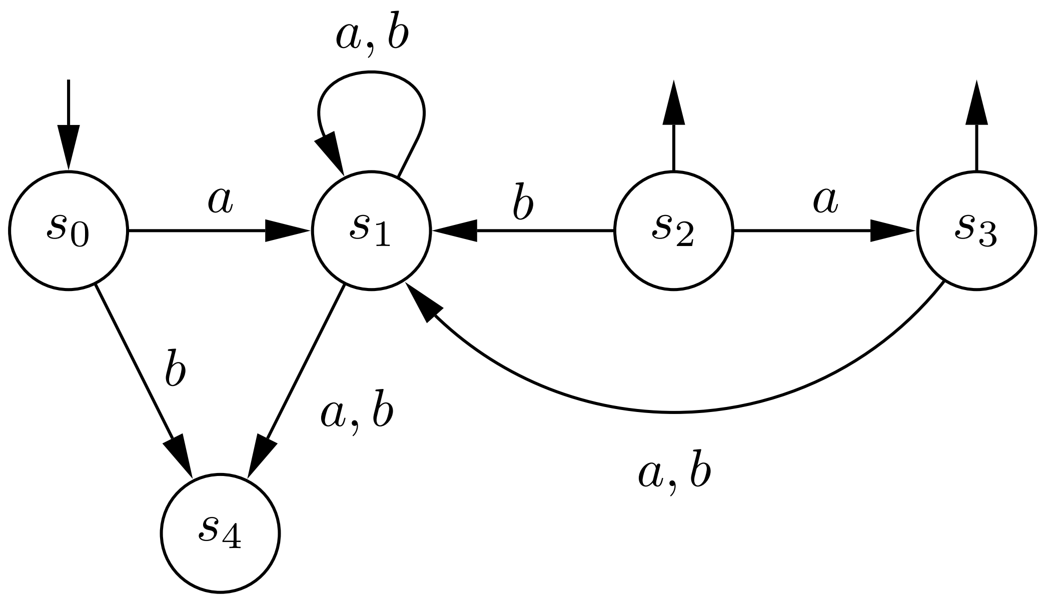
Leerheitsproblem

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

Beispiel

Eingabe:



Ausgabe: True

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:

$$leer(A) = \begin{cases} True, & \text{falls } L(A) = \emptyset \\ False & \text{sonst} \end{cases}$$

Überlegung

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

Idee

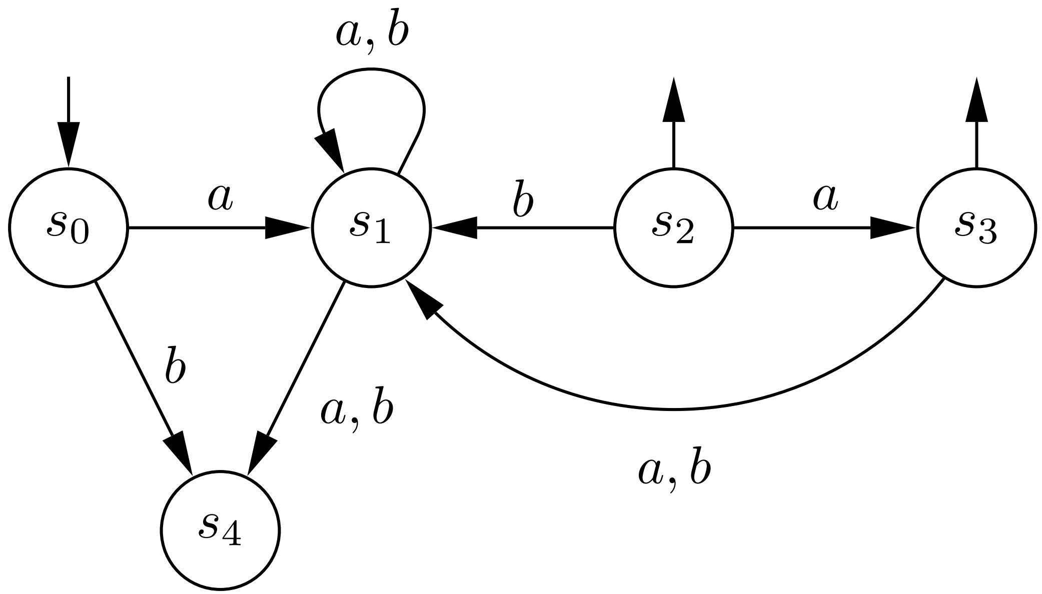
1. Sammle alle vom Starzustand erreichbaren Zustände auf.
2. $L(A) = \emptyset$ genau dann, wenn sich in der Menge der gesammelten Zustände kein Endzustand befindet.

