

# Kapitel 1

## Endliche Automaten

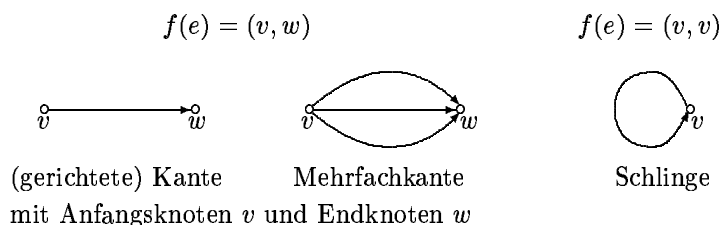
### 1.1 Halbautomaten und Akzeptorsprachen

Endlicher, deterministischer, vollständiger *Halbautomat*:  $(X, S, t)$  mit

$X = \{x_1, x_2, \dots, x_n\} \neq \emptyset$  Eingabealphabet  
 $S = \{s_1, s_2, \dots, s_m\} \neq \emptyset$  Menge von Zuständen und  
 $t : S \times X \rightarrow S$  Überföhrungsfunktion

*Gerichteter Graph*:  $G = \langle V, E, f \rangle$  mit *Knotenmenge*  $V$  (vertex), *Kantenmenge*  $E$  (edge) und *Inzidenzabbildung*  $f : E \rightarrow V \times V$

$G = \langle V, E, f \rangle$  gerichteter Graph,  $e \in E, v, w \in V$



*Zustandsgraph eines Halbautomaten*  $(X, S, t)$ :  $G = \langle S, \{(s_i, s_j) \mid \text{es gibt ein } x_k \in X \text{ mit } t(s_i, x_k) = s_j\}, f \rangle$  (ohne Mehrfachkanten) mit (injektiver) Inzidenzabbildung  $f$ ; dient zur graphischen Darstellung von  $t$ .

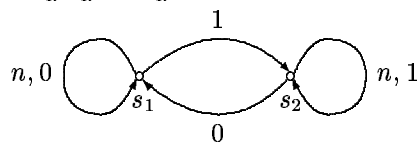
Neben jeder Kante  $(s_i, s_j)$  werden alle  $x_k \in X$  vermerkt, die  $s_i$  in  $s_j$  überföhren (für die  $t(s_i, x_k) = s_j$  gilt).

**Beispiel** IR-Flip-Flop

$X = \{n, 0, 1\}, S = \{s_1, s_2\}$ , Überföhrungsfunktion:

$t$	$n$	$0$	$1$
$s_1$	$s_1$	$s_1$	$s_2$
$s_2$	$s_2$	$s_1$	$s_2$

Zustandsgraph des Halbautomaten:



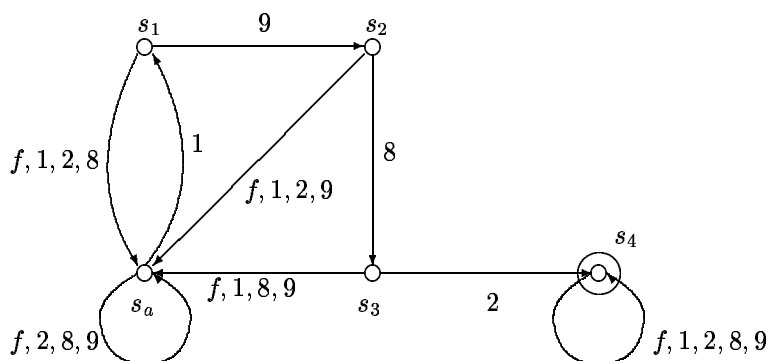
*Akzeptor*  $(X, S, t, s_a, S_E)$ : Halbautomat  $(X, S, t)$  mit einem *Anfangszustand*  $s_a \in S$  und einer Menge  $S_E \subseteq S$  von Endzuständen. Die Endzustände werden im Zustandsgraphen durch kleine Kreise markiert.

### Beispiel Arbeitsweise eines Schlosses

Man konstruiere einen Halbautomaten, der die Arbeitsweise eines Schlosses an einem Tresor simuliert. Das Schloss bestehe aus einem Drehknopf mit den Markierungen  $0, 1, 2, \dots, 9$  und öffnet den Tresor durch Eingabe der Ziffernfolge  $1, 9, 8, 2$ . (Ist eine Ziffer falsch eingestellt worden, so muss mit der Eingabe der gesamten Ziffernkombination von neuem begonnen werden.)

