

Grundbegriffe der Informatik

Tutorium 33

Lukas Bach, lukas.bach@student.kit.edu | 02.02.2017



Mealy-Automat

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

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Mealy-Automat

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

- endliche Zustandsmenge Z

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Mealy-Automat

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

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Mealy-Automat

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

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Mealy-Automat

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

- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \rightarrow Z$

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Darstellung als Graph

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Darstellung als Graph

- Zustände \rightarrow Knoten

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Darstellung als Graph

- Zustände \rightarrow Knoten
- Startzustand \rightarrow Pfeil an diesen Knoten (ohne Anfang)

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Darstellung als Graph

- Zustände \rightarrow Knoten
- Startzustand \rightarrow Pfeil an diesen Knoten (ohne Anfang)
- Zustandsüberföhrungsfunktion \rightarrow Kanten mit Beschriftung

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Mealy-Automat

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

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

Darstellung als Graph

- Zustände \rightarrow Knoten
- Startzustand \rightarrow Pfeil an diesen Knoten (ohne Anfang)
- Zustandsüberföhrungsfunktion \rightarrow Kanten mit Beschriftung
- Ausgabefunktion \rightarrow zusätzliche Kantenbeschriftung

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

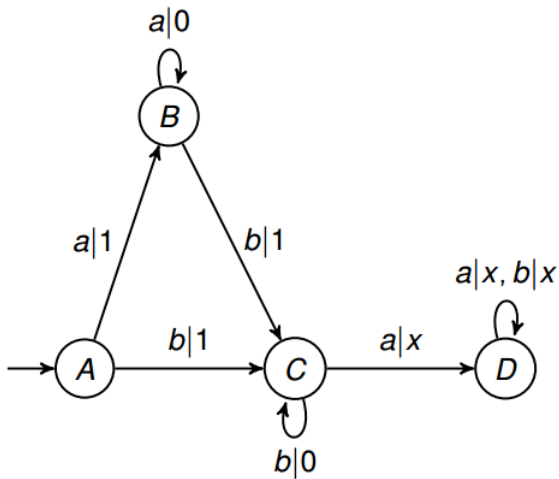
Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken



Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Moore-Automat

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

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

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Moore-Automat

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

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

→ Bis hierhin alles wie bei Mealy!

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare
Grammatiken

Moore-Automat

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

- endliche Zustandsmenge Z
 - Anfangszustand $z_0 \in Z$
 - Eingabealphabet X
 - Zustandsübergangsfunktion $f : Z \times X \rightarrow Z$
 - Ausgabealphabet Y
- Bis hierhin alles wie bei Mealy!
- Ausgabefunktion $h : Z \rightarrow Y^*$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Moore-Automat

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

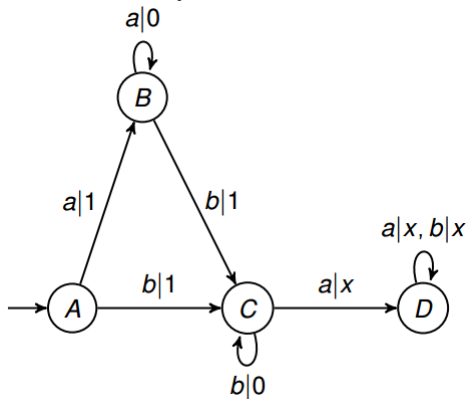
- endliche Zustandsmenge Z
 - Anfangszustand $z_0 \in Z$
 - Eingabealphabet X
 - Zustandsübergangsfunktion $f : Z \times X \rightarrow Z$
 - Ausgabealphabet Y
- **Bis hierhin alles wie bei Mealy!**
- Ausgabefunktion $h : Z \rightarrow Y^*$