Algorithmus (Skizze)

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

1. (Aufsammeln der erreichbaren Zustände)
   (a) $R_0 := \{s_0\}$; $i := 0$
   (b) $R_{i+1} := R_i \cup \{d(s, x) \mid s \in R_i, x \in I\}$
   (c) If $R_{i+1} = R_i$ then weiter bei 2. else $i := i + 1$ und weiter bei b
2. (Entscheiden)
   If $R_i \cap F = \emptyset$ then True else False

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 erkannte Sprache ist leer.

Korrektheit

Termination

Es existiert ein $m \in \mathbb{N}$: $R_m = R_{m+1}$.

Partielle Korrektheit

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

Modularisierung und Kompositionalität

- große DV-Systeme werden aus kleineren Komponenten aufgebaut, die selbst aus Unterkomponenten bestehen können usw.
  - welche Bausteine eignen sich?
  - wie können sie zusammengesetzt werden?
  - wie wirkt sich eine Komponente auf das Ganze aus?
- endliche Automaten als Bausteine haben exellente Kompositionalitätseigenschaften.
  - Vereinigung, Konkatenation, Kleene-Hülle, Durschschnitt, Differenz

Syntax	Semantik
endliche Beschreibungen datenverarbeitender Systeme (von Mensch und Computer bearbeitbar)	Systemverhalten, mögliche Abläufe und Berechnungen (das eigentlich Interessierende, oft unendlich)
Programme	berechnete Funktionen
endliche Automaten	erkannte Sprachen

- Was verrät Syntax über Semantik? (Z.B. Wortproblem für endliche Automaten ist lösbar.)
- Wie lässt sich syntaktisch semantischer Effekt festlegen? (Z.B. Produktautomat erkennt Durchschnitt; die erkannten Sprachen sind regulär.)

Reguläre Sprachen

Sprachoperationen

Gegeben: I: Alphabet, $L, L_1, L_2 \subseteq I^*$

Vereinigung: $L_1 \cup L_2 = \{w \in I^* \mid w \in L_1 \text{ oder } w \in L_2\}$

Konkatenation: $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$

Kleene-Hülle: $L^* = \bigcup_{i \in \mathbb{N}} L^i$ mit $L^0 = \{\lambda\}$ und $L^{i+1} = L^i L$

Reguläre Sprachen

Definition

Die Menge $\mathcal{L}_{REG(I)}$ der regulären Sprachen ist rekursiv wie folgt definiert:

- $\emptyset, \{\lambda\}, \{x\} \in \mathcal{L}_{REG(I)}$ für $x \in I$,
- $L, L_1, L_2 \in \mathcal{L}_{REG(I)}$ impliziert $L_1 \cup L_2$, L_1L_2, $L^* \in \mathcal{L}_{REG(I)}$.

Beispiel

$(\{\lambda\} \cup (\{a\}\{b\}\{b\}))^* = (\{\lambda\} \cup \{abb\})^* = \{(abb)^n \mid n \in \mathbb{N}\}$.

Induktionsprinzip für reguläre Sprachen

I.A. Zeige THEOREM für $\emptyset$, $\{\lambda\}$ und alle $\{x\}$ mit $x \in I$.

I.V. Nimm THEOREM für reguläre Sprachen L, L_1 und L_2 an.

