Potenzautomat

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

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

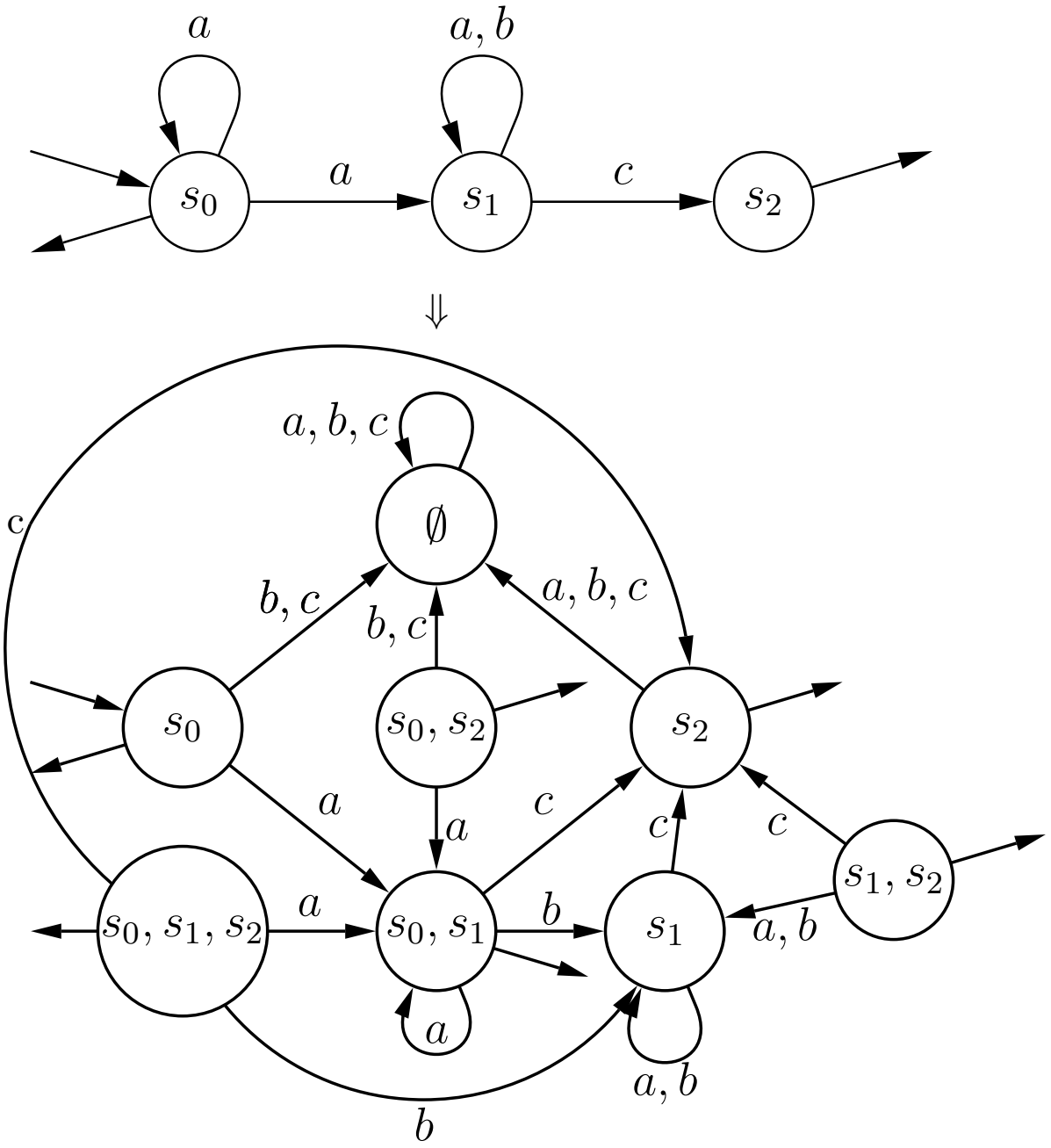
- $\mathcal{P}(Z) = \{S \mid S \subseteq Z\}$: *Potenzmenge* von Z;
- $D\colon \mathcal{P}(Z) \times I \to \mathcal{P}(Z)$ mit

