Formale Grundlagen der Informatik 1
Kapitel 5
Abschlusseigenschaften

Frank Heitmann
heitmann@informatik.uni-hamburg.de

18. April 2016

Der deterministische, endliche Automat

Definition (DFA)

Ein **deterministischer endlicher Automat** (DFA) ist ein 5-Tupel

$$A = (Z, \Sigma, \delta, z_0, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times \Sigma \to Z$.
- Dem *Startzustand* $z_0 \in Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Der nichtdeterministische, endliche Automat

Definition (NFA)

Ein **nichtdeterministischer endlicher Automat** (NFA) ist ein 5-Tupel

$$A = (Z, \Sigma, \delta, Z_{start}, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times \Sigma \to 2^Z$.
- Der Menge der *Startzustände* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Der nichtdeterministische, endliche Automat

Definition (NFA - Alternative)

Ein **nichtdeterministischer endlicher Automat** (NFA) ist ein 5-Tupel

$$A = (Z, \Sigma, K, Z_{start}, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der **Zustandsübergangsrelation** $K \subseteq Z \times \Sigma \times Z$
- Der Menge der *Startzustände* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Überführungsfunktion und Rechnung

Definition (Erweiterte Überführungsfunktion)

Die **erweiterte Überführungsfunktion** $\hat{\delta} : 2^Z \times \Sigma^* \to 2^Z$ wird für alle $Z' \subseteq Z$, $x \in \Sigma$ und $w \in \Sigma^*$ rekursiv definiert durch

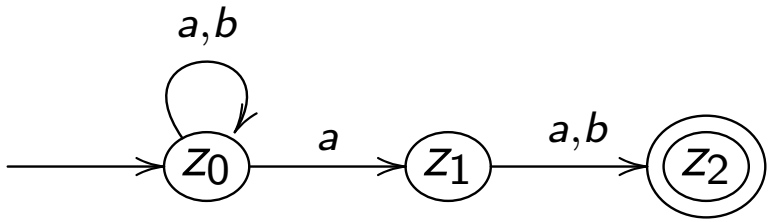
$$\begin{aligned} \hat{\delta}(Z', \lambda) &:= Z' \\ \hat{\delta}(Z', xw) &:= \cup_{z \in Z'} \hat{\delta}(\delta(z, x), w) \end{aligned}$$

oder alternativ durch

$$\begin{aligned} \hat{\delta}(Z', \lambda) &:= Z' \\ \hat{\delta}(Z', xw) &:= \hat{\delta}(\cup_{z \in Z'} \delta(z, x), w) \end{aligned}$$

Überführungsfunktion und Rechnung

$$\begin{aligned}\hat{\delta}(Z', \lambda) &:= Z' \\ \hat{\delta}(Z', xw) &:= \hat{\delta}(\cup_{z \in Z'} \delta(z, x), w)\end{aligned}$$



$$\hat{\delta}(\{z_0\}, aba) = \hat{\delta}(\{z_0, z_1\}, ba) = \hat{\delta}(\{z_0\} \cup \{z_2\}, a) =$$
$$\hat{\delta}(\{z_0, z_2\}, a) = \hat{\delta}(\{z_0, z_1\} \cup \emptyset, \lambda) = \hat{\delta}(\{z_0, z_1\}, \lambda) = \{z_0, z_1\}$$

Akzeptierte Sprache

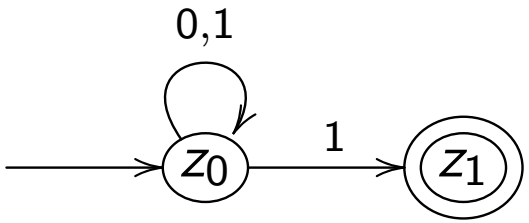
Definition (Akzeptierte Sprache)

Die von einem NFA A **akzeptierte Sprache** ist die Menge

$$\begin{aligned} L(A) &:= \{w \in \Sigma^* \mid \hat{\delta}(Z_{start}, w) \cap Z_{end} \neq \emptyset\} \\ &= \{w \in \Sigma^* \mid (z_0, w) \vdash^* (z_e, \lambda), z_0 \in Z_{start}, z_e \in Z_{end}\} \end{aligned}$$

Für die untere Notation definiert man sich ähnlich wie bei DFAs die Begriffe Konfiguration, Konfigurationsübergang, Rechnung und Erfolgsrechung (siehe vorherigen Foliensatz).

Der NFA - Ein Beispiel



Alle möglichen Rechnungen auf dem Wort 011:

- $(z_0, 011) \vdash (z_0, 11) \vdash (z_1, 1)$ – blockiert
- $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_0, \lambda)$ – lehnt ab
- $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_1, \lambda)$ – akzeptiert!

Da es (mindestens) eine akzeptierende Rechnung gibt, akzeptiert der Automat die Eingabe!

Äquivalenz?

DFAs vs. NFAs

NFAs sind mindestens so mächtig wie DFAs.

Die Idee

Jeder DFA kann auch als spezieller NFA gesehen werden, indem man

$$\delta_N(z, x) := \{\delta_D(z, x)\} \text{ und } Z_{start} := \{z_0\}$$

setzt (δ_N ist die Überführungsfunktion des NFA, δ_D die des DFA).

Die Rückrichtung geht dann mit der Potenzautomatenkonstruktion...

