

Motivation: Automaten für die Modellierung, Spezifikation und Verifikation verwenden!

**Definition** Ein *Endlicher Automat*  $A = (S, I, \Sigma, T, F)$  besteht aus

- Menge von Zuständen  $S$  (normalerweise endlich)
- Menge von Initialzuständen  $I \subseteq S$
- Eingabe-Alphabet  $\Sigma$  (normalerweise endlich)
- Übergangsrelation  $T \subseteq S \times \Sigma \times S$   
schreibe  $s \xrightarrow{a} s'$  gdw.  $(s, a, s') \in T$  gdw.  $T(s, a, s')$  "gilt"
- Menge von Finalzuständen  $F \subseteq S$

**Definition** Der Produkt Automat  $A = A_1 \times A_2$  von zwei EA  $A_1$  und  $A_2$  mit gemeinsamen Eingabealphabet  $\Sigma_1 = \Sigma_2$  hat folgende Komponenten:

$$\begin{aligned} S &= S_1 \times S_2 & I &= I_1 \times I_2 \\ \Sigma &= \Sigma_1 = \Sigma_2 & F &= F_1 \times F_2 \\ T((s_1, s_2), a, (s'_1, s'_2)) & \text{ gdw. } & T_1(s_1, a, s'_1) & \text{ und } T_2(s_2, a, s'_2) \end{aligned}$$

**Satz** Seien  $A, A_1,$  und  $A_2$  wie oben, dann  $L(A) = L(A_1) \cap L(A_2)$

**Beispiel:** Konstruktion eines Automaten, der Wörter mit Prefix  $ab$  und Suffix  $ba$  akzeptiert.  
(als regulärer Ausdruck:  $a \cdot b \cdot \mathbf{1}^* \cap \mathbf{1}^* \cdot b \cdot a,$  wobei  $\mathbf{1}$  für alle Buchstaben steht)

**Definition** Ein EA  $A$  akzeptiert ein Wort  $w \in \Sigma^*$  gdw. es  $s_i$  und  $a_i$  gibt mit

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots \xrightarrow{a_{n-1}} s_{n-1} \xrightarrow{a_n} s_n,$$

wobei  $n \geq 0, s_0 \in I, s_n \in F$  und  $w = a_1 \dots a_n$  ( $n = 0 \Rightarrow w = \epsilon$ ).

**Definition** Die Sprache  $L(A)$  von  $A$  ist die Menge der Wörter die er akzeptiert.

- benutze Reguläre Sprachen zur Spezifikation von Syntax  
z.B. Scanner eines Parsers
- benutze EA oder reguläre Sprachen zur Beschreibung von Ereignisströmen!

**Definition** Zu  $s \in S, a \in \Sigma$  bezeichne  $s \xrightarrow{a}$  die Menge der Nachfolger von  $s$  definiert als

$$s \xrightarrow{a} = \{s' \in S \mid T(s, a, s')\}$$

**Definition** Ein EA ist *vollständig* gdw.  $|I| > 0$  und  $|s \xrightarrow{a}| > 0$  für alle  $s \in S$  und  $a \in \Sigma$ .

**Definition** ... *deterministisch* gdw.  $|I| \leq 1$  und  $|s \xrightarrow{a}| \leq 1$  für alle  $s \in S$  und  $a \in \Sigma$ .

**Fakt** ... deterministisch und vollständig gdw.  $|I| = 1$  und  $|s \xrightarrow{a}| = 1$  für alle  $s \in S, a \in \Sigma$ .

**Definition** Der Power-Automat  $A = \mathbb{P}(A_1)$  eines EA  $A_1$  hat folgende Komponenten

$$S = \mathbb{P}(S_1) \quad (\mathbb{P} = \text{Potenzmenge}) \quad I = \{I_1\}$$

$$\Sigma = \Sigma_1 \quad F = \{F' \subseteq S_1 \mid F' \cap F_1 \neq \emptyset\}$$

$$T(S', a, S'') \text{ gdw. } S'' = \bigcup_{s \in S'} s \xrightarrow{a}$$

**Satz**  $A, A_1$  wie oben, dann  $L(A) = L(A_1)$  und  $A$  ist deterministisch und vollständig.

**Beispiel:** Spam-Filter basierend auf der White-List “abb”, “abba”, und “abacus”!

(Regulärer Ausdruck: “abb” | “abba” | “abacus”)

**Idee:** ersetze Nicht-Determinismus durch Orakel

**Definition** Der Orakelisierte EA  $A = \text{Orakel}(A_1)$  eines EA  $A_1$  hat die Komponenten:

- $S = S_1$
- $I = I_1$
- $\Sigma = \Sigma_1 \times S_1$
- $T(s, (a, t), s') \text{ gdw. } s' = t \text{ und } T_1(s, a, t)$
- $F = F_1$

**Definition** Der Komplementär-Automat  $A = K(A_1)$  eines endlichen Automaten  $A_1$  hat dieselben Komponenten wie  $A_1$ , bis auf  $F = S \setminus F_1$ .

**Satz** Der Komplementär-Automat  $A = K(A_1)$  eines deterministischen und vollständigen Automaten  $A_1$  akzeptiert die komplementäre Sprache  $L(A) = \overline{L(A_1)} = \Sigma^* \setminus L(A_1)$ .

