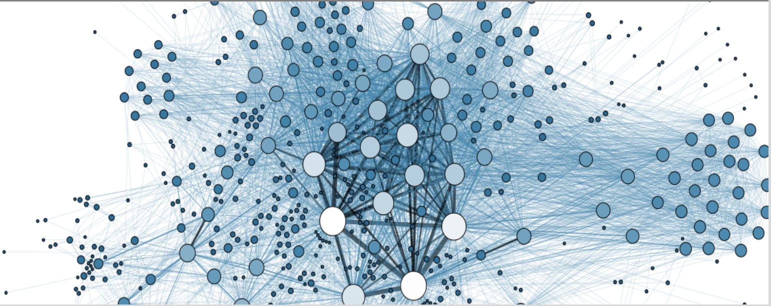
Grundbegriffe der Informatik
Tutorium 33

Maximilian Staab, maximilian.staab@fsmi.uni-karlsruhe.de
Lukas Bach, lukas.bach@student.kit.edu | 02.02.2017



Mealy-Automat

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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

Mealy-Automat

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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

- endliche Zustandsmenge Z

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

Mealy-Automat

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

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

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

Karlsruher Institut für Technologie

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

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

Karlsruher Institut für Technologie

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 $f : Z \times X \to Z$

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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 $f : Z \times X \to Z$
- Ausgabealphabet Y

Maximilian Staab,
maximilian.staab@fsmi.un
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

Karlsruher Institut für Technologie

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 $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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 $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

Darstellung als Graph

Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

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 $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

Darstellung als Graph

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

Mealy-Automat

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

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 $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

Darstellung als Graph

- Zustände $\to$ Knoten
- Startzustand $\to$ Pfeil an diesen Knoten (ohne Anfang)
- Zustandsüberführungsfunktion $\to$ Kanten mit Beschriftung

Maximilian Staab,
maximilian.staab@fsmi.un
Lukas Bach,
lukas.bach@student.kit.e

Mealy-Automat

Mealy-Automat

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

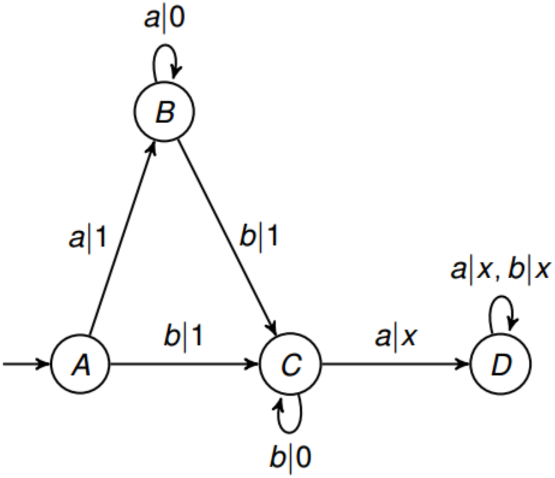
- endliche Zustandsmenge Z
- Anfangszustand $z_0 \in Z$
- Eingabealphabet X
- Zustandsübergangsfunktion $f : Z \times X \to Z$
- Ausgabealphabet Y
- Ausgabefunktion $h : Z \times X \to Y^*$

Darstellung als Graph

- Zustände $\to$ Knoten
- Startzustand $\to$ Pfeil an diesen Knoten (ohne Anfang)
- Zustandsüberführungsfunktion $\to$ Kanten mit Beschriftung
- Ausgabefunktion $\to$ zusätzliche Kantenbeschriftung

Beispiel Mealy-Automat

Maximilian Staab,
maximilian.staab@fsmi.uni
Lukas Bach,
lukas.bach@student.kit.ed



Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

Moore-Automat

Karlsruher Institut für Technologie

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 $f : Z \times X \to Z$
- Ausgabealphabet Y

Moore-Automat

Karlsruher Institut für Technologie

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

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 $f : Z \times X \to Z$
- Ausgabealphabet Y

$\rightarrow$ Bis hierhin alles wie bei Mealy!

Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.un
Lukas Bach,
lukas.bach@student.kit.e

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

$\rightarrow$ Bis hierhin alles wie bei Mealy!

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

Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

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 $f : Z \times X \to Z$
- Ausgabealphabet Y

$\to$ Bis hierhin alles wie bei Mealy!

