Formale Grundlagen der Informatik 1
Kapitel 3
Mehr zu regulären Sprachen

Frank Heitmann
heitmann@informatik.uni-hamburg.de

14. April 2015

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

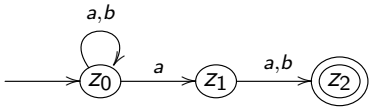
$$\begin{aligned} \hat{\delta}(Z', \lambda) &:= Z' \\ \hat{\delta}(Z', xw) &:= \cup_{z \in Z'} \hat{\delta}(\delta(z, x), w) \end{aligned}$$

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

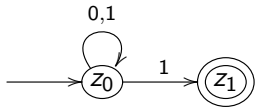
Definition (Akzeptierte Sprache)

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

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

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
- $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_0, \lambda)$ – ablehnen
- $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_1, \lambda)$ – akzeptieren!

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

Äquivalenz?

Anmerkung

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). Alternativ geht auch:

$$K := \{(z,x,z') \mid z \in Z, x \in \Sigma, z' = \delta(z,x)\}$$

Aber geht das auch andersherum ??

Hier endeten wir gestern ...

... und machen jetzt weiter ...

Die Idee

Ausgehend von der Arbeitsweise eines NFA macht es Sinn

- Sich die Zustände, in denen der NFA nach Lesen eines Teilwortes sein kann, zu merken (z.B. in einer Menge).

Aufgrund der Akzeptanzbedingung eines NFA

- Sollten wir genau dann akzeptieren, wenn in der gemerkten Menge ein Endzustand auftritt.

Erinnern wir uns an die Konstruktionsmethoden von gestern:

- Ist $|Z| = n$, so gibt es 2^n Teilmengen von Z. Dies sind endlich viele! Wir haben also nur endliche viele Informationen, die wir uns merken müssen!
- Die Endzustände sind dann leicht rauszufinden.
- Die Übergangsfunktion (des DFA) kann dann fast so wie $\hat{\delta}$ (des NFA) definiert werden.

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

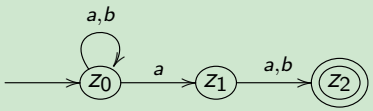
1. $Z' := 2^Z$
2. $\Sigma' := \Sigma$
3. $\delta'(M,x) := \cup_{z \in M} \delta(z,x)$ oder alternativ $\delta'(M,x) := \cup_{z \in M}\{z' \in Z \mid (z,x,z') \in K\}$
4. $z_0 := Z_{start}$
5. $Z'_{end} := \{M \in 2^Z \mid M \cap Z_{end} \neq \emptyset\}$

Der Potenzautomat - Anmerkungen

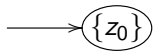
Anmerkungen zur Konstruktion:

- Z' ist die Menge aller Teilmengen von Z.
- Das Eingabealphabet wird übernommen.
- Startzustand des DFA ist gerade die Menge der Startzustände des NFA (denn in einem dieser Zustände startet ja jede Rechnung des NFA).
- Endzustände des DFA sind all jene Mengen, die mindestens einen Endzustand des NFA enthalten (denn, wenn wir uns in der Menge merken, wo der NFA gerade potentiell sein könnte und dort ein Endzustand dabei ist, dann akzeptieren wir ja!)
- Die Überführungsfunktion geht alle Zustände in der aktuellen Menge durch, schaut, wo wir von dort mit dem aktuellen Symbol hinkommen und tut all die dabei herauskommenden Zustände in eine Menge.

Der Potenzautomat - Ein Beispiel

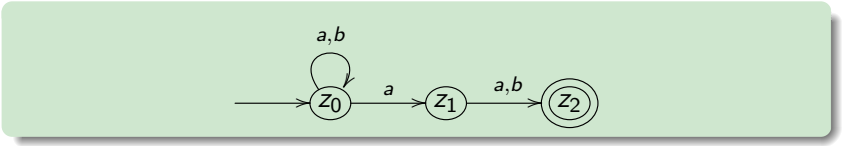


Wir bauen nur die initiale Zusammenhangskomponente und beginnen mit dem Startzustand:

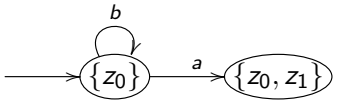


Wohin gelangen wir mit einem a? Wohin mit einem b?

Der Potenzautomat - Ein Beispiel

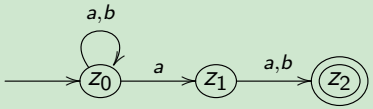


Mit a zu $\{z_0, z_1\}$, mit b zu $\{z_0\}$

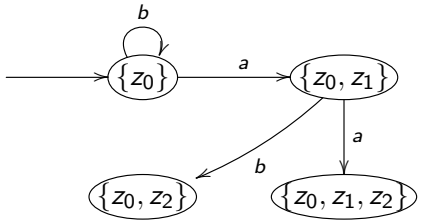


Wohin nun von $\{z_0, z_1\}$ mit a und b?

