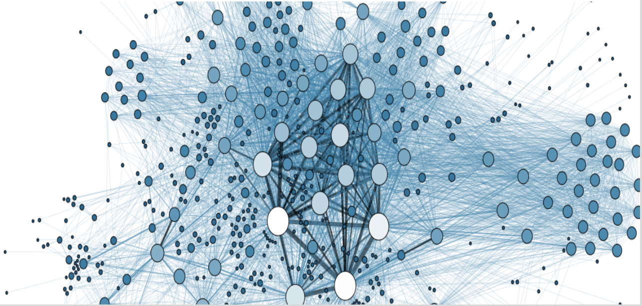
Karlsruher Institut für Technologie

Grundbegriffe der Informatik
Tutorium 38

Automaten und reguläre Sprachen

Patrick Fetzer, `uxkln@student.kit.edu` | 01.02.2018



Mealy-Automat

Mealy-Automat

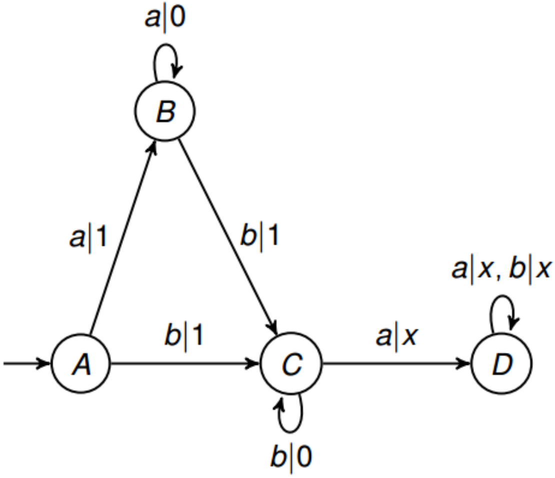
Ein Mealy-Automat ist ein Tupel $A = (Z, z_0, X, f, Y, h)$ mit...

- Z endliche, nichtleere Zustandsmenge
- $z_0 \in Z$ Startzustand
- X Eingabealphabet
- $f : Z \times X \rightarrow Z$ Zustandsübergangsfunktion
- Y Ausgabealphabet
- $h : Z \times X \rightarrow Y^*$ Ausgabefunktion

Darstellung als Graph

- Zustände $\rightarrow$ Knoten
- Startzustand $\rightarrow$ Pfeil an diesen Knoten (nicht vergessen!)
- Zustandsüberführungsfunktion $\rightarrow$ Kanten mit Beschriftung $w \in X$
- Ausgabefunktion $\rightarrow$ zusätzliche Kantenbeschriftung $w \in Y^*$

Beispiel Mealy-Automat



Mealy-Automat

$$f_* : Z \times X^* \to Z$$
$$f_*(z, \varepsilon) := z$$
$$\forall w \in X^* \forall x \in X : f_*(z, wx) := f(f_*(z, w), x)$$

Anschaulich: „Endzustand“

Mealy-Automat

$$f_{**} : Z \times X^* \to Z^*$$
$$f_{**}(z, \varepsilon) := z$$
$$\forall w \in X^* \forall x \in X : f_{**}(z, wx) := f_{**}(z, w) \cdot f(f_*(z, w), x)$$

Anschaulich: „durchlaufene Zustände“

Mealy-Automat

$$g_* : Z \times X^* \to Y^*$$
$$g_*(z, \varepsilon) := \varepsilon$$
$$\forall w \in X^* \forall x \in X : g_*(z, wx) := g(f_*(z, w), x)$$