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

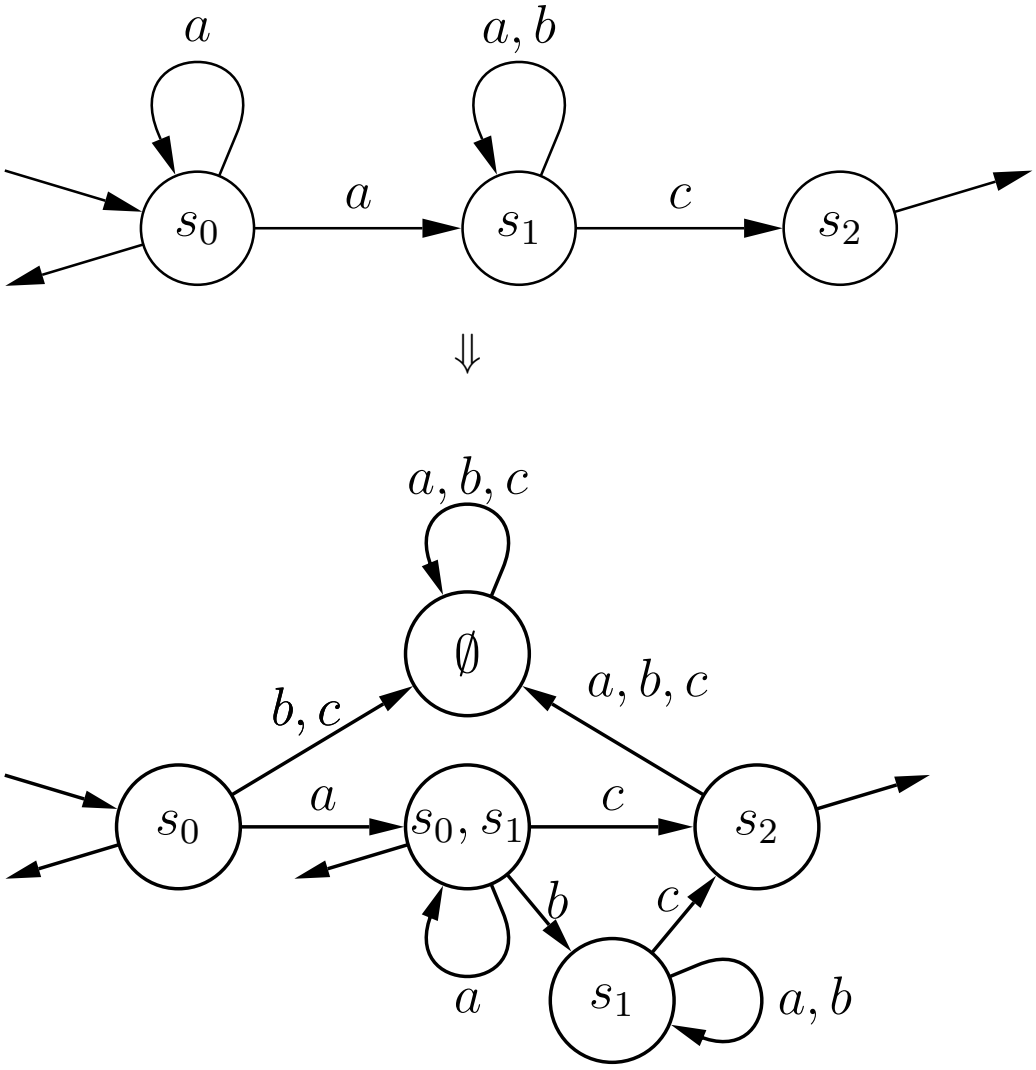
für alle $S \in \mathcal{P}(Z)$, $x \in I$;

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

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_1, s_2
s_0, s_1, s_2
a
s_0, s_1
b
s_1
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