Der Potenzautomat - Ein Beispiel

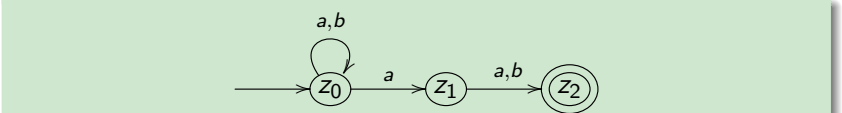


Es ist $\delta(z_0, a) = \{z_0, z_1\}$ und $\delta(z_1, a) = z_2$ also insgesamt $\delta'(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$ und ähnlich für b:

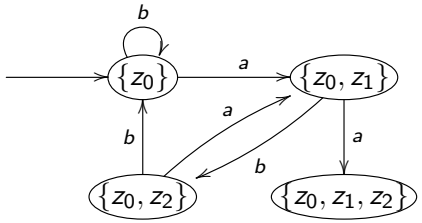


Wohin nun von $\{z_0, z_2\}$ mit a und b?

Der Potenzautomat - Ein Beispiel

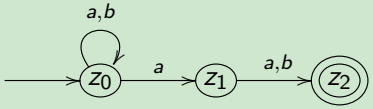


Es ist $\delta(z_0, b) = \{z_0\}$ und $\delta(z_2, b) = \emptyset$ also insgesamt $\delta'(\{z_0, z_2\}, b) = \{z_0\}$ und ähnlich für a:

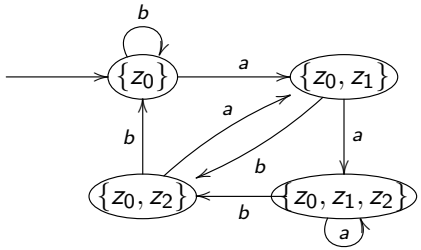


Und von $\{z_0, z_1, z_2\}$?

Der Potenzautomat - Ein Beispiel

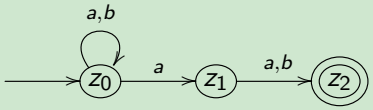


Es ist $\delta(z_0, b) = \{z_0\}$, $\delta(z_1, b) = \{z_2\}$ und $\delta(z_2, b) = \emptyset$ also insgesamt $\delta'(\{z_0, z_1, z_2\}, b) = \{z_0, z_2\}$ und ähnlich für a:

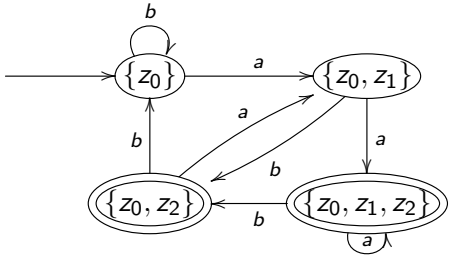


Endzustände?

Der Potenzautomat - Ein Beispiel



Und fertig!



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 $L(A) = L(B)$. □

Das Ergebnis

Satz

DFAs und NFAs sind äquivalente Automatenmodelle. Beide akzeptieren damit die regulären Sprachen.

Anmerkung

Akzeptieren zwei DFAs A_1 und A_2 die gleiche Sprache, gilt also $L(A_1) = L(A_2)$, so sagt man A_1 und A_2 sind *äquivalent*.
Ebenso sprechen wir auch bei zwei Automaten unterschiedlicher Automatenmodelle davon, dass sie äquivalent sind, wenn sie die gleiche Sprache akzeptieren.
Ist es so wie hier möglich zu jedem Automaten des einen Modells einen äquivalenten Automaten des anderen Modells zu konstruieren, so sagen wir, dass die Automatenmodelle äquivalent sind.

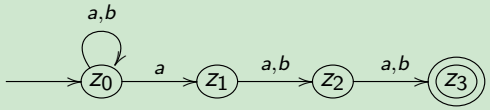
Eine letzte Anmerkung dazu...

Anmerkung

Der Potenzautomat hat nach Konstruktion exponentiell mehr Zustände als der NFA ($2^{|Z|}$ im Vergleich zu $|Z|$). Oft werden diese nicht alle gebraucht, es gibt aber Beispiele wo diese Zahl erreicht wird. Die hohe Zahl ist also keine Schwäche der Konstruktion!

Bemerkung

Eine gute Näherung hat man schon mit:



bzw. Verallgemeinerungen davon. Ein äquivalenter DFA für diese Sprache hat mindestens 2^{n-1} Zustände.