I.S. Zeige THEOREM für $L_1 \cup L_2$, L_1L_2 und L^*.

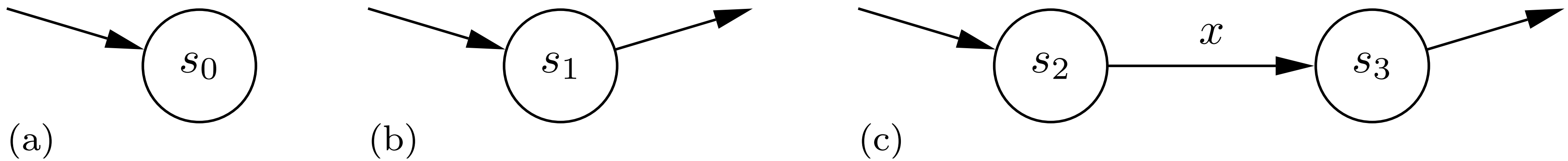
Satz

Jede reguläre Sprache wird von einem endlichen Automaten erkannt.

Beweisskizze (Induktion)

Induktionsanfang

Endliche Automaten für $\emptyset$, $\{\lambda\}$ und $\{x\}$:



Induktionsvoraussetzung

Seien $L, L_1, L_2 \in \mathcal{L}_{REG(I)}$. Dann existieren endliche Automaten A, A_1 und A_2 mit $L(A) = L$,$L(A_1) = L_1$ und $L(A_2) = L_2$.

Induktionsschluss

Zu zeigen: Es existieren endliche Automaten $A_1 \cup A_2$, $A_1 \circ A_2$ und A_*, so dass

1. $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$
2. $L(A_1 \circ A_2) = L(A_1)L(A_2)$
3. $L(A_*) = L(A)^*$

1. Gegeben: nichtdeterministische endliche Automaten $A_1 = (Z_1, I, d_1, s_{0_1}, F_1)$ und $A_2 = (Z_2, I, d_2, s_{0_2}, F_2)$ mit $Z_1 \cap Z_2 = \emptyset$.

$A_1 \cup A_2 = (Z_1 \cup Z_2 \cup \{s_0\}, I, d, s_0, F)$ mit

- $s_0 \notin Z_1 \cup Z_2$;
- $F = \begin{cases} F_1 \cup F_2 \cup \{s_0\}\}, & \text{falls } s_{0_i} \in F_i, i = 1, 2 \\ F_1 \cup F_2 \text{ sonst;} \end{cases}$
- $d = d_1 \cup d_2 \cup \{(s_0, x, s) \mid (s_{0_i}, x, s) \in d_i, i = 1, 2\}$

Bleibt zu zeigen: $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$.

2. Gegeben: nichtdeterministische endliche Automaten $A_1 = (Z_1, I, d_1, s_{0_1}, F_1)$ und $A_2 = (Z_2, I, d_2, s_{0_2}, F_2)$ mit $Z_1 \cap Z_2 = \emptyset$.

$A_1 \circ A_2 = (Z_1 \cup Z_2, I, d, s_{0_1}, F)$ mit

- $F = \begin{cases} F_1 \cup F_2, & \text{falls } s_{0_2} \in F_2 \\ F_2 & \text{sonst.} \end{cases}$
- $d = d_1 \cup d_2 \cup \{(s', x, s) \mid (s_{0_2}, x, s) \in d_2, s' \in F_1\}$

Bleibt zu zeigen: $L(A_1 \circ A_2) = L(A_1)L(A_2)$.

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

$$\boxed{\begin{array}{l} A_* = (Z \cup \{s_*\}, I, d_*, s_*, F \cup \{s_*\}) \text{ mit } s_* \notin Z \text{ und} \\[2em] \qquad d' = d \cup \{(s', x, s) \mid (s_0, x, s) \in d, s' \in F \cup \{s_*\}\} \end{array}}$$

Bleibt zu zeigen: $L(A_*) = L(A)^*$.