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

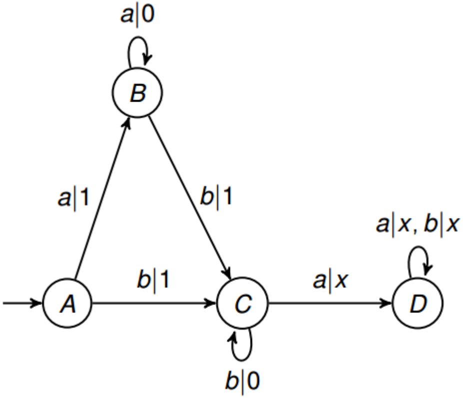
Bemerkung

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

Umwandlung Mealy- in Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.uni
Lukas Bach,
lukas.bach@student.kit.ed

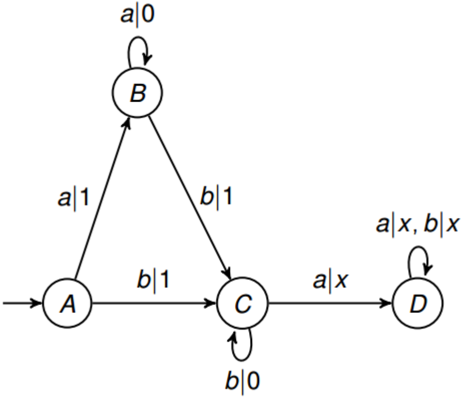
Links ein Mealy-, rechts ein Moore-Automat

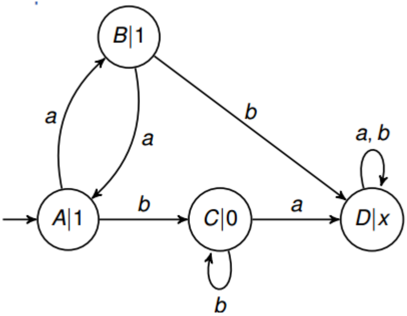


Umwandlung Mealy- in Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.uni
Lukas Bach,
lukas.bach@student.kit.ed

Links ein Mealy-, rechts ein Moore-Automat

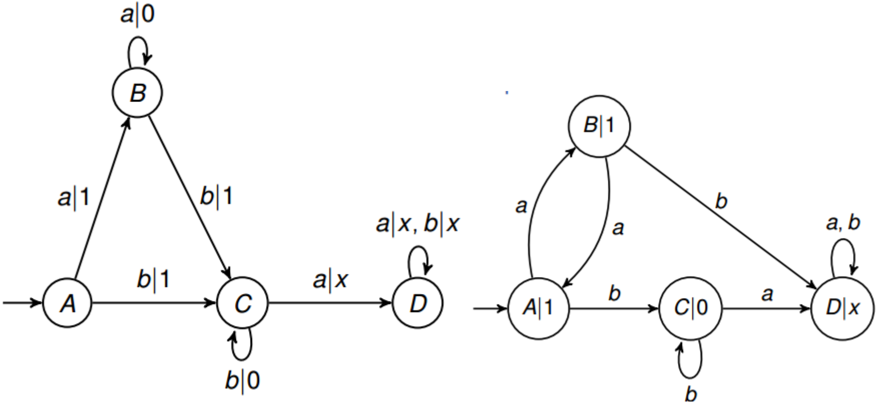




Umwandlung Mealy- in Moore-Automat

Maximilian Staab,
maximilian.staab@fsmi.uni
Lukas Bach,
lukas.bach@student.kit.ed

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?

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- Sonderfall von Moore-Automaten

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- 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

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- 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

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- 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

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Akzeptierte Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Akzeptierte Wörter

Akzeptierte Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Akzeptierte Wörter

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

Akzeptierte Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

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.

Akzeptierte Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

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 Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

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

Akzeptierte Wörter und Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

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

Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

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.

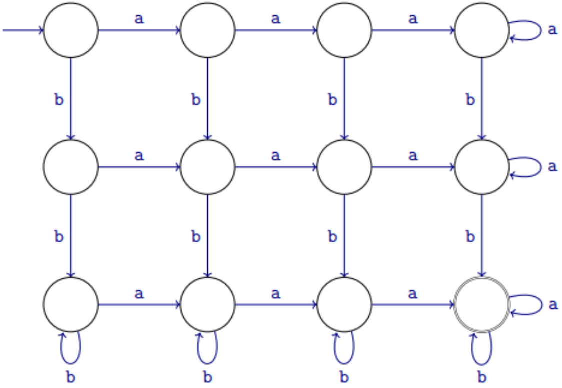
Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

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

Maximilian Staab,
maximilian.staab@fsmi.un
Lukas Bach,
lukas.bach@student.kit.e

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.

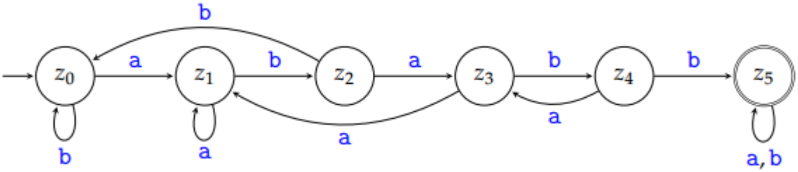
Maximilian Staab,
maximilian.staab@fsmi.ur
Lukas Bach,
lukas.bach@student.kit.e

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



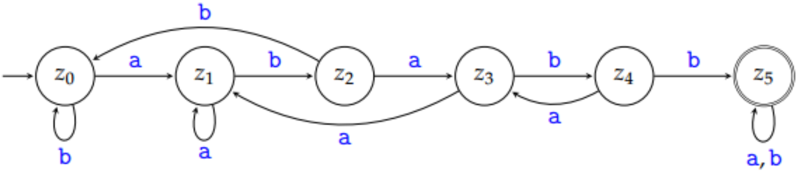
Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

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.