Zusammenfassung

Wichtige Anmerkung

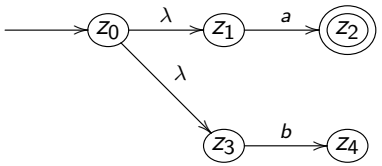
- Jeder DFA *akzeptiert* eine Sprache. Man kann auch sagen, dass er eine (kompakte) *Beschreibung* dieser Sprache ist bzw. jener Wörter, die in der Sprache enthalten sind.
- Eine Menge/Sprache ist gerade dann **regulär**, wenn es einen DFA gibt, der sie akzeptiert. Regulär zu "sein" ist also gerade über die Automaten definiert!
- Die **Familie der regulären Sprachen** ist dann die "Ansammlung" all jener Sprachen für die es einen DFA gibt, der sie akzeptiert.
- Wir haben nun gelernt: Der NFA tut auch genau dies! Die Aussage eine Sprache ist regulär ist also gleichbedeutend damit, dass es einen DFA gibt, der sie akzeptiert oder damit, dass es einen NFA gibt, der sie akzeptiert.

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.



Lambda-Kanten

Lasst euch von der Einfachheit des Beispiels nicht täuschen. Die Kanten können sehr nützlich sein.

Begriffe wie Rechnung etc. wären nun anzupassen

Wissen

Man kann zeigen, dass man einen NFA mit λ-Kanten in einen λ-freien NFA umwandeln kann (und letztendlich dann auch in einen DFA).

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 im Skript und in [HMU] (siehe Literaturhinweis zu Anfang).

Reguläre Sprachen

Bisher kennen wir zur Beschreibung einer regulären Sprache

- DFAs
- NFAs
- NFAs mit λ-Kanten

Wir lernen jetzt noch eine dritte 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.

Bemerkung

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

Äquivalenzbeweis

Hinweis

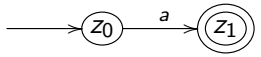
Wir skizzieren jetzt recht knapp beide Richtungen. Für später genügt es, wenn ihr wisst, dass die Äquivalenz besteht. Trotzdem hier jetzt einmal die Idee(n) und die Literaturhinweise für Interessierte...

Äquivalenzbeweis - Die Idee 1

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

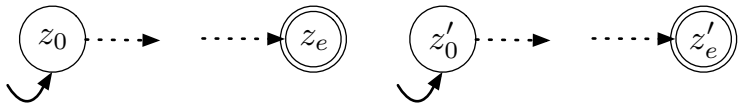
- Zu jeder Operation bei der Erstellung eines regulären Ausdrucks gibt man ein Verfahren an, wie man einen NFA konstruieren kann.

Z.B. zum rationalen Ausdruck a einfach



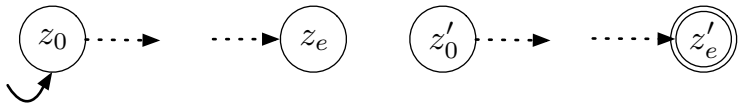
Äquivalenzbeweis - Die Idee 1

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



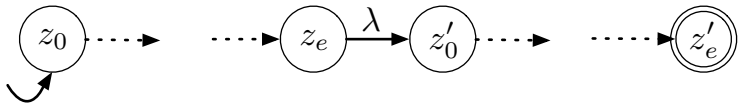
Äquivalenzbeweis - Die Idee 1

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



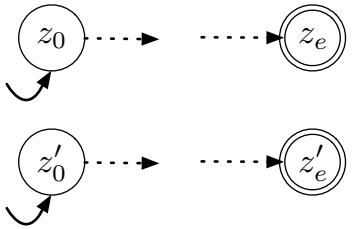
Äquivalenzbeweis - Die Idee 1

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



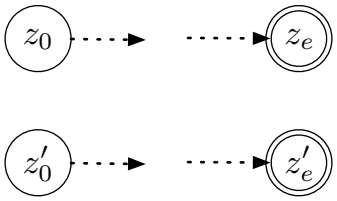
Äquivalenzbeweis - Die Idee 1

Z.B. zum rationalen Ausdruck $A + B$



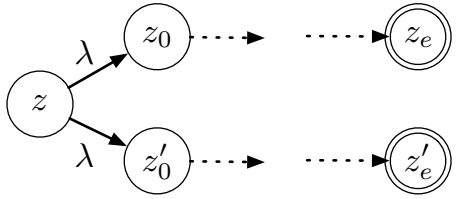
Äquivalenzbeweis - Die Idee 1

Z.B. zum rationalen Ausdruck $A + B$



Äquivalenzbeweis - Die Idee 1

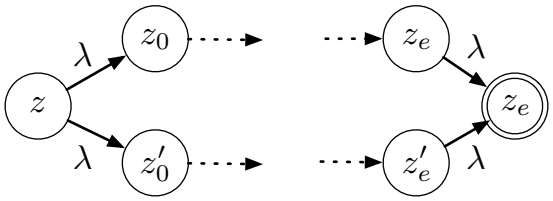
Z.B. zum rationalen Ausdruck $A + B$



(z ist neuer Startzustand!)

