4.2.4 Reguläre Grammatiken

Eine reguläre Grammatik ist eine kontextfreie Grammatik, deren Produktionsregeln weiter eingeschränkt sind

- Linksreguläre Grammatik: $\forall A \rightarrow w \in P$ gilt:

$$w = \varepsilon \text{ oder } w = Ba \text{ mit } a \in T \text{ und } B \in N$$

- Rechtsreguläre Grammatik: $\forall A \rightarrow w \in P$ gilt:

$$w = \varepsilon \text{ oder } w = aB \text{ mit } a \in T \text{ und } B \in N$$

Normalerweise wird mit „regulär“ die rechtsregulären Grammatiken gemeint

Theorem: Für jede rechtsreguläre Grammatik G_r gibt es eine linksreguläre Grammatik G_l, so dass $L(G_r) = L(G_l)$. Umgekehrt genauso.

4.2.4 Reguläre Grammatiken

Beispiel: Was ist die entsprechende linksreguläre Grammatik der folgenden rechtsregulären Grammatik G = (N, T, P, S)?

$T = \{ 0, 1 \}$
$N = \{ S, A \}$
$P = \{ S \rightarrow 1A,\ A \rightarrow 0A,\ A \rightarrow 1B,\ B \rightarrow \varepsilon \}$
Startsymbol S

Beispiel: Was ist die entsprechende linksreguläre Grammatik der folgenden rechtsregulären Grammatik G_{ab} = (N, T, P, S)?

$T = \{ a, b \}$
$N = \{ S \}$
$P = \{ S \rightarrow aS,\ S \rightarrow bS,\ S \rightarrow \varepsilon \}$
Startsymbol S

4.2.4 Reguläre Grammatiken

Beispiel: Die folgende von G_{ab} abgeänderte Grammatik G = (N, T, P, S)

T = { a, b }

N = { S }

P = { $S \rightarrow aS$, $S \rightarrow Sb$, $S \rightarrow \varepsilon$ }

Startsymbol S

ist weder links- noch rechtsregulär. $L(G_{ab}) = \{a,b\}^* \neq L(G)$, z.B. $ba \notin L(G)$

Beispiel: Was erzeugt die folgende Grammatik G = (N, T, P, S)?

T = { a, b }

N = { S, U_1, ……, U_{100} }

P = { $S \rightarrow aSU_1$, $U_1 \rightarrow U_2$, ……, $U_{99} \rightarrow U_{100}$, $U_{100} \rightarrow \varepsilon$, $S \rightarrow bS$, $S \rightarrow \varepsilon$ }

Startsymbol S

Ist {a,b}* wirklich eine reguläre (Typ-3) Sprache?

4.2.4 Reguläre Grammatiken

Charakteristische Eigenschaft regulärer Grammatiken:

- Mit einer Regel der Form $S \rightarrow aS$ hat man keine Kontrolle darüber, wie oft sie zur Ableitung angewandt wird, daher kann man alle Wörter a^n, $n \geq 0$, erzeugen
- Es ist nicht die Größe der zu erzeugenden Sprache, die beim Formulieren einer Grammatik Probleme bereitet, sondern die Feinheit der Unterscheidung zwischen erlaubten und auszuschließenden Wörtern:
  - $a^n b^m$ kann durch eine reguläre Grammatik erzeugt werden
  - Die Restriktion n = m kann aber nicht erzwungen werden für beliebig lange Wörter
- $\forall k (\geq 0)$ kann man eine reguläre Grammatik formulieren, die alle $a^n b^n$ mit $n \leq k$ erzeugt, indem man die endlich vielen Produktionen für alle Fälle von 0 bis k direkt in die Grammatik schreibt!

4.3 Automaten

Eine Grammatik G definiert eine Sprache L(G). Aus praktischer Sicht ist das Entscheidungsproblem von großer Bedeutung.

> Gegeben sei eine Grammatik G mit Terminal-Alphabet T und ein Wort w ∈ T^*. Ist w ∈ L(G) oder nicht?