Die Zustände des Schlosses seien:  $s_a$ : das Schloss versperrt den Tresor,  $s_1$ : 1 wurde eingestellt,  $s_2$ : die Folge  $1, 9$  wurde eingestellt,  $s_3$ : die Folge  $1, 9, 8$  wurde eingestellt,  $s_4$ :  $1, 9, 8, 2$  wurde eingestellt. Bei  $s_4$  ist das Schloss geöffnet, d.h.  $s_4$  ist der einzige (zum Zweck des Öffnens) angestrebte Endzustand. Gesucht ist also ein Akzeptor  $(X, S, t, s_a, S_E)$  mit  $S = \{s_a, s_1, s_2, s_3, s_4\}$ ,  $S_E = \{s_4\}$  und  $X = \{0, 1, 2, \dots, 9\}$ .

Schreiben wir zur Abkürzung für jede Ziffer  $\neq 1, 9, 8, 2$  stellvertretend  $f$ , so können wir die Übergangsfunktion  $t$ , welche die Arbeitsweise des Schlosses wiedergibt, aus dem im folgenden dargestellten Zustandsgraphen entnehmen. Damit ist der gesuchte Akzeptor vollständig bestimmt.



Eine Ziffernfolge, welche das Schloss öffnet, ist z.B.  $3, 2, 1, 9, 5, 8, 2, 1, 9, 8, 2, 3$ .

*Wort*: endliche Folge von Eingabesymbolen

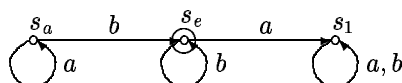
Die Menge aller Wörter, welche den Anfangszustand  $s_a$  in einen Endzustand aus  $S_E$  überführt, heißt *Akzeptorsprache*.

Abkürzung:  $a^r b^s c^t \dots$  für  $\underbrace{a, \dots, a}_{r\text{-mal}} \underbrace{b, \dots, b}_{s\text{-mal}} \underbrace{c, \dots, c}_{t\text{-mal}} \dots$

Der Kürze halber lassen wir zumeist die Beistriche zwischen den einzelnen Symbolen eines Wortes weg.

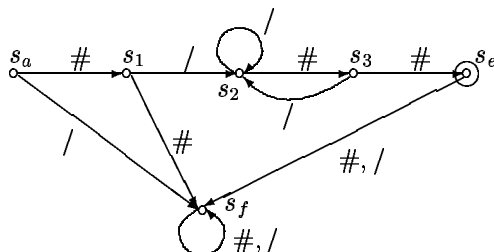
### Beispiel

1. Zustandsgraph eines Akzeptors  $(\{a, b\}, \{s_a, s_1, s_e\}, t, s_a, s_e)$ ; zugehörige Akzeptorsprache  $\{a^i b^j / i \geq 0, j \geq 1\}$ .



2. Sei  $M$  die Menge von Wörtern  $w$  über dem Alphabet  $X = \{\#, / \}$ , die mit genau einem  $\#$  beginnen und mit genau zwei  $\#$  enden, dazwischen enthalte  $w$  stets eine nichtleere Sequenz von  $/$  und  $\#$ .

Beispiel für einen Halbautomaten (Akzeptor), der  $M$  akzeptiert:



**Formale Sprache:** Menge von Wörtern aus Symbolen aus einer Menge  $X$  (einschließlich des "leeren Wortes"  $\Lambda$ )

Eine Akzeptorsprache ist eine *Formale Sprache*.

**Grammatik:** besteht aus einer endlichen Menge  $X$  (*Terminals, Basiszeichen*), einer dazu disjunkten endlichen Menge  $\Theta$  (*Nonterminals, grammatikalische Symbole*) und *Produktionen (Ersetzungsregeln)*  $g \rightarrow w$  (Wort  $g \neq \Lambda$  aus Terminals wird ersetzt durch Wort  $w$  aus Terminals und Nonterminals;  $w$  auch  $= \Lambda$ )

In  $\Theta$  ist ein spezielles Symbol als Startsymbol  $\alpha$  ausgezeichnet.

Sei  $v := w_1 g w_2$  mit  $w_1, w_2$  Wörter aus  $X \cup \Theta$  (einschließlich  $\Lambda$ ),  $g \in \Theta$ ,  $g \neq \Lambda$ . Gilt  $g \rightarrow w$ , so heißt  $v_1 := w_1 w w_2$  aus  $v$  herleitbar ( $v$  in  $v_1$  überführbar,  $v_1$  (unmittelbar) ableitbar aus  $v$ ), in Zeichen  $v \Rightarrow v_1$ .