Äquivalenzbeweis - Die Idee 1

Z.B. zum rationalen 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] (und auf andere Weise im Skript).

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

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

In $R^k_{i,j}$ 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 $R^n_{1,x}$ mit $z_x \in Z_{end}$, vereinigt diese und ist fertig.

Literaturhinweis

Interessierte finden die Konstruktion im Detail inkl. Beweis im Skript und 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.

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

Oft schreibt man die Operatoren in Infix-Notation (d.h. den Operator zwischen die Argumente). Beispiele in einem anderen Kontext sind z.B. die Addition bei den natürlichen Zahlen.

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:

$$\hat{\delta_3}(z_{3,0}, w) \in Z_{3,end} \Leftrightarrow \hat{\delta_1}(z_{1,0}, w) \in Z_{1,end} \wedge \hat{\delta_2}(z_{2,0}, w) \in Z_{2,end}$$

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 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].

Eine Anmerkung

Bemerkung

Im Skript wird dies teilweise andersherum gemacht. Wir haben hier mittels regulären Ausdrücken einige der Abschlusseigenschaften sofort gekriegt. Im Skript wird zunächst gezeigt, dass z.B. REG gegenüber $\cup$ abgeschlossen ist und dies dann beim Beweis benutzt, dass die regulären Ausdrücke und DFAs äquivalent sind.

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)

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

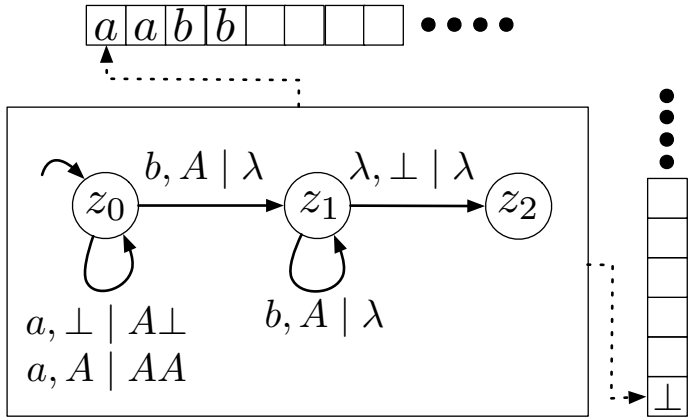
dennoch wissen wir, dass wir nur endlich viele Informationen speichern können (der DFA hat nur endlich viele Zustände).

Eine Sprache, die nicht geht, oder?

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

Ideen für einen DFA?
Was ist das Problem?

Der Kellerautomat



Ausblick

Nächstes Mal:

- Kellerautomat einführen ($\Rightarrow$ Lesestoff dazu ab Mittwoch!)
- Werkzeuge kennenlernen, um zu zeigen, dass es wirklich nicht mit DFAs geht

Zusammenfassung

Wir haben heute

- die **Potenzautomatenkonstruktion** kennengelernt und
- ihre Korrektheit bewiesen und
- dabei die Idee einer **Induktion über die Wortlänge** kennengelernt.

Diese Konstruktion solltet ihr einordnen und ausführen können!

Zusammenfassung

Außerdem haben wir heute

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

eingeführt

Sollt ihr wissen

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

Könnt ihr wissen...

Interessierte können die Konstruktionen im Detail und die zugehörigen Beweise im Skript finden.

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

Wir haben uns dann noch Abschlusseigenschaften angesehen:

Satz

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

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

Zusammenfassung

Was ihr bis zur nächsten Woche kennen solltet:

- Was DFAs sind und wie die funktionieren
- Was NFAs sind und wie die funktionieren
- Was reguläre Ausdrücke sind und wie man damit Sprachen beschreibt.
- λ-Kanten kennen
- Wissen, dass DFA, NFA, NFA mit λ-Kanten und reguläre Ausdrücke im Prinzip "das Gleiche" sind.
- Konstruktionsmethoden (zu einer Sprache M einen Automaten A machen)
- und der Beweis $L(A) = M$.
- Potenzautomatenkonstruktion (die Konstruktion)
- Abschlusseigenschaften für REG. Welche gelten? Wie gehen die Konstruktionen?