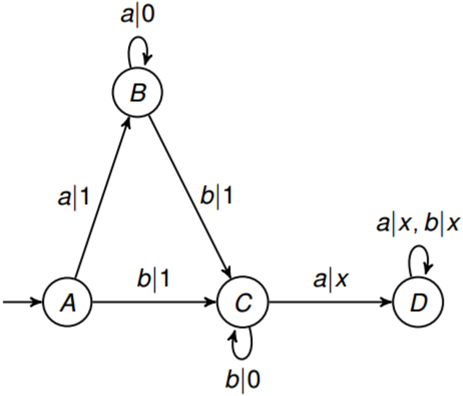
Anschaulich: „letzte Ausgabe“

Mealy-Automat

$$g_{**} : Z \times X^* \to Y^*$$
$$g_{**}(z, \varepsilon) := \varepsilon$$
$$\forall w \in X^* \forall x \in X : f_{**}(z, wx) := g_{**}(z, w) \cdot g_*(z, wx)$$

Anschaulich: „alle Ausgaben konkateniert“

Beispiel Mealy-Automat



- $f_*(A, aabba) = D$
- $f_{**}(A, aabba) = ABBCCD$
- $g_*(A, aabba) = x$
- $g_{**}(A, aabba) = 1010x$

  Für welche $w \in X^*$ gilt $f_*(A, w) \neq D$?

Moore-Automat

Moore-Automat

Ein Moore-Automat ist ein Tupel $A = (Z, z_0, X, f, Y, h)$ mit...

- Z endliche Zustandsmenge
- $z_0 \in Z$ Anfangszustand
- X Eingabealphabet
- $f : Z \times X \to Z$ Zustandsübergangsfunktion
- Y Ausgabealphabet

$\to$ Bis hierhin alles wie bei Mealy!

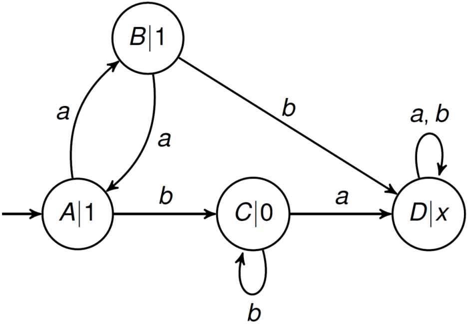
- $h : Z \to Y^*$ Ausgabefunktion

- Zustände $\to$ Knoten
- Startzustand $\to$ Pfeil an diesen Knoten (nicht vergessen!)
- Zustandsüberführungsfunktion $\to$ Kanten mit Beschriftung $w \in X$
- Ausgabefunktion $\to$ zusätzliche Knotenbeschriftung $w \in Y^*$

Funktionen

- f_* und f_{**} wie beim Mealy-Automaten
- $g_* := h \circ f_*$
- $g_{**} := h_{**} \circ f_{**}$ Dabei bezeichnet h_{**} den durch h induzierten Homomorphismus.

Beispiel



- $f_*(A, aabba) = D$
- $f_{**}(A, aabba) = ABACCD$
- $g_*(A, aabba) = x$
- $g_{**}(A, aabba) = 11100x$

  Wann gilt $g_*(A, w) = 0$?
  Genau dann, wenn $w \in \{aa\}^*\{b\}^+$.

Umwandlung

Bemerkung

Für jeden Mealy-Automaten kann man einen Moore-Automaten konstruieren, der genau die gleiche Aufgabe erfüllt, und umgekehrt.

Umwandlung Mealy- in Moore-Automat

Links ein Mealy-, rechts ein Moore-Automat

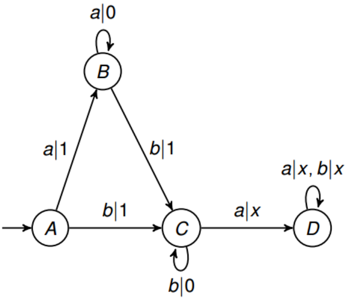


Abbildung: Mealy

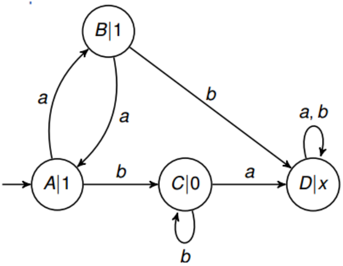


Abbildung: Moore

Endliche Akzeptoren

- Sonderfall von Moore-Automaten
- Bei einem Akzeptor will man nur wissen, ob die Eingabe akzeptiert wurde oder nicht (also reicht ein Bit als Ausgabealphabet)
- Statt der Ausgabefunktion h schreibt man einfach die Menge der akzeptierenden Zustände $F \subseteq Z$ auf
- Zustände, die nicht akzeptieren, heißen ablehnend
- Im Graphen werden akzeptierende Zustände einfach mit einem doppelten Kringel gekennzeichnet 