$v \xRightarrow{*} v_r$  ( $v_r$  ist aus  $v$  ableitbar): Es gibt Wörter  $v_1, \dots, v_{r-1}$ , so dass  $v \Rightarrow v_1, v_1 \Rightarrow v_2, \dots, v_{r-1} \Rightarrow v_r$ .

Die Menge aller aus  $\alpha$  ableitbaren Wörter heißt *die von der Grammatik erzeugte Sprache*.

**Beispiel**  $X = \{a, b\}$ ,  $\Theta = \{A, B, \alpha\}$ , Produktionen:  $\alpha \rightarrow aA, \alpha \rightarrow bB, \alpha \rightarrow a, A \rightarrow aA, A \rightarrow a\alpha, A \rightarrow bB, B \rightarrow bB, B \rightarrow b, B \rightarrow a$ .

Aus  $\alpha \rightarrow aA$  und  $A \rightarrow aA$  folgt  $\alpha \xRightarrow{*} a^k A, k \geq 1$ . Weiters folgt mit  $A \rightarrow a\alpha$  und  $\alpha \rightarrow a$ , dass  $\alpha \xRightarrow{*} a^{k+2}$ , für  $k \geq 1$ . Damit gilt wegen  $\alpha \rightarrow a$ , dass  $\alpha \xRightarrow{*} a^n$  mit  $n \neq 2$ . Sei  $W_1 := \{a^n / n \in \mathbb{N}^*, n \neq 2\}$ . ( $\mathbb{N}^* = \{1, 2, 3, \dots\}$ )

Analog erhält man  $\alpha \xRightarrow{*} b^m$  für  $m \neq 1$ ; Es sei  $W_2 := \{b^m / m \in \mathbb{N}^*, m \neq 1\}$ . Weiters sind die Wörter aus  $W_3 := \{a^k b^m / k, m \in \mathbb{N}^*, m \neq 1\}$  aus  $\alpha$  ableitbar, da  $\alpha \xRightarrow{*} a^k A \xRightarrow{*} a^k b B \xRightarrow{*} a^k b^m$  für  $k \geq 1, m \neq 1$ .

Analog gilt für  $W_4 := \{b^k a / k \in \mathbb{N}^*\}$  und  $W_5 := \{a^k b^l a / k, l \in \mathbb{N}^*\}$ .

Die von der gegebenen Grammatik erzeugte Sprache besteht aus  $W_1 \cup W_2 \cup W_3 \cup W_4 \cup W_5$ .

**Reguläre (rechtslineare, rechtsreguläre) Grammatik:** Grammatik, in der alle Produktionen die Form  $U \rightarrow x, U \rightarrow xV$  (mit  $U, V \in \Theta$  und  $x \in X$ ) haben

**Rechtslineare Sprache:** eine von einer rechtslinearen Grammatik erzeugte Sprache  
Sprachen, welche durch rechtslineare Grammatiken erzeugt werden = Sprachen, welche durch *linkslineare Grammatiken* erzeugt werden (Definition von linkslinear analog zu rechtslinear) = *reguläre Sprachen*

**Satz 1** (ohne Beweis) Eine formale Sprache ist genau dann regulär, wenn sie eine Akzeptorsprache ist.

Die Theorie der Halbautomaten ist eng mit der Theorie der Halbgruppen verknüpft.

*Halbgruppe*  $\langle H, \circ \rangle$ : Grundmenge  $H$  mit einer assoziativen zweistelligen Operation  $\circ$ , d.h.  $(a \circ b) \circ c = a \circ (b \circ c) \forall a, b, c \in H$

*Neutrales Element*: Element  $n \in H$ , für das  $a \circ n = n \circ a = a \forall a \in H$  ("Einselement", wenn  $\circ = \cdot$ )

*Monoid*  $\langle H, \circ, n \rangle$ : Halbgruppe mit neutralem Element, kurz:  $H$

*Untermonoid*: Teilmenge  $T$  eines Monoids  $\langle H, \circ, n \rangle$ , so dass  $\langle T, \circ, n \rangle$  selbst ein Monoid ist