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



Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

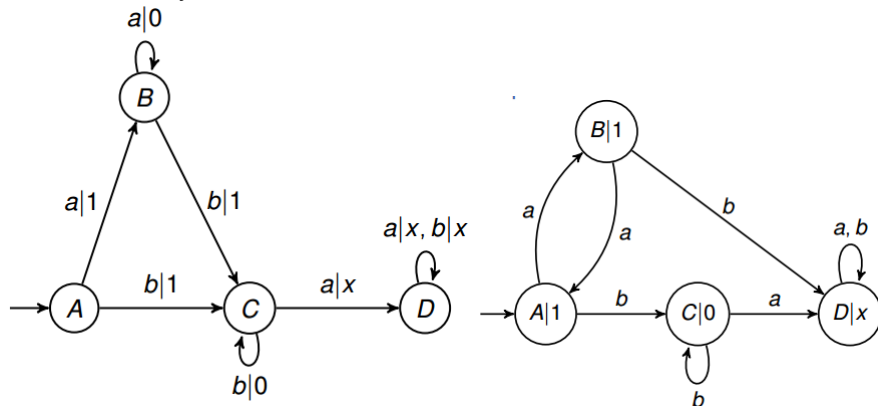
Ausdrücke

Rechtslineare

Grammatiken

Umwandlung Mealy- in Moore-Automat

Links ein Mealy-, rechts ein Moore-Automat



Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

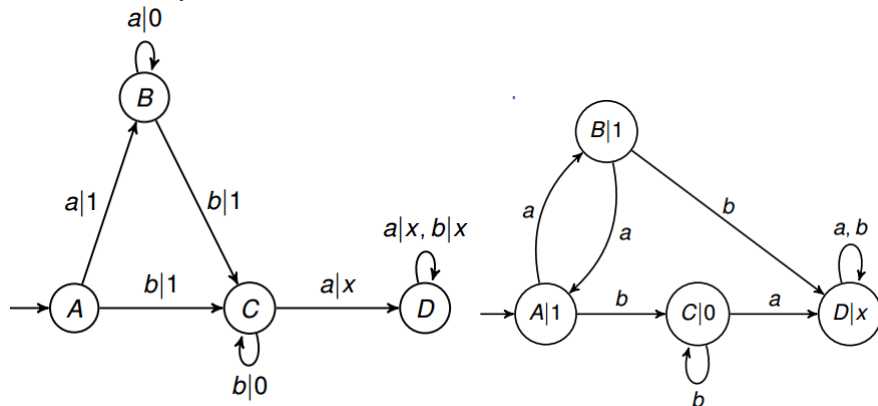
Ausdrücke

Rechtslineare

Grammatiken

Umwandlung Mealy- in Moore-Automat

Links ein Mealy-, rechts ein Moore-Automat



Aufgabe

Wie sieht der Mealy-Automat als äquivalenter Moore-Automat aus, wie sieht der Moore-Automat als äquivalenter Mealy-Automat aus?

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

■ Sonderfall von Moore-Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- Sonderfall von Moore-Automaten
- Bei einem Akzeptor will man nur wissen, ob die Eingabe akzeptiert wurde oder nicht (also reicht ein Bit als Ausgabealphabet)

Automaten

Mealy-Automat

Moore-Automat

Endliche Akzeptoren

Reguläre Ausdrücke

Rechtslineare Grammatiken

- 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

Automaten

Mealy-Automat

Moore-Automat

Endliche Akzeptoren

Reguläre Ausdrücke

Rechtslineare Grammatiken

- 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

Automaten


Mealy-Automat

Moore-Automat

Endliche Akzeptoren

Reguläre Ausdrücke

Rechtslineare Grammatiken

- 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 

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Akzeptierte Wörter

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Akzeptierte Wörter

Ein Wort $w \in X^*$ wird vom endlichen Akzeptor akzeptiert

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

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.

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

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.

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.

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

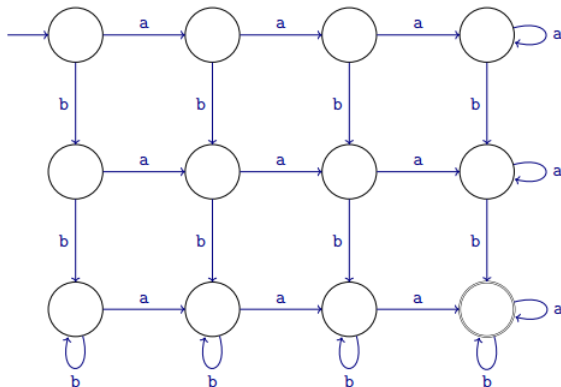
Reguläre
Ausdrücke

Rechtslineare
Grammatiken

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



Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu endlichen Akzeptoren

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

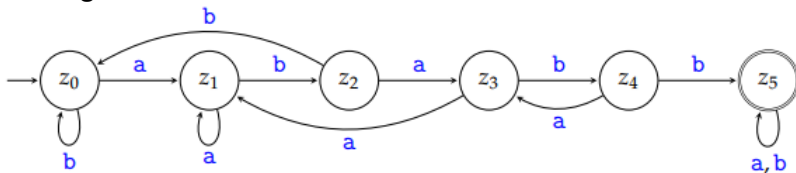
Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu endlichen Akzeptoren