Akzeptierte Wörter und Sprachen

Akzeptierte Wörter

Ein Wort $w \in X^*$ wird vom endlichen Akzeptor akzeptiert, wenn man ausgehend vom Anfangszustand bei Eingabe von w in einem akzeptierenden Zustand endet.

Bemerkung

- Wird ein Wort nicht akzeptiert, dann wurde es abgelehnt

Akzeptierte formale Sprache

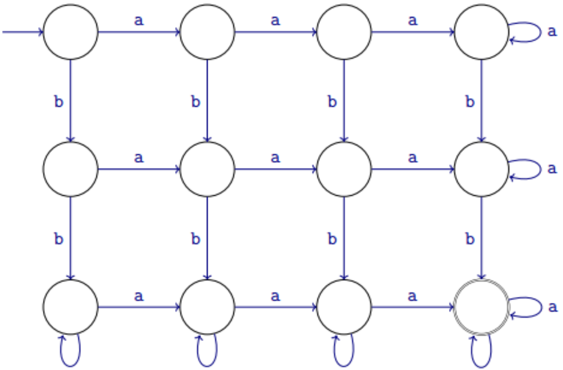
Die von einem Akzeptor A akzeptierte formale Sprache $L(A)$ ist die Menge aller von ihm akzeptierten Wörter.

Endliche Akzeptoren

Aufgabe zu endlichen Akzeptoren

Konstruiere einen endlichen Akzeptor, der die Sprache
$L_1(A) = \{w \in \{a,b\}^* : (N_a(w) \geq 3 \wedge N_b(w) \geq (2)\}$ erkennt.

Lösung

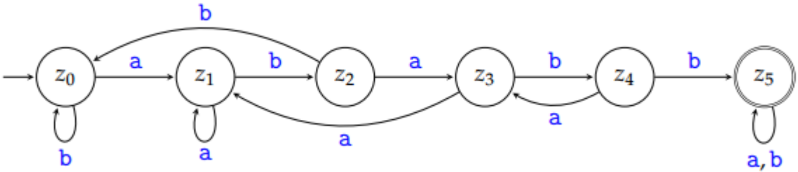


Endliche Akzeptoren

Aufgabe zu endlichen Akzeptoren

Konstruiere einen endlichen Akzeptor, der die Sprache
$L_2(A) = \{w_1 ababbw_2 | w_1, w_2 \in \{a, b\}^*\}$ erkennt.

Lösung



Aufgabe

Konstuiere einen endlichen Akzeptor der die Sprache
$L_3 = \{w \in \{a, b\}^* | w \notin L_2\}$ akzeptiert.

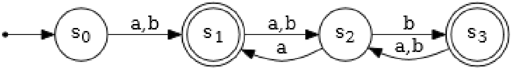
Lösung

Ablehnende Zustände wereden zu akzeptierenden und andersrum.

Endliche Akzeptoren

Aufgaben zu endlichen Akzeptoren

- Gebe für den unten stehenden Automaten an, welche Sprache dieser akzeptiert.
- Gebe für die folgende Sprache über dem Alphabet $\{a, b\}$ einen endlichen Akzeptor an: $L = \{w \in \Sigma^* | N_a(w) \bmod 3 > N_b(w) \bmod 2\}$