**Beispiele** für Monoide

1.  $\langle M^M, \circ, \epsilon \rangle$  mit  
 $M^M$ : Menge aller Abbildungen  $f : M \rightarrow M$  einer Menge  $M$  in sich,  
 $\circ$ : Hintereinanderausführung von Abbildungen  $((f \circ g)(x) := f(g(x)) \forall x \in M)$ ,  
 $\epsilon$ : identische Abbildung
2. Menge aller bijektiven Abbildungen (Permutationen) von  $M$  bildet ein Untermonoid von  $M^M$
3.  $\langle W, \circ, \Lambda \rangle$  mit  
 $W$ : Menge aller Wörter aus Elementen von  $X = \{x_1, x_2, \dots, x_n\}$ ,  
 $\circ$ :  $w_i \circ w_j := x_{i_1} x_{i_2} \dots x_{i_r} x_{j_1} x_{j_2} \dots x_{j_s}$  für  
 $w_i = x_{i_1} x_{i_2} \dots x_{i_r}, w_j = x_{j_1} x_{j_2} \dots x_{j_s} \in W$ ,  
 $\Lambda$ : leeres Wort,  $w \circ \Lambda = \Lambda \circ w = w \forall w \in W$

Sei  $W$  die *Menge der Inputfolgen*, d.h. die Menge aller Wörter (einschließlich  $\Lambda$ ) aus Eingabesymbolen eines Halbautomaten  $(X, S, t)$ .

$t : S \times X \rightarrow S$  kann wie folgt rekursiv zu einer Abbildung  $t^* : S \times W \rightarrow S$  fortgesetzt werden: Für alle  $s \in S, x, x_{i_1}, x_{i_2}, \dots, x_{i_k} \in X$  sei

$$t^*(s, \Lambda) = s$$

$$t^*(s, x) = t(s, x)$$

$$t^*(s, x_{i_1}, x_{i_2}) = t(t^*(s, x_{i_1}), x_{i_2})$$

$\vdots$

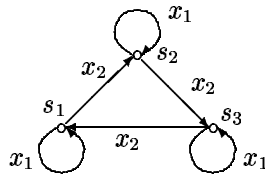
$$t^*(s, x_{i_1}, x_{i_2}, \dots, x_{i_k}) = t(t^*(s, x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}}), x_{i_k})$$

Zustand  $s \in S$  wird durch Eingabe der Inputfolge  $w = x_{i_1}, x_{i_2}, \dots, x_{i_k} \in W$  in den Zustand  $t^*(s, x_{i_1}, x_{i_2}, \dots, x_{i_k})$  übergeführt

Definiere für jede Inputfolge  $w \in W$  eine Abbildung  $g_w : S \rightarrow S, g_w(s) = t^*(s, w)$ .  $\circ$ : Hintereinanderausführung von Abbildungen, wobei  $g_{w_2 w_1}(s) := (g_{w_1} \circ g_{w_2})(s) = g_{w_1}(g_{w_2}(s))$  für  $w_1, w_2 \in W, s \in S$ .

*Halbgruppe des Halbautomaten (zum Halbautomaten gehöriges Monoid)*:  $\langle \{g_w/w \in W\}, \circ, g_\Lambda \rangle$  (Untermonoid von  $S^S$ )

**Beispiel** Zustandsgraph eines Halbautomaten



$g_{x_1}(s) = t(s, x_1) = s \forall s \in S \Rightarrow g_{x_1} = g_\Lambda \Rightarrow$  Verknüpfung mit  $g_{x_1}$  kann keine neue Abbildung ergeben

$$g_{x_2}(s) = \begin{cases} s_2 & \text{für } s = s_1 \\ s_3 & \text{für } s = s_2 \\ s_1 & \text{für } s = s_3 \end{cases}, g_{x_2x_2} = \begin{cases} s_3 & \text{für } s = s_1 \\ s_1 & \text{für } s = s_2 \\ s_2 & \text{für } s = s_3 \end{cases}, g_{x_2x_2x_2} = g_{x_1} = g_\Lambda$$

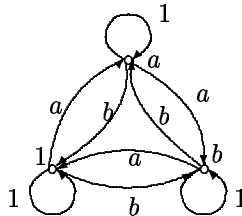
	$\circ$	$g_\Lambda$	$g_{x_2}$	$g_{x_2x_2}$
Halbgruppe des Halbautomaten:	$g_\Lambda$	$g_\Lambda$	$g_{x_2}$	$g_{x_2x_2}$
	$g_{x_2}$	$g_{x_2}$	$g_{x_2x_2}$	$g_\Lambda$
	$g_{x_2x_2}$	$g_{x_2x_2}$	$g_\Lambda$	$g_{x_2}$