**Beispiel:** Spam-Filter basierend auf der Black-List “abb”, “abba”, und “abacus”!

(Regulärer Ausdruck: “abb” | “abba” | “abacus”)

**Fakt**  $\pi_1(L(\text{Orakel}(A))) = L(A_1)$  ( $\pi_1$  Projektion auf erste Komponente)

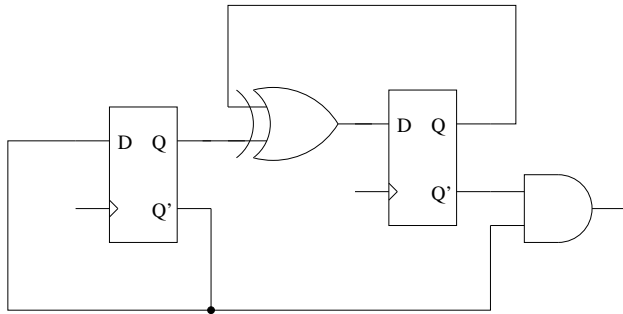
**Fakt**  $\text{Orakel}(A_1)$  ist deterministisch gdw.  $|I_1| \leq 1$ .

**Fakt**  $\text{Orakel}(A_1)$  ist fast immer unvollständig (z.B.  $T_1 \neq S_1 \times \Sigma_1 \times S_1$  und  $|S_1| > 1$ ).

**Hinweis** Vollständigkeit lässt sich erreichen, wenn schon  $A_1$  vollständig ist und man statt  $S_1$  die Menge  $\{0, \dots, n-1\}$  zu  $\Sigma_1$  hinzufügt, wobei  $n$  die maximale Anzahl der Nachfolger darstellt:  $n = \max_{s \in S, a \in \Sigma} |s \xrightarrow{a}|$ .

$$T(s, (a, i), s') \text{ gdw. } s' = s_j, \quad s \xrightarrow{a} = \{s_0, \dots, s_{m-1}\}, \quad j \equiv i \text{ mod } m$$

**Übung** Man führe beide Orakel-Konstruktionen für  $a \cdot b \cdot \mathbf{1}^* \cap \mathbf{1}^* \cdot b \cdot a$  durch.



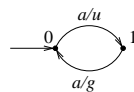
Verhalten eines Ein/Ausgabe-Automaten

Sei  $w \in \Sigma^*$  und  $a \in \Sigma$ .

**Definition** Man interpretiere  $T$  als *erweiterte* Übergangsfunktion  $T \subseteq S \times \Sigma^* \rightarrow S$  wie folgt: es gelte  $s = T(s, \epsilon)$  und  $s' = T(s, a \cdot w) \Leftrightarrow \exists s'' [s'' = T(s, a) \wedge s' = T(s'', w)]$ .

**Definition** Ebenso interpretiere man  $O$  als *erweiterte* Ausgabefunktion  $O: S \times \Sigma^* \rightarrow \Theta^*$ : es gelte  $O(s, \epsilon) = \epsilon$  und  $O(s, a \cdot w) = b \cdot w'$ , mit  $s' = T(s, a)$  und  $w' = O(s', w)$ .

**Definition** Das Verhalten  $V: \Sigma^* \rightarrow \Theta^*$  eines Ein/Ausgabe-Automaten ist definiert durch  $V(w) = O(i, w)$ .

**Beispiel**  $S = \{0, 1\}$ ,  $\Sigma = \{a\}$ ,  $\Theta = \{g, u\}$ , 

$$T(0, a^{2n}) = 0, \quad T(0, a^{2n+1}) = 1, \quad T(1, a^{2n}) = 1, \quad T(1, a^{2n+1}) = 0$$

$$V(a^{2n}) = (ug)^n, \quad V(a^{2n+1}) = (ug)^n u$$

Ein/Ausgabe-Automat

Implementierungen von Automaten müssen deterministisch sein.

**Definition** Ein/Ausgabe Automat  $A = (S, i, \Sigma, T, \Theta, O)$  besteht aus Folgendem:

- einer (endlichen) Menge von Zuständen  $S$ ,
- genau einem Initialzustand  $i$ ,
- einem Eingabealphabet  $\Sigma$ ,
- einer Übergangsfunktion  $T \subseteq S \times \Sigma \rightarrow S$
- einem Ausgabealphabet  $\Theta$ , mit
- Ausgabefunktion  $O: S \times \Sigma \rightarrow \Theta$  (Moore-Maschine:  $O: S \rightarrow \Theta$ )

Ein/Ausgabe-Automat als EA

Gegeben ein Ein/Ausgabe-Automat  $A = (S, i, \Sigma, T, \Theta, O)$ .

**Definition** Der EA zu  $A$  sei definiert als  $A' = (S, \{i\}, \Sigma \times \Theta, T', S)$  mit

$$T'(s, (a, b), s') \text{ gdw. } s' = T(s, a) \text{ und } b = O(s, a).$$

**Fakt**  $V(w) = w'$  gdw.  $(w, w') \in L(A')$

**Fortsetzung des Beispiels:** 

(graphisch fast kein Unterschied)

Gegeben ein EA  $A = (S, I, \Sigma, T, F)$ .

**Definition** Der Ein/Ausgabe-Automat zu  $A$  sei definiert als  $A' = (\mathbb{P}(S), I, \Sigma, T', \{0, 1\}, O)$  mit  $T'$  die Übergangsrelation von  $\mathbb{P}(A)$  und  $O(S', a) = 1$  gdw.  $S' \cap F \neq \emptyset$ .

**Fakt**  $w \in L(A)$  gdw.  $V(w \cdot x) \in \mathbf{1}^{|w|} \cdot 1$  für ein  $x \in \Sigma$

## Fazit des Vergleiches von Ein/Ausgabe-Automat mit EA:

Im wesentlichen stellen beide die gleiche mathematische Struktur dar.

Wir konzentrieren uns auf die kompaktere und elegantere Version des EA.

Insbesondere Nicht-Determinismus lässt sich mit EA besser darstellen.