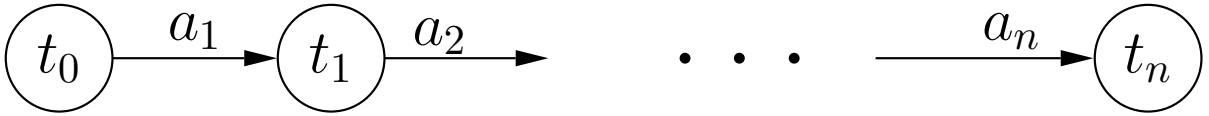
Satz

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

Beweisskizze

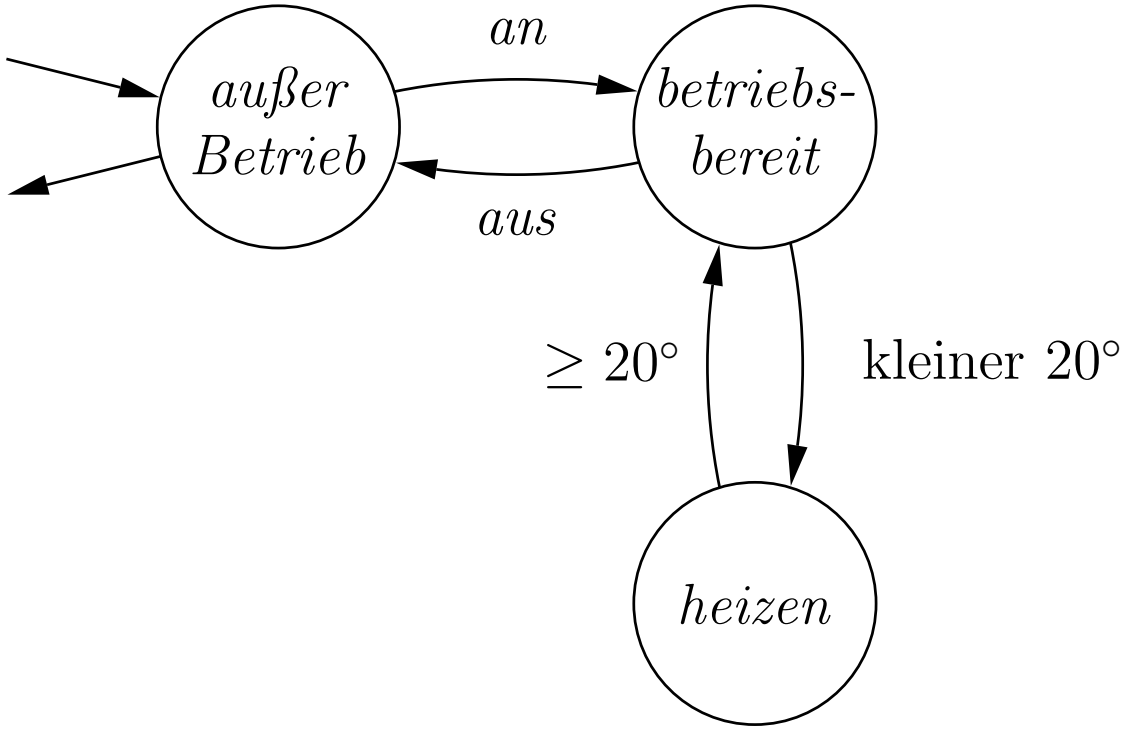
Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat.

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

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

Vereinigungsautomat

Gegeben: deterministische endliche Automaten

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

Vereinigungsautomat

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

wobei d wie beim Produktautomaten definiert ist.

Der Vereinigungsautomat erkennt die Vereinigung

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

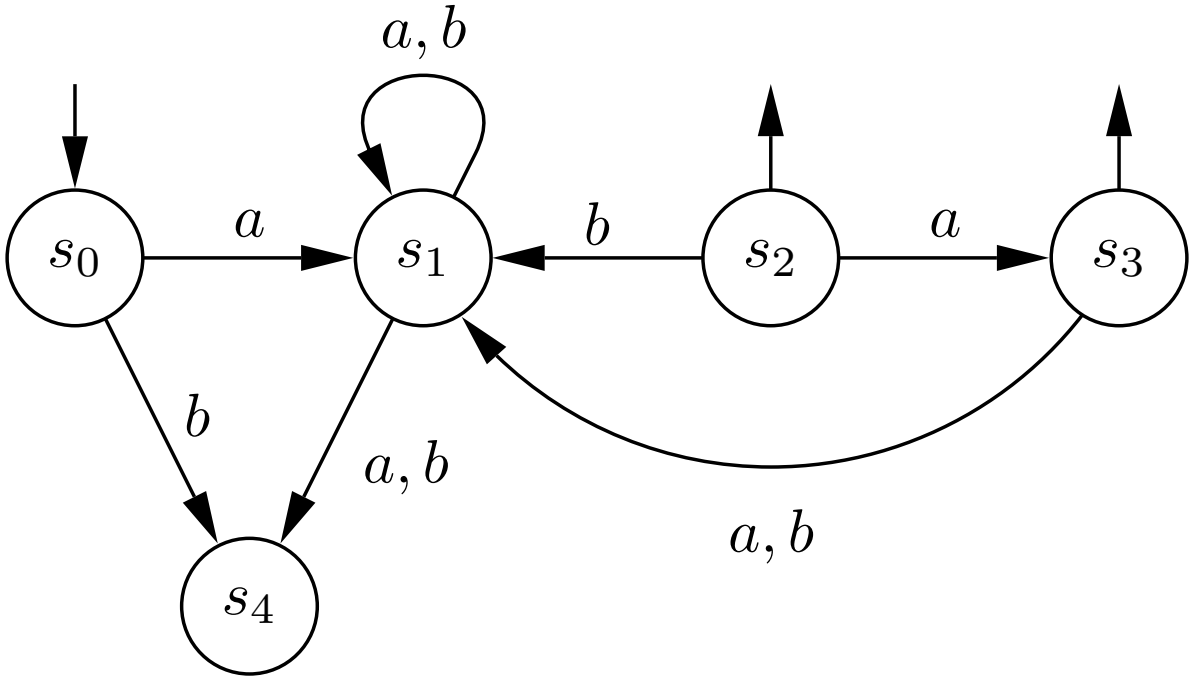
$$L(A_1 \cup A_2) = L(A_1) \cup L(A_2).$$