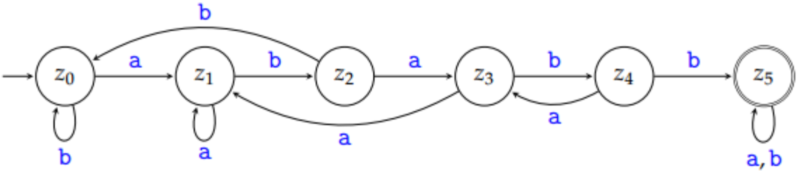
Maximilian Staab,
maximilian.staab@fsmi.u
Lukas Bach,
lukas.bach@student.kit.e

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

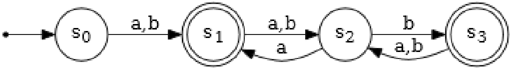
Ablehnende Zustände wereden zu akzeptierenden und andersrum.

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Lösung 1

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

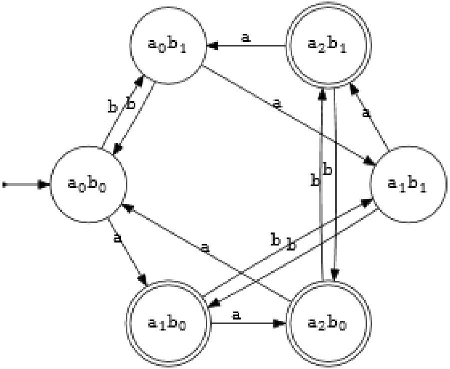
Lösungen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Lösung 1

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

Lösung 2



Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Endliche Akzeptoren

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

$\varepsilon \in L(A)$ gilt genau dann, wenn der Startzustand akzeptiert wird.

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Regulärer Ausdruck

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Regulärer Ausdruck

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

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Regulärer Ausdruck

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

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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.

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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_1R_2)

Regulärer Ausdruck

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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 $(R*)$

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\longrightarrow$ $R_1 | R_2 R_3 *$ Kurzform für $(R_1 | (R_2 (R_3 *)))$

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\rightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

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

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

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- "Stern- vor Punktrechnung"
- "Punkt- vor Strichrechnung"

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

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

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

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

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

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$
- $((a(a|b))|b)$

Klammerregeln

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

- “Stern- vor Punktrechnung”
- “Punkt- vor Strichrechnung”

$\longrightarrow$ $R_1|R_2R_3*$ Kurzform für $(R_1|(R_2(R_3*)))$

- Bei mehreren gleichen Operatoren ohne Klammern links geklammert

$\longrightarrow$ $R_1|R_2|R_3$ Kurzform für $((R_1|R_2)|R_3)$

Aufgabe

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

- $(((((ab)b)*)*)|(\emptyset*)) \longrightarrow (abb)**|\emptyset*$
- $((a(a|b))|b) \longrightarrow a(a|b)|b$

Alternative Definition

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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 $P = \{R \rightarrow \emptyset, R \rightarrow$

Alternative Definition

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Alternative Definition

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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

Wieso brauchen wir ε?

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

- $\langle \emptyset \rangle =$

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle, \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Durch R beschriebene Sprache

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle , \rangle$

Regeln

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

Durch R beschriebene Sprache

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Notation

- Spitze Klammern $\langle, \rangle$

Regeln

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

Charakterisierung regulärer Sprachen

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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 heißten regulär.

Anwendung von regulären Ausdrücken

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

Zum selbst probieren:
http://regexr.com/

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

Rechtslineare Grammatiken

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.edu

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 $w \in T^*$, $Y \in N$

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0, 1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

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

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ jeweils einen regulären Ausdruck R und eine rechtslineare Grammatik G an, sodass $L = \langle R \rangle = L(G)$ gilt.

Lösung

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

Maximilian Staab,
maximilian.staab@fsmi.uni-karlsruhe.de,
Lukas Bach,
lukas.bach@student.kit.e

Aufgabe zu rechtslinearen Grammatiken

Gebe zu $L = \{w \in \{0,1\}^* | \exists k \in \mathbb{N}_0 : Num_2(w) = 2^k + 1\}$ 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

Maximilian Staab,
maximilian.staab@fsmi.uni
Lukas Bach,
lukas.bach@student.kit.ed