Der Potenzautomat

Satz

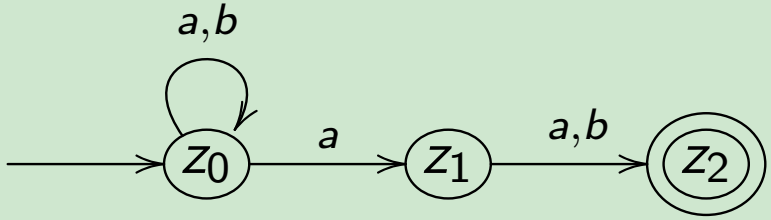
Zu jeder von einem NFA A akzeptierte Menge L kann ein DFA B konstruiert werden mit $L(B) = L$.

Die Konstruktion

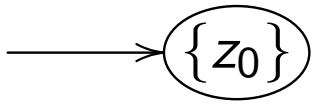
Sei $A = (Z, \Sigma, \delta, Z_{start}, Z_{end})$ der NFA. Wir definieren $B = (Z', \Sigma', \delta', z_0, Z'_{end})$ mit

1. $Z' := 2^Z$
2. $\Sigma' := \Sigma$
3. $\delta'(M, x) := \cup_{z \in M} \delta(z, x)$
4. $z_0 := Z_{start}$
5. $Z'_{end} := \{M \in 2^Z \mid M \cap Z_{end} \neq \emptyset\}$

Der Potenzautomat - Ein Beispiel

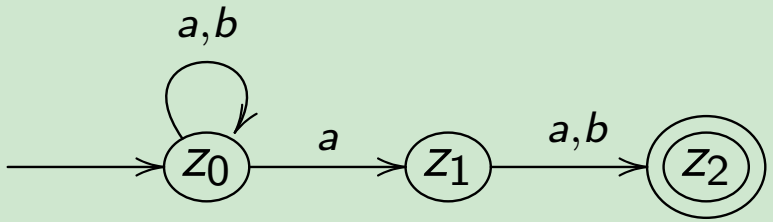


Wir bauen nur die initiale Zusammenhangskomponente und beginnen mit dem Startzustand:

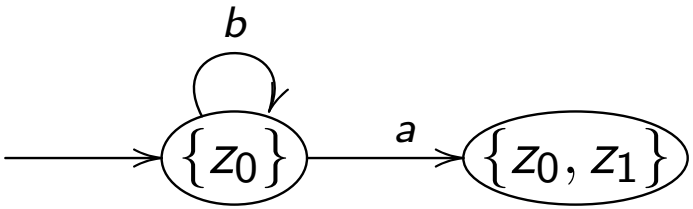


Wohin gelangen wir mit einem a? Wohin mit einem b?

Der Potenzautomat - Ein Beispiel

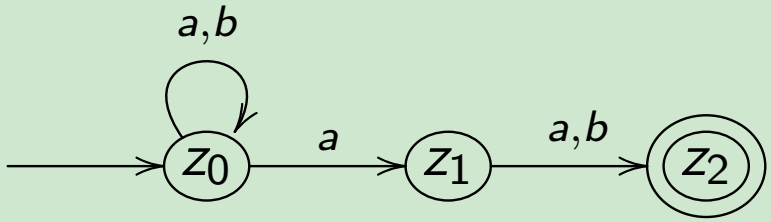


Mit a zu $\{z_0, z_1\}$, mit b zu $\{z_0\}$

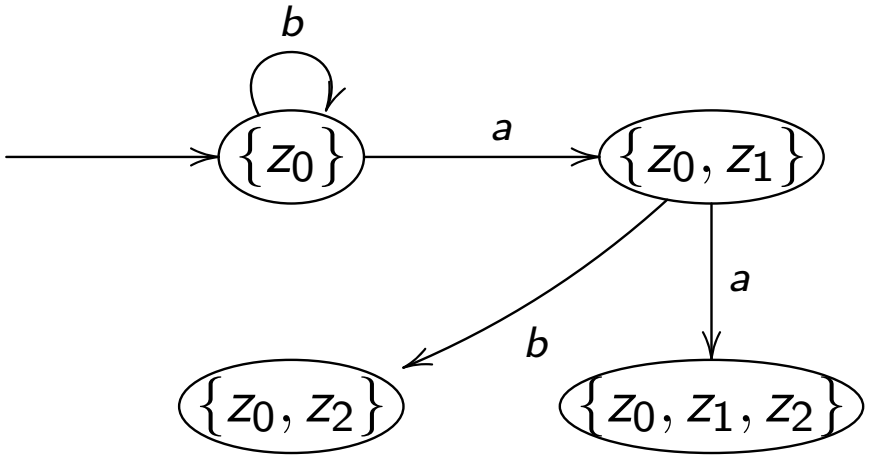


Wohin nun von $\{z_0, z_1\}$ mit a und b?

Der Potenzautomat - Ein Beispiel

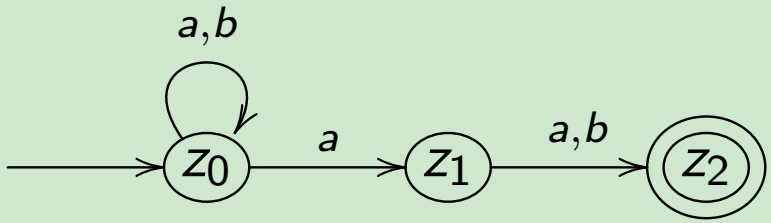


Es ist $\delta(z_0, a) = \{z_0, z_1\}$ und $\delta(z_1, a) = \{z_2\}$ also insgesamt $\delta'(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$ und ähnlich für b:

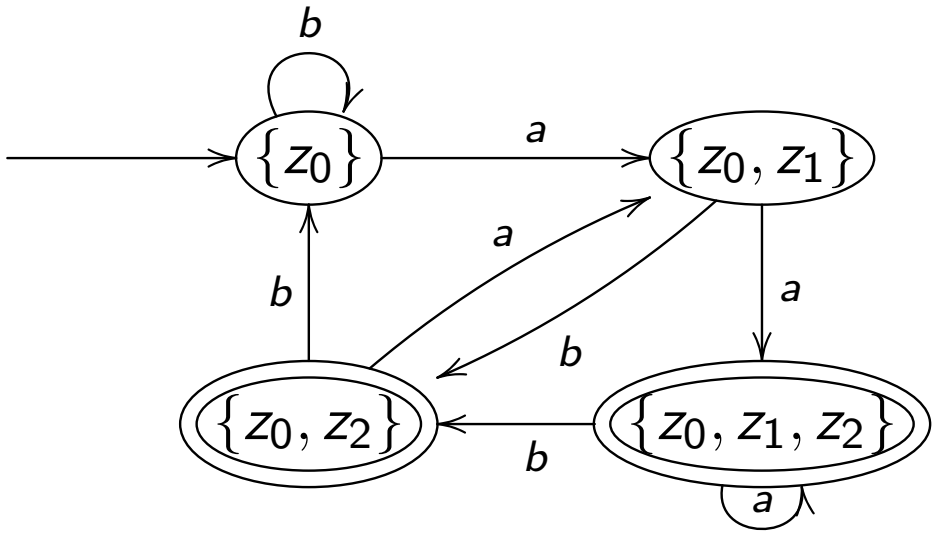


So fahren wir fort...

Der Potenzautomat - Ein Beispiel



... und fügen noch die Endzustände hinzu ...



Der Potenzautomat - Korrektheit der Konstruktion

Beweis.

Wir zeigen per Induktion über $|w|$, dass für jedes $w \in \Sigma^*$ und $M \subseteq Z$

$$\hat{\delta}'(M, w) = \hat{\delta}(M, w)$$

- Ind.-Anfang $|w| = 0$, d.h. $w = \lambda$. In diesem Fall ist $\hat{\delta}'(M, w) = \hat{\delta}(M, w) = M$ (wegen der Definition der erweiterten Überführungsfunktion).
- Ind.-Annahme: Gelte die Aussage für Wörter der Länge n.
- Ind.-Schritt: Sei $w \in \Sigma^*$ mit $|w| = n + 1$. Wir zerlegen w in xv (also $w = xv$) mit $x \in \Sigma$. x ist also das erste Symbol...

Der Potenzautomat - Korrektheit der Konstruktion

Beweis.

Fortsetzung des Induktionsschrittes: Nach den Definitionen der erweiterten Überführungsfunktionen folgt nun

- $\hat{\delta}(M, xv) = \hat{\delta}(\cup_{z \in M} \delta(z, x), v)$
- $\hat{\delta}'(M, xv) = \hat{\delta}'(\delta'(M, x), v) = \hat{\delta}'(\cup_{z \in M} \delta(z, x), v)$

Letztere Gleichheit aufgrund der Definition von δ'. Sei $\cup_{z \in M} \delta(z, x) = M'$. Nun gilt aber $|v| = n$ und damit ist die Induktionsannahme anwendbar, d.h. es ist $\hat{\delta}'(M', v) = \hat{\delta}(M', v)$ und damit insgesamt $\hat{\delta}'(M, w) = \hat{\delta}(M, w)$.

Der Potenzautomat - Korrektheit der Konstruktion

Beweis.

Abschluss des Beweises: Setzen wir für M die Menge Z_{start} bzw. z_0 ein (was das gleiche ist nach Konstruktion), so erhalten wir

$$\hat{\delta}'(z_0, w) = \hat{\delta}(Z_{start}, w)$$

Nach Konstruktion und Definition der Akzeptanzbedingung akzeptieren nun sowohl A als auch B genau dann, wenn in $\hat{\delta}'(z_0, w)$ bzw. $\hat{\delta}(Z_{start}, w)$ ein Endzustand des NFA auftritt. Da diese Mengen gleich sind, akzeptiert A folglich genau dann, wenn B akzeptiert und damit ist $L(A) = L(B)$. □

Zusammenfassung

DFAs und NFAs

Zu jeder regulären Sprache L gibt es

- einen DFA A mit $L(A) = L$
- einen NFA B mit $L(B) = L$

DFAs und NFAs sind also äquivalent.

Wichtige Anmerkung

Führt man das praktisch aus bzw. implementiert man dies, so wählt man die algorithmische Variante aus dem Beispiel, nicht die eher statische aus dem Beweis.

Fragen...

Sie A ein nichtdeterministischer endlicher Automat. Was ist richtig?

1. $L(A) \subseteq REG$
2. $L(A) \in REG$
3. $L(A) \subsetneq REG$
4. $L(A) = REG$

Fragen...

Wie ist die Endzustandsmenge des DFA in der Potenzautomatenkonstruktion definiert?

1. Z_{end} ist die Menge aller Endzustände des NFA
2. Z_{end} ist die Menge aller Mengen, die Endzustände des NFA enthalten.
3. Z_{end} ist die Menge aller Mengen, die geschnitten mit der Menge der Endzustände des NFA leer sind.
4. Z_{end} ist die Menge aller Mengen, die geschnitten mit der Menge der Endzustände des NFA nicht leer sind.

Fragen...

Welche Aussage stimmt für einen Potenzautomaten, der wie im Beweis angegeben konstruiert wurde?

1. Der Potenzautomat ist stets vollständig und initial zusammenhängend.
2. Der Potenzautomat ist stets vollständig, aber nicht zwingend initial zusammenhängend.
3. Der Potenzautomat ist nicht zwingend vollständig, aber stets initial zusammenhängend.
4. Der Potenzautomat ist weder zwingend vollständig noch zwingend initial zusammenhängend.

Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

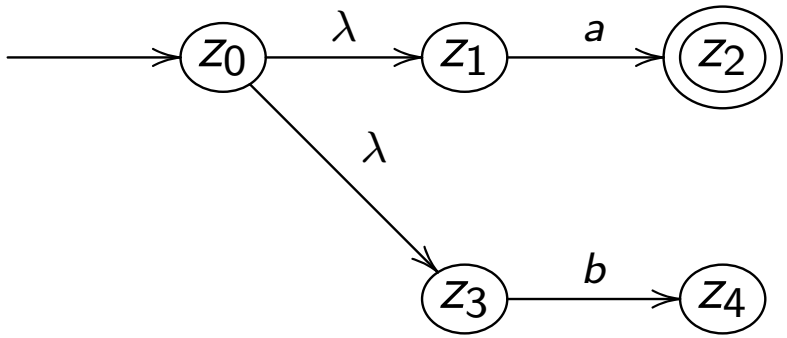
1. 2
2. 2 und 4
3. 2

Lambda-Kanten

Man kann einem NFA zusätzlich zu dem bisherigen noch erlauben,

- λ-Kanten (oder ϵ-Kanten)

zu benutzen. Nutzt man diese, liest man nichts vom Eingabeband (und der Lesekopf auf dem Eingabeband geht nicht weiter). Es findet nur ein Zustandswechsel statt.



Die Einfachheit des Beispiels täuscht. Die Kanten können sehr nützlich sein.

Lambda-Kanten

Definitionen

Begriffe wie Rechnung etc. wären nun anzupassen

Satz

Zu jedem NFA mit λ-Kanten kann ein äquivalenter NFA ohne λ-Kanten konstruiert werden.

Lambda-Kanten

Vorgehen

Wir wollen λ-Kanten in bestimmten Kontexten benutzen. Wir wissen dann, dass wir solch einen NFA auch in einen DFA umwandeln können, die Sprache also regulär ist. Wir werden aber weder formal die λ-Kanten einführen, noch den Beweis führen, dass auch NFA mit ihnen wieder genau die regulären Sprachen akzeptieren. Wir werden sie einfach informal benutzen.

Literaturhinweis

Interessierte finden die Konstruktion in [HMU] (siehe Literaturhinweis zu Anfang).

Reguläre Sprachen

Bisher kennen wir zur Beschreibung einer regulären Sprache damit

- DFAs
- NFAs
- NFAs mit λ-Kanten

Wir lernen nun noch eine vierte Möglichkeit kennen...

Zu regulären Ausdrücken

Wir wollen Sprachen mit Ausdrücken der Art

- a^+
- $(a + (b \cdot c))^*$
- usw.

beschreiben...

Man kennt das vielleicht aus Programmiersprachen...

Reguläre Ausdrücke

Definition (Reguläre Ausdrücke)

Sei Σ ein Alphabet. Die **regulären Ausdrücke über** Σ sind induktiv definiert durch:

1. $\emptyset$ ist ein regulärer Ausdruck, der die Menge $M_\emptyset = \emptyset$ beschreibt.
2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck, der die Menge $M_a = \{a\}$ beschreibt.
3. Sind X und Y reguläre Ausdrücke, die die Mengen M_X und M_Y beschreiben, dann beschreibt
   - $(X + Y)$ die Menge $M_X \cup M_Y$
   - $(X \cdot Y)$ die Menge $M_X \cdot M_Y$
   - X^* die Menge M_X^* und
   - X^+ die Menge M_X^+.
4. Nur die so erzeugten Ausdrücke sind reguläre Ausdrücke.

Beispiele

Beispiele ($\equiv$ für "beschreibt die Menge"):

- $(a + b)^* \equiv (\{a\} \cup \{b\})^* = \{a, b\}^*$
- $(a \cdot b) \equiv \{a\} \cdot \{b\} = \{ab\}$
- $a \cdot b^* + b \cdot a^* \equiv \{a\} \cdot \{b\}^* \cup \{b\} \cdot \{a\}^*$
- $\emptyset^* \equiv \emptyset^* = \{\lambda\}$

Anmerkung

- Abgesehen von $+$ für $\cup$ also recht ähnlich zu den Mengenoperationen, die wir schon kennen!
- Wir benutzen die üblichen Klammerersparnis regeln wie $\cdot$ vor $+$ und hoch $+$ bzw. hoch $*$ vor $\cdot$.

Äquivalenzbeweis

Man kann nun zeigen:

Satz

Sei A ein regulärer Ausdruck und M_A die von A beschriebene Menge. Dann gibt es einen DFA A' mit $L(A') = M_A$.

Satz

Sei A' ein DFA. Dann gibt es einen regulären Ausdruck A so dass $L(A') = M_A$ für die von A beschriebene Menge M_A.

Wichtige Anmerkung

Die beiden Modelle sind also wieder äquivalent. Auch mit regulären Ausdrücken lassen sich also genau die regulären Sprachen beschreiben.

Äquivalenzbeweis

Hinweis

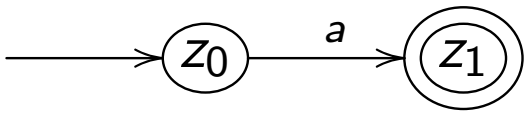
Wir skizzieren jetzt recht knapp beide Richtungen. Für später genügt es, wenn ihr wisst, dass die Äquivalenz besteht. Trotzdem hier jetzt einmal die Idee(n) und die Literaturhinweise für Interessierte...