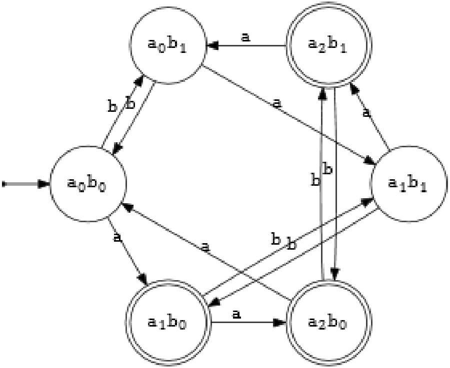


Lösungen

Lösung 1

$L = \{w \in \Sigma^* | |w| \bmod 2 = 1\}$ (Worte ungerader Länger)

Lösung 2



Endliche Akzeptoren

Wann wird das leere Wort ε von einem endlichen Akzeptor akzeptiert?
$\varepsilon \in L(A)$ gilt genau dann, wenn der Startzustand akzeptiert wird.

Gibt es einen endlichen Akzeptor, der die Sprache $L_1 = \{w \in \{1\}^* | \text{w ist Teilwort der Binärdarstellung von } \pi\}$ erkennt

Gibt es einen endlichen Akzeptor, der die Sprache $L_2 = \{0^k 1^k | k \in \mathbb{N}_0\}$ erkennt

Turingmaschinen

Was sind Turingmaschinen?

- Sehr mächtige Erweiterung Automat
  - Was heißt mächtig?
  - Turingmaschinen können eine große Vielfalt von Problemen lösen, einschließlich vieler in GBI besprochener Probleme
- Gesteuert durch einen endlichen Automaten, aber mit einem unendlichen Arbeitsband zum Zwischenspeichern von Informationen
- Besitzen einen Kopf um auf dem Band zu lesen und zu schreiben
- Turingmaschinen sind sozusagen genauso mächtig wie Computer
  - können also benutzt werden, um für Probleme zu entscheiden, ob sie gelöst werden können oder nicht

Definition von Turingmaschinen

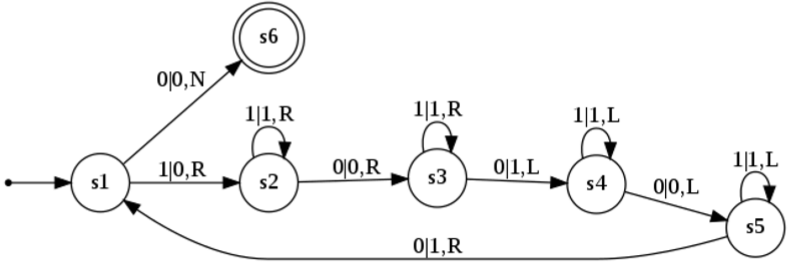
Definition von Turingmaschinen

Eine Turingmaschine $T = (Z, z_0, X, f, g, m)$ besteht aus:

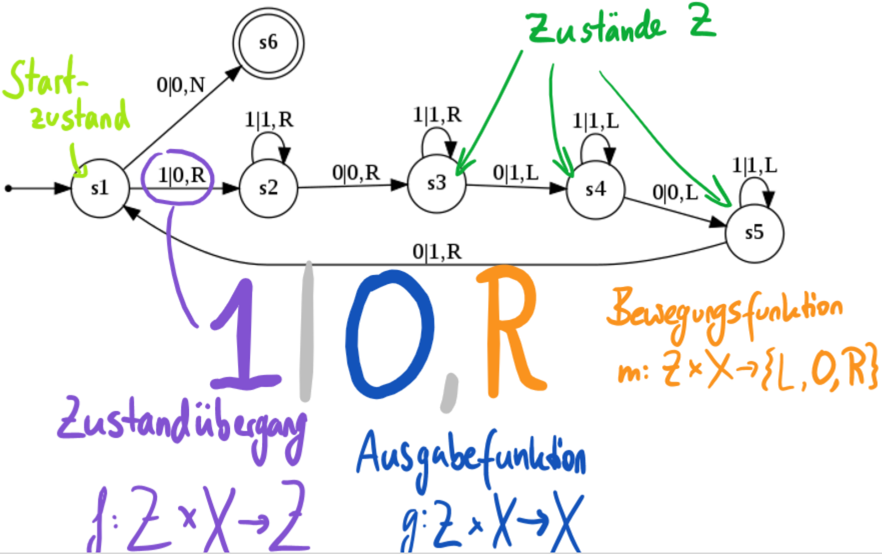
- Z Zustandsmenge
- $z_0 \in Z$ Startzustand
- X Bandalphabet
- $\square$ Blanksymbol (sozusagen Markierung für Leerzeichen)
- $f : Z \times X \to Z$ partielle Zustandsübergangsfunktion
- $g : Z \times X \to X$ partielle Ausgabefunktion
- $m : Z \times X \to \{L, N, R\}$ partielle Bewegungsfunktion