Automaten verschiedener Typen lösen dieses Problem

Grammatik	Regeln	Sprachen	Automaten
Typ-0	$\alpha \in V^*NV^*, \beta \in V^*, \alpha \neq \epsilon$	rekursiv aufzählbar	Turingmaschine
Typ-1	$\alpha A\beta \rightarrow \alpha\gamma\beta$ $A \in N, \alpha, \beta, \gamma \in V^*, \gamma \neq \varepsilon$ $S \rightarrow \varepsilon$ ist erlaubt, wenn es keine Regel $\alpha \rightarrow \beta S\gamma$ in P gibt	kontextsensitiv	linear beschränkter Automat
Typ-2	$A \rightarrow \gamma, A \in N, \gamma \in V^*$	kontextfrei	nichtdeterministischer Kellerautomat
Typ-3	$A \rightarrow aB$ (rechtsregulär) oder $A \rightarrow Ba$ (linksregulär) $A \rightarrow \epsilon, A, B \in N$	regulär	endlicher Automat

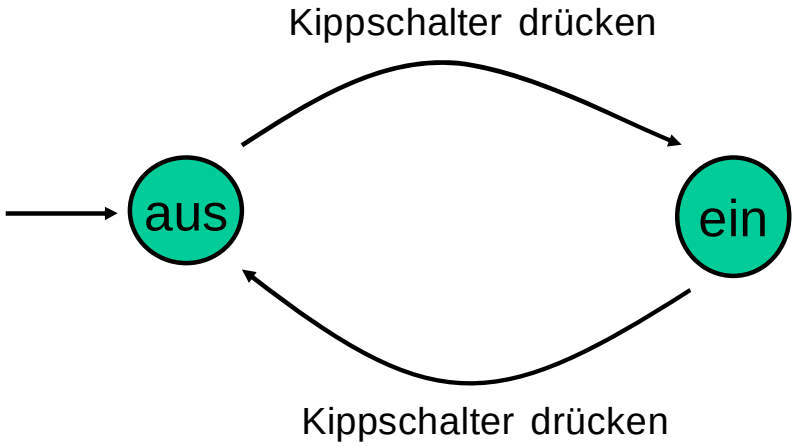
4.3.1 Endliche Automaten

Modellierung einfacher Automaten (Schalter, Fahrkarten, etc.)

- Definierter Ablauf von Aktionen
- Determinierte Folge von (akzeptierten) Aktionen

Beispiel: Geräteschalter

Anfangszustand, Endzustand, Eingabefolge. Endliche Automaten dienen zur „Klassifikation“ von Eingabefolgen.



4.3.1 Endliche Automaten

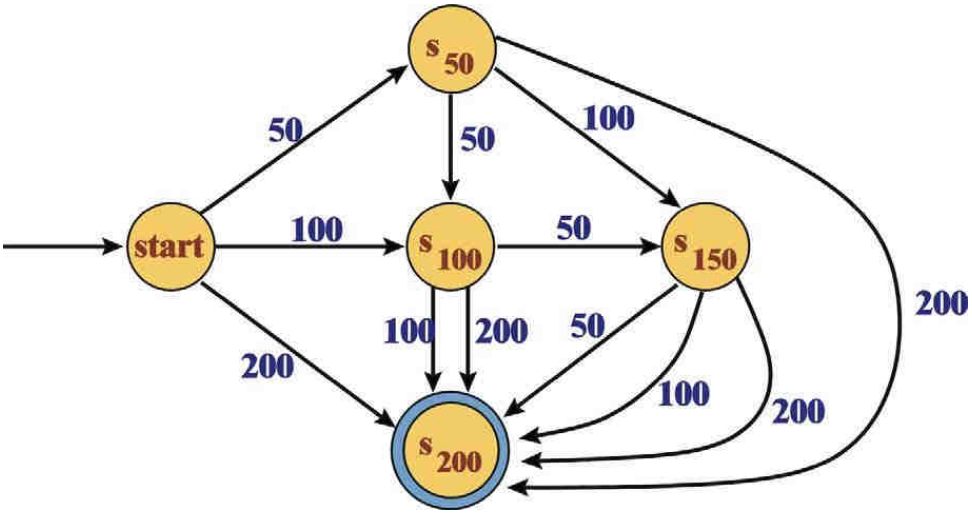
Ausstattung eines endlichen Automaten A:

- Eingabe: A wird von außen mit Eingabedaten versorgt
- Interne Zustände: A befindet sich immer in einem definierten Zustand. Es wird ein Anfangszustand definiert, von dem aus A unter Einfluss der Eingabe Zustandsübergänge durchführt
- Ausgabe: A kann unter gewissen Umständen Ausgabeinformation erzeugen

4.3.1 Endliche Automaten

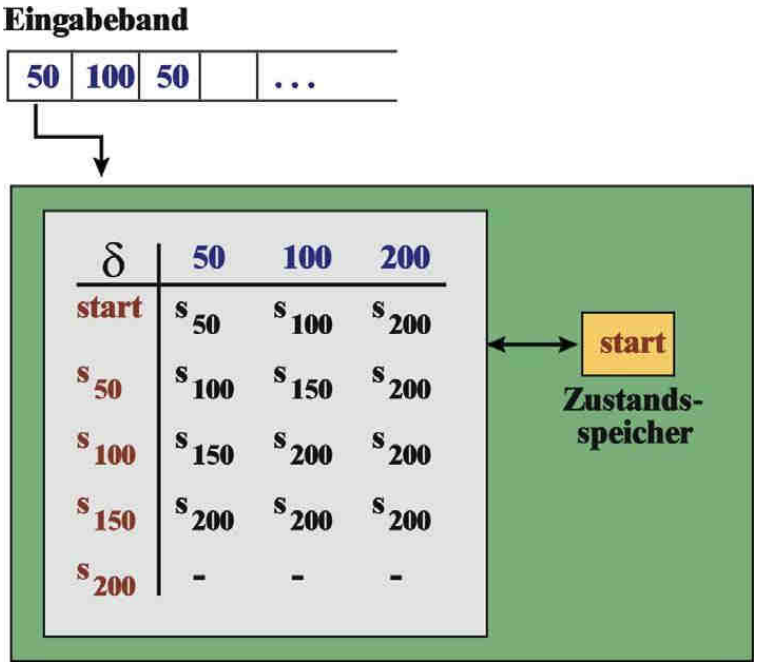
Beispiel: Zustandsdiagramm des Automaten A

- Münzeingaben mit Werten 50, 100, 200 in beliebiger Reihenfolge
- Nach Einwurf von insgesamt ≥ 200 akzeptiert A: Eintritt freigegeben!
- Gesamtwert der bisherigen Eingabe ist im aktuellen Zustand vermerkt



4.3.1 Endliche Automaten

- Eingabeband enthält Eingaben als Folgen von Zeichen
- Zustandsspeicher enthält jeweils aktuellen Zustand
- Programm, Kontrolle: Zustandsübergangsfunktion δ



4.3.1 Endliche Automaten

Definition: Ein deterministischer endlicher Automat (DFA) ist gegeben durch $A = (T, S, \delta, s_0, F)$:

- eine endliche menge S von Zuständen
- eine endliche Menge T von Eingabezeichen
- einen eindeutig ausgezeichneten Anfangszustand $s_0 \in S$
- eine Endzustandsmenge $F \subseteq S$
- eine Übergangsfunktion $\delta : S \times T \rightarrow S$. $\delta(s, a) = t$ spezifiziert den Folgezustand t im Fall von Zustand s und Eingabezeichen $a \in T$

δ kann durch einen Zustandsübergangs-Graphen oder als Menge von Tripeln (s, a, t) mit δ(s, a) = t gegeben sein

δ ist manchmal nicht total (überall definiert)

4.3.1 Endliche Automaten

Beispiel: Endlicher Automat $A = (T, S, \delta, s_0, F)$ zum Erkennen der Menge $\{ (ab)^n, n \geq 0 \}$