Äquivalenzbeweis - Die Idee 1

Die Richtung "regulärer Ausdruck $\rightarrow$ DFA" ist relativ einfach.

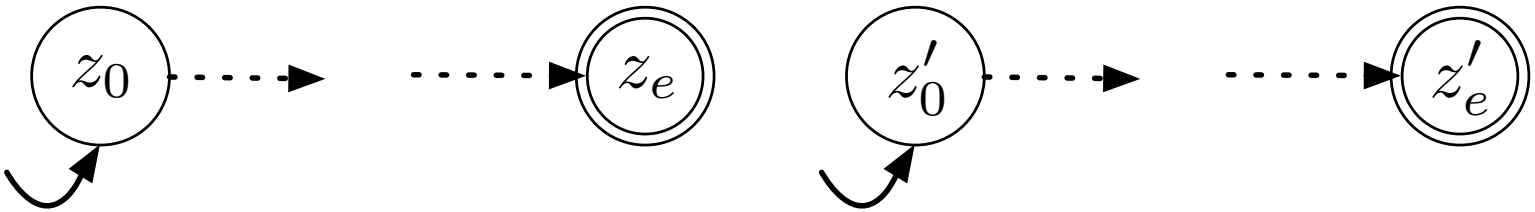
- Zu jeder Operation bei der Erstellung eines regulären Ausdrucks gibt man ein Verfahren an, wie man einen NFA (mit λ-Kanten) konstruieren kann.
- Dieser hat stets genau einen Start- und genau einen Endzustand.

Z.B. zum regulären Ausdruck a einfach



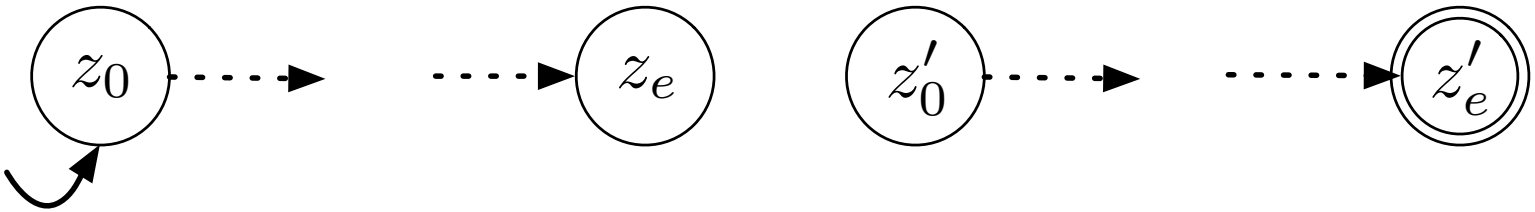
Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A \cdot B$



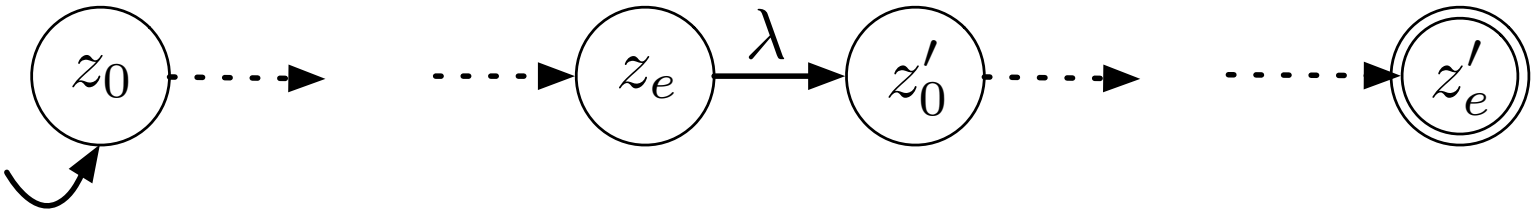
Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A \cdot B$



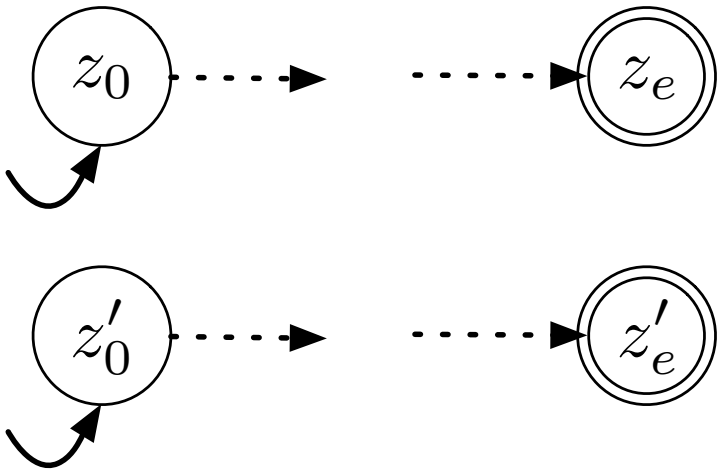
Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A \cdot B$



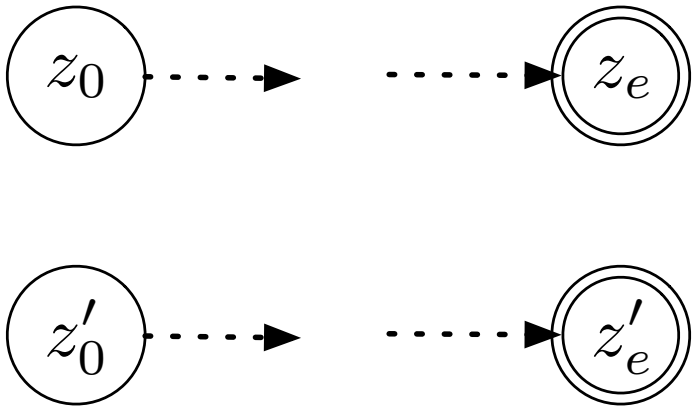
Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A + B$