Anmerkung: partielle Funktionen sind nicht linkstotal, also manche Elemente des Definitionsbereichs werden nicht abgebildet.

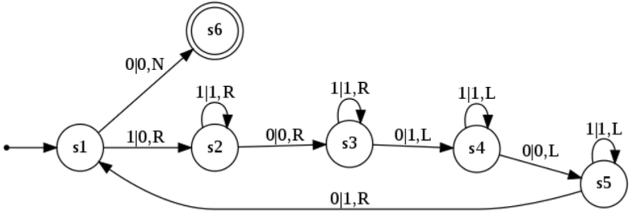
Beispiel einer Turingmaschine



Beispiel einer Turingmaschine



Funktionen von Turingmaschinen



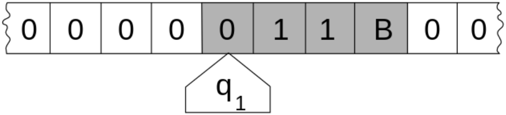
Wie sehen die konkreten Abbildungsvorschriften der linken vier Pfeile aus?

- $f : Z \times X \to Z$
- $g : Z \times X \to X$
- $m : Z \times X \to \{L, N, R\}$

	f	g	m
$0\|0, N$	$(s1, 0) \mapsto s6$	$(s1, 0) \mapsto 0$	$(s1, 0) \mapsto N$
$1\|0, R$	$(s1, 1) \mapsto s2$	$(s1, 1) \mapsto 0$	$(s1, 1) \mapsto R$
$1\|1, R$	$(s2, 1) \mapsto s2$	$(s2, 1) \mapsto 1$	$(s2, 1) \mapsto R$
$0\|0, R$	$(s2, 1) \mapsto s3$	$(s2, 1) \mapsto 0$	$(s2, 1) \mapsto R$

Das Band einer Turingmaschine

- Unendliche Anreihung von Zeichen, die nach links und rechts unbegrenzt weiter geht



- Die Turingmaschine hat einen Kopf, mit dem sie das aktuelle Zeichen lesen oder überschreiben kann, oder kann ihn nach links oder rechts bewegen.
- Das Band einer Turingmaschine wird benutzt als...
  - Erhalten der Eingabe: Bevor die Turingmaschine startet, steht das Eingabewort auf dem Band, der Kopf steht auf dem ersten Zeichen der Eingabe.
  - Rückgabe der Ausgabe: Nach Beenden steht auf dem Band die Ausgabe (und der Kopf irgendwo).
  - Zwischenspeicher: Die Turingmaschine kann überall Informationen zwischenspeichern, diese müssen von der TM am Ende aber gelöscht

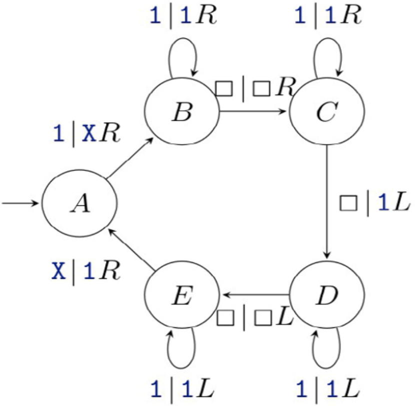
Beispielabarbeitungen

Gemeinsame Übung

Arbeite folgende Wörter mit der Turingmaschine ab:

- 0
- 1
- 11
- 111

Was macht die Turingmaschine?




That's all Folks!