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

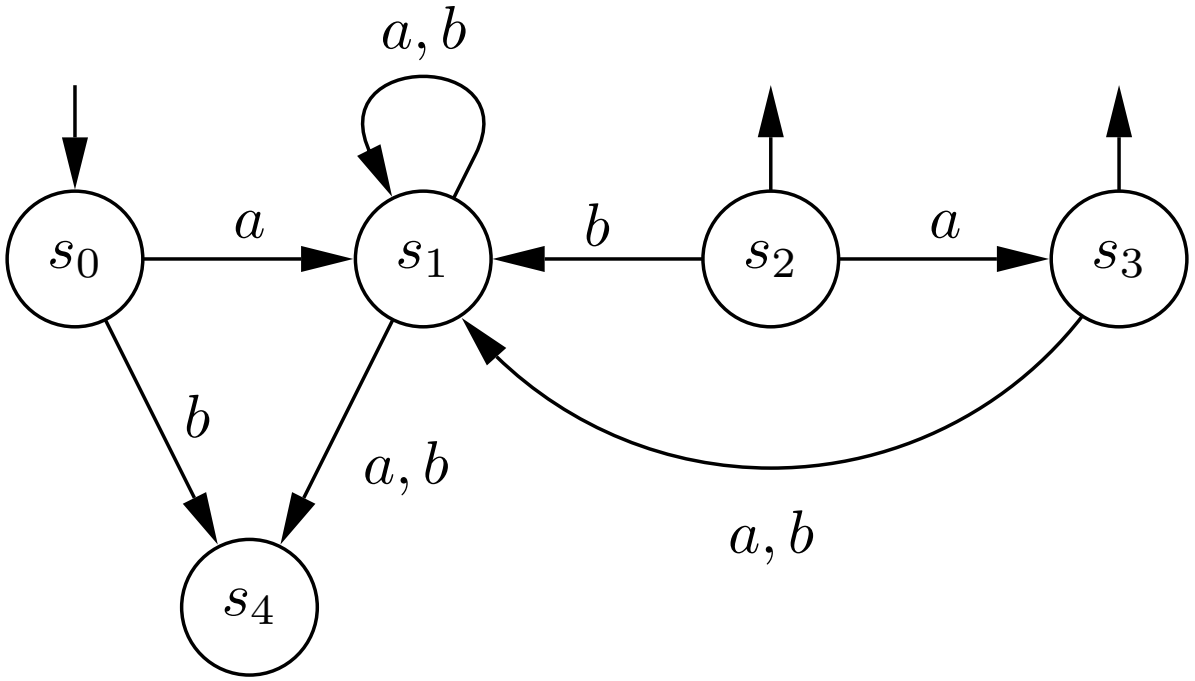
1. Sammle alle vom Startzustand erreichbaren Zustände auf.
2. $L(A) = \emptyset$ genau dann, wenn sich in der Menge der gesammelten Zustände kein Endzustand befindet.

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. $R_m = \{d^*(s_0, w) \mid w \in I^*\}$.