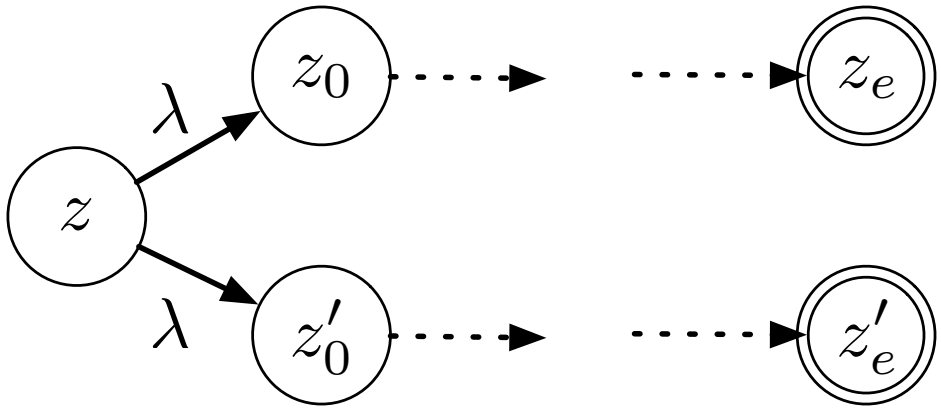
Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A + B$



Äquivalenzbeweis - Die Idee 1

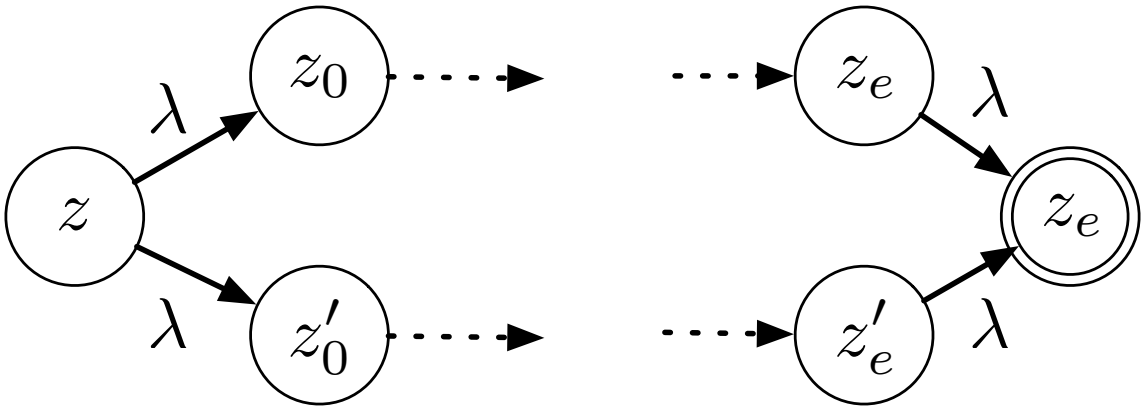
Z.B. zum regulären Ausdruck $A + B$



(z ist neuer Startzustand!)

Äquivalenzbeweis - Die Idee 1

Z.B. zum regulären Ausdruck $A + B$



(z ist neuer Startzustand!)

Äquivalenzbeweis - Die Idee 1

Anmerkung

- Dies kann man für alle Operatoren machen und kann dann so wie der reguläre Ausdruck Stück für Stück aufgebaut ist, auch den Automaten dafür Stück für Stück aufbauen.
- Im richtigen Verfahren wird noch auf Dinge geachtet, wie dass man nur einen Startzustand und Endzustand hat etc. (so wie oben).

Literaturhinweis

Interessierte finden die Konstruktion in [HMU].

Äquivalenzbeweis - Die Idee 2

Die andere Richtung (DFA $\rightarrow$ regulärer Ausdruck) ist schwieriger!

Die Idee

Mittels einer Rekursionsformel werden Worte beschrieben, die auf bestimmten Pfaden im Automaten liegen. Damit lassen sich letztendlich alle akzeptierten Worte beschreiben (also die akzeptierte Sprache) und (!) wir benötigen dafür nur endlich oft die einfachen Operationen $\cup$, $\cdot$ und $*$, was mit regulären Ausdrücken geht.

Äquivalenzbeweis - Die Idee 2

Kern des Beweises ist es die Zustände durchzunummerieren $z_1, z_2, \ldots, z_n$ (z_1 ist der Startzustand) und dann folgende Rekursionsformel zu benutzen

$$R_{i,j}^k = \begin{cases} \{x \mid \delta(z_i, x) = z_j \text{ oder } (x = \lambda) \wedge (i = j)\}, & \text{falls } k = 0 \\ R_{i,j}^{k-1} \cup R_{i,k}^{k-1} \cdot (R_{k,k}^{k-1})^* \cdot R_{k,j}^{k-1}, & \text{falls } k \geq 1 \end{cases}$$

In $R_{i,j}^k$ sind genau jene Worte enthalten, die gelesen werden können, wenn

- in z_i begonnen wird
- in z_j geendet wird und
- dabei nur Zustände aus $\{z_1, z_2, \ldots, z_k\}$ besucht werden.

Man berechnet dann gerade die $R_{1,x}^n$ mit $z_x \in Z_{end}$, vereinigt diese und ist fertig.

Literaturhinweis

Interessierte finden die Konstruktion im Detail inkl. Beweis in [HMU].