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

Lösung



Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

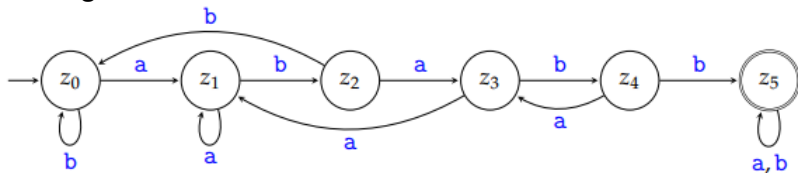
Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu endlichen Akzeptoren

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

Lösung



Aufgabe

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

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

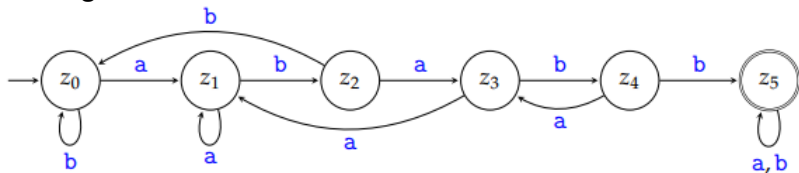
Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu endlichen Akzeptoren

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

Lösung



Aufgabe

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

Lösung

Ablehnende Zustände werden zu akzeptierenden und andersrum.

Automaten

Mealy-Automat

Moore-Automat

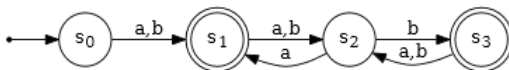
Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

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: $L = \{w \in \Sigma^* \mid N_a(w) \bmod 3 > N_b(w) \bmod 2\}$



Lösung 1

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

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

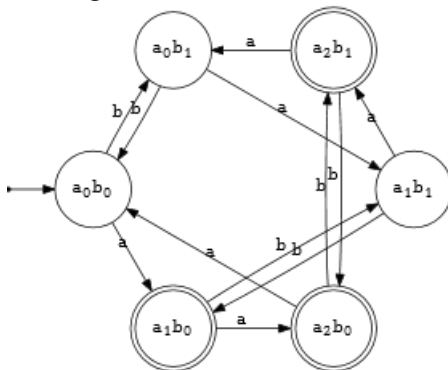
Rechtslineare

Grammatiken

Lösung 1

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

Lösung 2



Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Wann wird das leere Wort ε von einem endlichen Akzeptor akzeptiert?

Automaten

Mealy-Automat

Moore-Automat

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.

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

**Reguläre
Ausdrücke**

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{[, (,), *, \emptyset\}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{[, (,), *, \emptyset\}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
 - \emptyset ist ein RA

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
 - \emptyset ist ein RA
 - Für jedes $x \in A$ ist x ein RA

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
 - \emptyset ist ein RA
 - Für jedes $x \in A$ ist x ein RA
 - Wenn R_1 und R_2 RA sind, dann auch $(R_1 | R_2)$ und $(R_1 R_2)$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre Ausdrücke

Rechtslineare
Grammatiken

Regulärer Ausdruck

- Alphabet $Z = \{ |, (,), *, \emptyset \}$ von "Hilfssymbolen"
- Alphabet A enthalten keine Zeichen aus Z
- Ein **regulärer Ausdruck** (RA) über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.
- Vorschriften
 - \emptyset ist ein RA
 - Für jedes $x \in A$ ist x ein RA
 - Wenn R_1 und R_2 RA sind, dann auch $(R_1 | R_2)$ und $(R_1 R_2)$
 - Wenn R ein RA ist, dann auch (R^*)

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

■ “Stern- vor Punktrechnung”

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

- “Stern- vor Punktrechnung”

Mealy-Automat

- “Punkt- vor Strichrechnung”

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1 | R_2 R_3^*$ Kurzform für $(R_1 | (R_2 (R_3^*)))$

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1 | R_2 R_3^*$ Kurzform für $(R_1 | (R_2 (R_3^*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1 | R_2 R_3^*$ Kurzform für $(R_1 | (R_2 (R_3^*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

→ $R_1 | R_2 | R_3$ Kurzform für $((R_1 | R_2) | R_3)$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

→ $R_1 | R_2 | R_3$ Kurzform für $((R_1 | R_2) | R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(∅*))$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

→ $R_1 | R_2 | R_3$ Kurzform für $((R_1 | R_2) | R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(∅*)) → (abb) * *|∅*$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

→ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*))*|(\emptyset*)) \rightarrow (abb) * *|\emptyset*$

- $((a(a|b))|b)$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

- “Stern- vor Punktrechnung”

- “Punkt- vor Strichrechnung”

→ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

→ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

Entferne so viele Klammern wie möglich, ohne die Bedeutung des RA zu verändern.