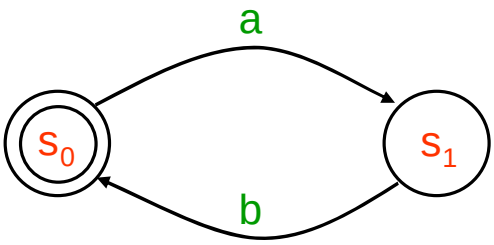
$$T = \{ a, b \}$$
$$S = \{ s_0 , s_1 \}$$
$$F = \{ s_0 \}$$

Beschreibung von δ durch

Übergangsfunktion

δ	s_0	s_1
a	s_1	-
b	-	s_0

Zustandsdiagramm



4.3.1 Endliche Automaten

Erweiterte Übergangsfunktion: Die Zustandsübergangsfunktion δ kann von Zeichen auf Wörter erweitert werden

$$\delta^*: \quad S \times T^* \rightarrow S$$

- $\delta^*(s, \varepsilon) = s$ für alle $s \in S$
- $\delta^*(s, aw) = \delta^*(\delta(s, a), w)$ für alle $a \in T$, $w \in T^*$

Für einen endlichen Automaten $A = (T, S, \delta, s_0, F)$ wird die von A akzeptierte Sprache (die Menge aller von A akzeptierten Eingabefolgen) $L(A) \subseteq T^*$ definiert durch:

$$L(A) = \{ w; \ \delta^*(s_0, w) \in F \}$$

4.3.1 Endliche Automaten

Beispiel: Endlicher Automat zum Erkennen der Menge $\{ (ab)^n, n \geq 0 \}$

Bearbeitung von abab:

$\delta^*(s_0 , abab) = \delta^*(\delta(s_0 , a), bab)$

$= \delta^*(s_1 , bab) = \delta^*(\delta(s_1 , b), ab)$

$= \delta^*(s_0 , ab) = \delta^*(\delta(s_0 , a), b)$

$= \delta^*(s_1 , b) = \delta^*(\delta(s_1 , b), \varepsilon)$

$= \delta^*(s_0 , \varepsilon) = s_0$

⇒ abab wird akzeptiert, da $s_0 \in F$

Bearbeitung von aba:

$\delta^*(s_0 , aba) = \delta^*(\delta(s_0 , a), ba)$

$= \delta^*(s_1 , ba) = \delta^*(\delta(s_1 , b), a)$

$= \delta^*(s_0 , a) = \delta^*(\delta(s_0 , a), \varepsilon)$

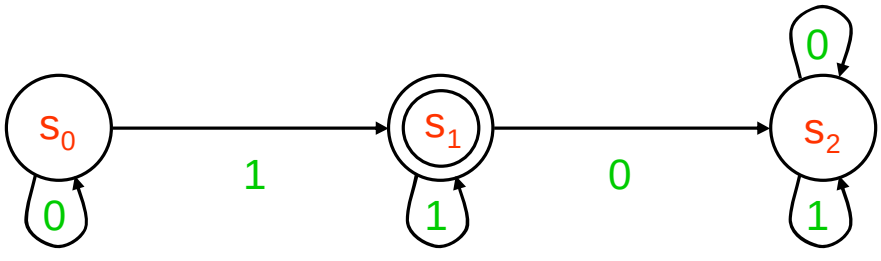
$= \delta^*(s_1 , \varepsilon) = s_1$

⇒ aba wird nicht akzeptiert, da $s_1 \notin F$

4.3.1 Endliche Automaten

Ein endlicher Automat kann einen Endzustand zwischendurch betreten und danach wieder verlassen

Beispiel: Erkennen von $\{ w \in \{ 0, 1 \}^*: w = 0^n1^m \text{ mit } n \geq 0, m > 0 \}$



Weglassen des Zustands s_2 und aller mit ihm verbundenen Kanten würde an der erkannten Sprache nichts ändern!