Zusammenfassung

Zusammenfassung

Zu jeder regulären Sprache L gibt es

- einen DFA A mit $L(A) = L$
- einen NFA B mit $L(B) = L$
- einen NFA mit λ-Kanten C mit $L(C) = L$
- einen regulären Ausdruck D, der L beschreibt ($M_D = L$).

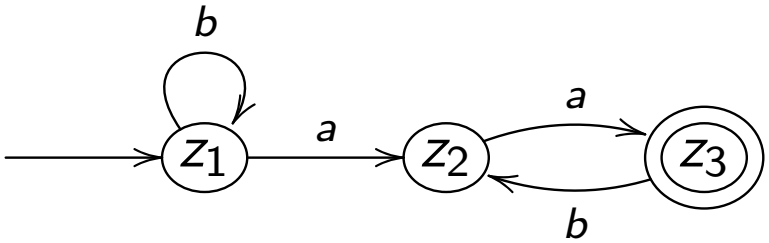
DFAs, NFAs, NFAs mit λ-Kanten und reguläre Ausdrücke sind also äquivalent.

Fragen...

Sie R ein regulärer Ausdruck. Was ist richtig?

1. $R \in REG$
2. $R \subseteq REG$
3. $R \subsetneq REG$
4. $R = REG$
5. nichts davon
6. mehrere davon

Fragen...



Welche Sprache akzeptiert obiger DFA?

1. b^*aa
2. $b^*aa(ba)^*$
3. $b^*(a+b)^*$
4. $b^+aa(ba)^+$

Fragen...

Sei $L \subseteq \Sigma^*$ eine Sprache. Ist dann $\Sigma^* \setminus L$ auch eine Sprache? Falls ja, ist sie eine reguläre Sprache?

1. Nein und Nein
2. Ja und Nein
3. Ja und Manchmal
4. Ja und Ja

Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

1. 5 (zunächst, wir wollen später 1 erlauben)
2. 2
3. 3 (wobei wir das manchmal noch genauer ausführen müssen)

Abschlusseigenschaften

Definition

Sei f_1 eine einstellige Operation auf Mengen und f_2 eine zweistellige Operationen. D.h. wenn M_1, M_2 zwei Mengen sind, dann sind auch $f_1(M_1)$ und $f_2(M_1, M_2)$ Mengen.

- Eine Sprachfamilie $\mathcal{C}$ ist **abgeschlossen** gegenüber der Operation f_1 bzw. f_2, wenn für jedes $R \in \mathcal{C}$ auch $f_1(R) \in \mathcal{C}$ gilt bzw. wenn für $R_1, R_2 \in \mathcal{C}$ auch $f_2(R_1, R_2) \in \mathcal{C}$ gilt.

Beispiel

Typische Beispiele sind z.B. Vereinigung oder Komplementbildung. Oft schreibt man die Operatoren in Infix-Notation (d.h. den Operator zwischen die Argumente).

Abschlusseigenschaften

Ziel

Wir wollen jetzt Abschlusseigenschaften für die regulären Sprachen untersuchen, also z.B. ob für zwei reguläre Sprachen $L_1, L_2 \in \text{REG}$ auch

- $L_1 \cup L_2 \in \text{REG}$
- $L_1^+ \in \text{REG}$
- $\overline{L} \in \text{REG}$
- ...

gilt.

Vorgehen

Die Argumentation ist dann immer "Seien L_1 und L_2 reguläre Sprachen, dann ..., also ist auch $L_1 \circ L_2$ regulär".

Einfache Abschlusseigenschaften

Viele Abschlusseigenschaften kriegen wir durch reguläre Ausdrücke "geschenkt", da wir ja wissen, dass diese reguläre Sprachen beschreiben.

Satz

Die regulären Sprachen sind abgeschlossen gegenüber $\cup, \cdot, +, *$.

Beweis.

Seien $L_1, L_2 \in \text{REG}$. Dann gibt es reguläre Ausdrücke A_1, A_2, die L_1 bzw. L_2 beschreiben. Nach Definition sind nun auch $A_1 + A_2$, $A_1 \cdot A_2$, A_1^+, A_1^* reguläre Ausdrücke, die die Mengen $L_1 \cup L_2$, $L_1 \cdot L_2$, L_1^+ und L_1^* beschreiben. Diese sind folglich auch regulär. (Da jede Sprache, die von einem regulären Ausdruck beschrieben werden kann, regulär ist.) □

Komplement

Und Komplementbildung ... ?

Definition

Sei $L \subseteq \Sigma^*$. Dann ist

$$\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$$

das Komplement von L.

Ideen ?!

Mit vollständigen DFAs?

Komplement

Satz

Die regulären Sprachen sind gegenüber Komplementbildung abgeschlossen, d.h. ist $L \in REG$, dann auch $\overline{L} \in REG$.

Beweis.

Sei $L \in$ REG. Dann gibt es einen vollständigen DFA A mit $L(A) = L$. Wir konstruieren nun einen vollständigen DFA A' aus A wie folgt:

- Alles von A übernehmen
- $z \in Z$ in A Endzustand, dann in A' nicht
- $z \in Z$ in A kein Endzustand, dann einer in A'

Komplement

Satz

Die regulären Sprachen sind gegenüber Komplementbildung abgeschlossen, d.h. ist $L \in REG$, dann auch $\overline{L} \in REG$.

Beweis.