**Satz 2** Sei  $\langle M, \cdot, 1 \rangle$  ein beliebiges endliches Monoid. Definiert man  $\bar{H} := (M, M, t)$  mit  $t(s, x) = s \cdot x \forall s, x \in M$ , so ist  $\bar{H}$  ein Halbautomat, dessen Monoid bis auf die Bezeichnung mit dem gegebenen Monoid  $\langle M, \cdot, 1 \rangle$  übereinstimmt ("isomorph" zum gegebenen Monoid ist).

**Beispiel** Monoid  $\langle \{a, b, 1\}, \cdot, 1 \rangle$  mit

$\cdot$	1	a	b
1	1	a	b
a	a	b	1
b	b	1	a

Zustandsgraph von  $\bar{H}$ :



Wir sehen: Verschiedene Halbautomaten können dasselbe Monoid haben.

## 1.2 Automaten

*Endlicher, deterministischer, vollständiger Automat*  $(X, S, Z, t, r)$ : Halbautomat  $(X, S, t)$ , bei dem auch Zeichen aus dem Ausgabealphabet  $Z$  ausgegeben werden; Ausgabefunktion  $r : S \times X \rightarrow Z$

$W$  sei Menge aller Inputfolgen

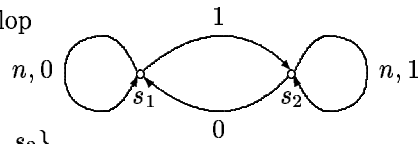
$r$  wird wie folgt zu einer Abbildung  $r^* : S \times W \rightarrow$  (Menge aller Outputfolgen)

fortgesetzt:

$$r^*(s, \Lambda) = \Lambda$$

$$r^*(s, wx) = r^*(s, w)r^*(t^*(s, w), x) \quad \forall s \in S, w \in W, x \in X$$

**Beispiel** IR-Flip-Flop



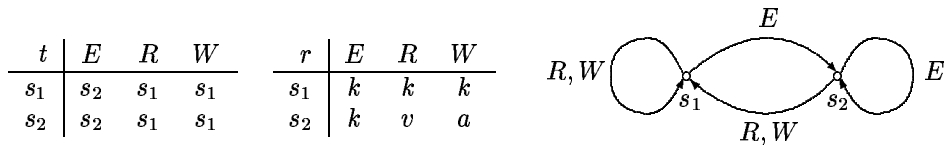
$$X = \{n, 0, 1\}, S = \{s_1, s_2\}$$

$$Z = \{0, 1\}, \text{Ausgabefunktion: } \begin{array}{c|ccc} r & n & 0 & 1 \\ \hline s_1 & 0 & 0 & 0 \\ s_2 & 1 & 1 & 1 \end{array}, r \text{ gibt gespeicherten Zustand}$$

wieder (0 für  $s_1$ , 1 für  $s_2$ )

**Beispiel** Einfacher Warenautomat

$X = \{E, R, W\}$   $E$ : Geld einwerfen,  $R$ : Rückgabeknopf drücken,  $W$ : Warentaste drücken  
 $S = \{s_1, s_2\}$   $s_1$ : Warenausgabe ist blockiert,  $s_2$ : Warenentnahme möglich  
 $Z = \{a, v, k\}$   $a$ : Warenausgabe,  $v$ : Geldrückgabe,  $k$ : keine Reaktion



Automat sei im Zustand  $s_1$ ; Inputfolge  $REWRE$

$$\begin{aligned}
 r^*(s_1, REWRE) &= r^*(s_1, REWR) \underbrace{r^*(t^*(s_1, REWR), w)}_{s_1} \\
 &= r^*(s_1, REW) \underbrace{r^*(t^*(s_1, REW), R)}_{s_1} r^*(s_1, W) \\
 &= r^*(s_1, RE) \underbrace{r^*(t^*(s_1, RE), W)}_{s_2} r^*(s_1, R) r^*(s_1, W) \\
 &= r^*(s_1, R) \underbrace{r^*(t^*(s_1, R), E)}_{s_1} r^*(s_2, W) r^*(s_1, R) r^*(s_1, W) \\
 &= kkakk \quad \text{Output}
 \end{aligned}$$