- Endzustandsmenge $F = \{ s_0 , s_1 \}$ ⇒
  $\{ w \in \{ 0, 1 \}^*: w = 0^n1^m \text{ mit } n \geq 0, m \geq 0 \}$ wird erkannt
- Endzustandsmenge $F = \{ s_2 \}$ ⇒
  $\{ w \in \{ 0, 1 \}^*: w = 0^n1^m0w' \text{ mit } n \geq 0, m > 0, w' \in \{ 0, 1 \}^* \}$ wird erkannt

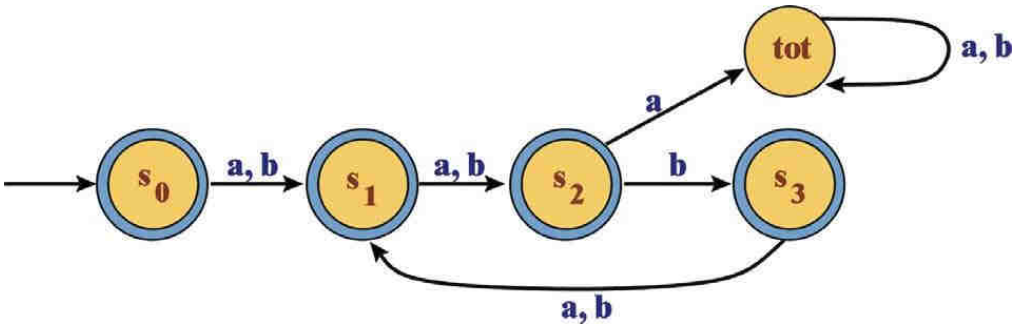
4.3.1 Endliche Automaten

Bisher muss die Funktion δ eines DFA nicht total sein

Ein DFA A = (T, S, δ, s_0, F) heisst vollständig, wenn dom(δ) = S x T.
D.h. δ(s, a) = t ist definiert für alle Paare (s, a).

Jeder DFA kann durch Hinzunahme eines Zustands tot vervollständigt werden: Wenn δ(s, a) nicht definiert ist, ergänze δ(s, a) = tot

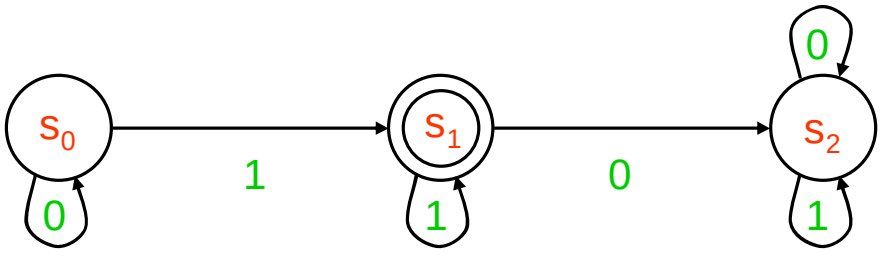
Beispiel:



4.3.1 Endliche Automaten

Lemma: Für jeden DFA A existiert eine reguläre Grammatik G mit L(A)=L(G)

Beispiel: DFA zum Erkennen von { w ∈ { 0, 1 }*: w = 0^n1^m mit n ≥ 0, m > 0 }



Reguläre Grammatik G = (N, T, P, S):

N = {s_0, s_1, s_2}, T = {0, 1}, S = s_0

P = { $s_0 \rightarrow 0s_0$, $s_0 \rightarrow 1s_1$, $s_1 \rightarrow 1s_1$, $s_1 \rightarrow \varepsilon$, $s_1 \rightarrow 0s_2$, $s_2 \rightarrow 0s_2$, $s_2 \rightarrow 1s_2$ }

erkennt genau dieselbe Sprache

4.3.1 Endliche Automaten

Lemma: Für jede reguläre Grammatik G existiert ein DFA A mit L(G)=L(A)

Theorem: Reguläre Grammatiken und deterministische endliche Automaten sind bezüglich ihrer Sprachen gleich mächtig

Endliche Automaten liefern die operativen Werkzeuge zum Lösen des Entscheidungsproblems für reguläre Grammatiken. Dieselbe Funktion erfüllen nichtdeterministische Kellerautomaten (kontextfreie Grammatiken), linear beschränkte Automaten (kontextsensitive Grammatiken), und Turingmaschine (Typ-0 Grammatiken).