(Wir haben gerade End- und Nicht-Endzustände getauscht.) Wurde ein Wort nun von A akzeptiert (die Rechnung endet in einem Endzustand), so wird es von A' nicht akzeptiert (kein Endzustand). Wurde ein Wort von A nicht akzeptiert (Rechnung endet in einem Nicht-Endzustand), so wird es von A' akzeptiert (Endzustand). Damit gilt $L(A') = \overline{L}$. □

Produkt/Durchschnitt

Satz

Seien $L_1, L_2 \in REG$, dann ist auch $L_1 \cap L_2 \in REG$.

Ideen?

Die Idee

Wir gehen von zwei Automaten aus (für L_1 und L_2). Ein neuer Automat muss dann die Wörter akzeptieren, die *beide* Automaten akzeptieren. Dazu merken wir uns in einem Zustand des neuen Automaten jeweils in welchem Zustand der erste *und* in welchem der zweite Automat ist.

Der Produktautomat

Beweis.

Seien $A_1 = (Z_1, \Sigma_1, \delta_1, z_{1,0}, Z_{1,end})$ $A_2 = (Z_2, \Sigma_2, \delta_2, z_{2,0}, Z_{2,end})$ vollständige DFAs mit $L(A_1) = L_1$ und $L(A_2) = L_2$. Wir konstruieren $C = (Z_3, \Sigma_3, \delta_3, z_{3,0}, Z_{3,end})$ mit

$$\begin{aligned} Z_3 &:= Z_1 \times Z_2 \\ \Sigma_3 &:= \Sigma_1 \cap \Sigma_2 \\ z_{3,0} &:= (z_{1,0}, z_{2,0}) \\ Z_{3,end} &:= Z_{1,end} \times Z_{2,end} \\ \delta_3((z_1, z_2), x) &:= (\delta_1(z_1, x), \delta(z_2, x)) \end{aligned}$$

□

Der Produktautomat

Im Produktautomaten C wird also

- in der ersten Komponente eine Rechnung von A_1 und
- in der zweiten Komponente eine Rechnung von A_2

gemacht. Daraus folgt schnell die Korrektheit:

$$\hat{\delta}_3(z_{3,0}, w) \in Z_{3,end} \Leftrightarrow \hat{\delta}_1(z_{1,0}, w) \in Z_{1,end} \wedge \hat{\delta}_2(z_{2,0}, w) \in Z_{2,end}$$

Dies wäre noch genauer zu zeigen. Z.B. wieder mit einer Induktion über die Wortlänge. Dafür bietet sich obiges aber nicht an. Warum? Was wäre eine bessere Induktionsbehauptung?

Der Produktautomat - Konstruktion

Will man einen Produktautomaten konstruieren, so

- beginnt man in $(z_{1,0}, z_{2,0})$
- berechnet $\delta(z_{1,0}, x)$ und $\delta(z_{2,0}, x)$ für jedes $x \in \Sigma$ (sind für ein x beide definiert, ist dies ein neuer Nachfolgezustand - so lässt sich das Verfahren auch auf nicht vollständige DFAs erweitern)
- So fährt man dann fort ...

Bemerkung

Im Grund genommen ähnlich zur Potenzautomatenkonstruktion, nur dass man hier im ersten Tupelelement immer wie im ersten Automaten und im zweiten Tupelelement immer wie im zweiten Automaten rechnet.

Abschlusseigenschaften - Zusammenfassung

Satz

Die regulären Sprachen sind abgeschlossen gegenüber

- *Vereinigung* $\cup$
- *Konkatenation* $\cdot$
- *"hoch* $+$*"*
- *"hoch* $*$*"*
- *Komplementbildung*
- *Durchschnitt* $\cap$

Für Interessierte

Es gelten weitere Abschlusseigenschaften. Siehe z.B. [HMU].

Möglichkeiten und Grenzen

Wir können nun sehr viele Sprachen konstruieren, die alle regulär sind. Sind z.B. L_1, L_2 regulär. Dann auch (in etwas schluderiger Schreibweise)

- $(L_1 + L_2)^* \cdot L_1 \cdot L_2^+ \cdot L_1^*$
- $(L_1^* \cdot a^3 \cdot L_2^*)^+$
- ...

dennoch wissen wir, dass wir nur endlich viele Informationen speichern können (der DFA hat nur endlich viele Zustände).

Eine Sprache, die nicht geht, oder?

$$L := \{a^n b^n \mid n \in \mathbb{N}\}$$

Ideen für einen DFA?
Welche? Oder was ist das Problem?

Zusammenfassung

Wir haben heute

- **NFAs mit λ-Kanten** und
- **reguläre Ausdrücke**

eingeführt.

Zusammenfassung

Zusammenfassung

Zu jeder regulären Sprache L gibt es

- einen DFA A mit $L(A) = L$
- einen NFA B mit $L(B) = L$
- einen NFA mit λ-Kanten C mit $L(C) = L$
- einen regulären Ausdruck D, der L beschreibt ($M_D = L$)

DFAs, NFAs, NFAs mit λ-Kanten und reguläre Ausdrücke sind also äquivalent.

Zusammenfassung

Sollt ihr wissen

NFA mit λ-Kanten und reguläre Ausdrücke solltet ihr lesen und benutzen können. Ihr solltet wissen, dass man die in DFAs umwandeln kann.

Könnt ihr wissen...

Interessierte können die Konstruktionen im Detail und die zugehörigen Beweise in [HMU] finden.

Zusammenfassung

Wir haben uns dann noch Abschlusseigenschaften angesehen:

Satz

Die regulären Sprachen sind (u.a.) abgeschlossen gegenüber

- *Vereinigung* $\cup$
- *Konkatenation* $\cdot$
- *"hoch* $+$*"*
- *"hoch* $*$*"*
- *Komplementbildung*
- *Durchschnitt* $\cap$
- *...*