cup \{d(s, x) \mid s \in R_i, x \in I\}$$
   (end of while)
2. (Entscheiden)
   If $R_i \cap F = \emptyset$ then Ja else Nein

Beispiel



1. $R_0 = \{s_0\}$

   $R_1 = \{s_0\} \cup \{s_1, s_4\} = \{s_0, s_1, s_4\}$

   $R_2 = \{s_0, s_1, s_4\} \cup \{s_1, s_4\} = \{s_0, s_1, s_4\}$

2. $\{s_0, s_1, s_4\} \cap \{s_2, s_3\} = \emptyset$, d.h., die erk. Sprache ist leer.

Korrektheit

Termination (Algorithmus hält.)
Es existiert ein $m \in \mathbb{N}$: $R_m = R_{m+1}$.

Partielle Korrektheit (Algorithmus liefert bei Halten korrektes Resultat.)
Sei $m \in \mathbb{N}$ die kleinste Zahl mit $R_m = R_{m+1}$. Dann enthält R_m alle von s_0 erreichbaren Zustände, d.h. $R_m = \{d^*(s_0, w) \mid w \in I^*\}$.

Endliche Automaten mit λ-Übergängen

- können aktuellen Zustand wechseln, ohne ein Zeichen zu lesen;
- sind praktisch (vereinfachen oft die Modellierung mit endlichen Automaten);
- sind äquivalent zu DEA's.

Verallgemeinerte endliche Automaten

- lesen in jedem Schritt ein Wort statt eines Zeichens;
- sind nützlich für die Modellierung mit endlichen Automaten ("Abkürzen" möglich);
- sind äquivalent zu DEA's.

Minimale endliche Automaten

- Jeder DEA A kann in einen äquivalenten DEA $Min(A)$ übersetzt werden, dessen Anzahl von Zuständen minimal ist. $Min(A)$ ist bis auf Zustandsnamen eindeutig.
- Minimierungsverfahren kann benutzt werden, um zu entscheiden, ob zwei endliche Automaten äquivalent sind, d.h., ob sie dieselbe Sprache erkennen.