- $(((((ab)b)*)*)|(\emptyset*)) \rightarrow (abb) * *|\emptyset*$

- $((a(a|b))|b) \rightarrow a(a|b)|b$

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$$G = (\{R\}, \{[, (,), *, \emptyset\} \cup A, R, P)$$

$$\text{mit } P = \{R \rightarrow \emptyset, R \rightarrow$$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$G = (\{R\}, \{[, (,), *, \emptyset\} \cup A, R, P)$
mit $P = \{R \rightarrow \emptyset, R \rightarrow x \text{ (mit } x \in A),$
 $R \rightarrow (R|R), R \rightarrow (RR),$
 $R \rightarrow (R*)$
 $R \rightarrow \varepsilon\}$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Wir können die Syntax von regulären Ausdrücken auch über eine kontextfreie Grammatik definieren.

Aufgabe

Vervollständigt die folgende Grammatik.

$$G = (\{R\}, \{[, (,), *, \emptyset\} \cup A, R, P)$$

mit $P = \{R \rightarrow \emptyset, R \rightarrow x \text{ (mit } x \in A),$
 $R \rightarrow (R|R), R \rightarrow (RR),$
 $R \rightarrow (R*)$
 $R \rightarrow \varepsilon\}$

Wieso brauchen wir ε ?

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Lukas Bach, lu-
kas.bach@student.kit.edu

Notation

- Spitze Klammern \langle, \rangle

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle =$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle =$

Automaten

Notation

Mealy-Automat

- Spitze Klammern \langle, \rangle

Moore-Automat

Regeln

Endliche

- $\langle \emptyset \rangle = \{ \}$

Akzeptoren

- $\langle x \rangle = \{ x \}$ für jedes $x \in A$

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle =$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_1 \rangle =$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R^* \rangle =$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Notation

- Spitze Klammern \langle, \rangle

Regeln

- $\langle \emptyset \rangle = \{ \}$
- $\langle x \rangle = \{ x \}$ für jedes $x \in A$
- $\langle R_1 | R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R^* \rangle = \langle R \rangle^*$

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Satz

Für jede formale Sprache L sind äquivalent:

1. L kann von einem endlichen Akzeptor erkannt werden.
2. L kann durch einen regulären Ausdruck beschrieben werden
3. L kann von einer rechtslinearen Grammatik erzeugt werden.

Solche Sprachen heißen regulär.

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken

Zum selbst probieren:

<http://regexr.com/>

Achtung: Reguläre Ausdrücke in praktischer Programmierung funktionieren zwar ähnlich, haben aber eine andere Syntax und können teils mehr!

Definition

Eine rechtslineare Grammatik ist eine reguläre Grammatik $G = (N, T, S, P)$ mit der Einschränkung, dass alle Produktionen die folgende Form haben:

- $X \rightarrow w$ mit $w \in T^*$ oder
- $x \rightarrow wY$ mit $w \in T^*$, $Y \in N$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0, 1\}^* \mid \exists k \in \mathbb{N}_0 : \text{Num}_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0, 1\}^* \mid \exists k \in \mathbb{N}_0 : \text{Num}_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

■ $R = (0 * 10) | (0 * 1(0) * 1) =$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare

Grammatiken

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0, 1\}^* \mid \exists k \in \mathbb{N}_0 : \text{Num}_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

■ $R = (0 * 10) | (0 * 1(0) * 1) = 0 * 10 | 0 * 10 * 1$

Automaten

Mealy-Automat

Moore-Automat

Endliche

Akzeptoren

Reguläre

Ausdrücke

Rechtslineare
Grammatiken

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0, 1\}^* \mid \exists k \in \mathbb{N}_0 : \text{Num}_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

- $R = (0 * 10) | (0 * 1(0) * 1) = 0 * 10 | 0 * 10 * 1$
- $G = (\{S, A\}, \{0, 1\}, S, \{S \rightarrow 0S | 10 | 1A, A \rightarrow 0A | 1\})$

Grundbegriffe der Informatik

Lukas Bach, lu-
kas.bach@student.kit.edu

Automaten

Mealy-Automat

Moore-Automat

Endliche
Akzeptoren

Reguläre
Ausdrücke

Rechtslineare
Grammatiken



That's all Folks!