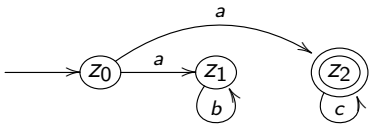
Formale Grundlagen der Informatik 1
Kapitel 4
DFAs und NFAs

Frank Heitmann
heitmann@informatik.uni-hamburg.de

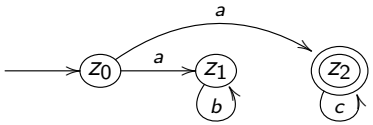
12. April 2016

Ein NFA...

Beim DFA gibt es zu einem $z \in Z$ und einem $x \in \Sigma$ einen Nachfolgezustand $\delta(z, x)$. Der Nachfolgezustand ist also *determiniert*. Dies kann man aufweichen...



Ein NFA...

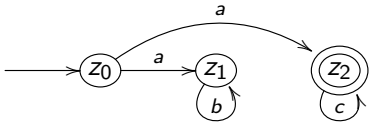


Wie würdet ihr die Überführungsfunktion definieren?

1. $\delta : Z \times \Sigma \to Z$
2. $\delta : Z \times \Sigma \to Z^*$
3. $\delta : Z \times \Sigma \to 2^Z$
4. $\delta : 2^{Z \times \Sigma} \to Z$
5. ganz anders ...

Nichtdeterministische Automaten

Beim DFA gibt es zu einem $z \in Z$ und einem $x \in \Sigma$ einen Nachfolgezustand $\delta(z, x)$. Der Nachfolgezustand ist also *determiniert*. Dies kann man aufweichen...



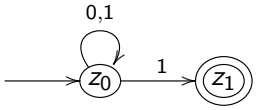
Nach Lesen von *a* ist der Automat *nichtdeterministisch* sowohl in z_1 als auch in z_2.

Informal akzeptiert der nichtdeterministische, endliche Automat ein Wort dann, wenn es *irgendeine Rechnung* gibt, in der er einen Endzustand erreicht.

Sinn?

ABER WARUM?!?

Man kann damit oft sehr schnell Automaten entwerfen. Z.B. für die Sprache vom letzten Mal (Wörter, die auf 1 enden):



... und wir werden später noch mehrere Stellen sehen, an denen das Konzept des Nichtdeterminismus hilfreich ist ...

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:

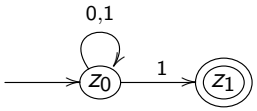
- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der **Zustandsübergangsrelation** $K \subseteq Z \times \Sigma \times Z$
- Der Menge der *Startzustände* $Z_{start} \subseteq Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

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

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



$\hat{\delta}(\{z_0\}, 011) = \hat{\delta}(\{z_0\}, 11) = \hat{\delta}(\{z_0, z_1\}, 1) =$
$\hat{\delta}(\{z_0, z_1\}, \lambda) \cup \hat{\delta}(\emptyset, \lambda) = \{z_0, z_1\}$

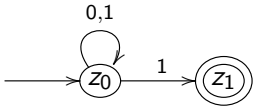
Überführungsfunktion und Rechnung

Anmerkung

Man kann die erweiterte Überführungsfunktion auch wie folgt definieren:

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

Diese Darstellung hat den Vorteil, dass sich im ersten Argument die Menge der bisher erreichten Zustände sammelt.



$\hat{\delta}(\{z_0, z_1\}, 1) = \hat{\delta}(\{z_0, z_1\} \cup \emptyset, \lambda) = \ldots$

Überführungsfunktion und Rechnung

Definition (Konfiguration)

1. Eine **Konfiguration** eines NFA A ist ein Tupel $(z, w) \in Z \times \Sigma^*$ mit der Bedeutung, dass A im Zustand z ist und noch das Wort w zu lesen ist.
2. Ein **Konfigurationsübergang** ist dann

$$(z, w) \vdash (z', v)$$

gdw. $w = xv$, $x \in \Sigma$ und $z' \in \delta(z, x)$ ist.

Wichtige Anmerkung

Achtung: Anders als beim DFA gehen hier bei der Konfiguration Informationen verloren! Ein NFA kann sozusagen in mehreren Konfigurationen gleichzeitig sein. Der Sinn hier ist, dass man über bestimmte Rechnungen ähnlich wie beim DFA sprechen will.

Überführungsfunktion und Rechnung

Definition (Rechnung)

1. Eine **Rechnung** auf dem Wort $w \in \Sigma^*$ ist eine Folge von Konfigurationsübergängen, die in (z_0, w) mit $z_0 \in Z_{start}$ beginnt.
2. Endet die Rechnung in (z', λ) und ist $z' \in Z_{end}$, so ist dies eine **Erfolgsrechnung**.
3. Existiert auf einem Eingabewort w eine Erfolgsrechnung, d.h. gibt es eine Rechnung, die in (z_0, w) mit $z_0 \in Z_{start}$ beginnt und in (z_e, λ) mit $z_e \in Z_{end}$ endet, dann wird w akzeptiert.

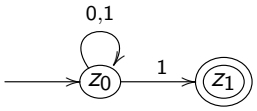
Akzeptierte Sprache

Definition (Akzeptierte Sprache)

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

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

Ein Beispiel

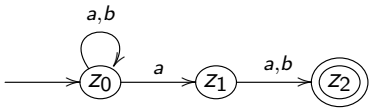


Alle möglichen Rechnungen auf dem Wort 011:

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

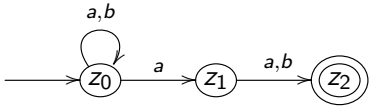
Fragen...



Was ist $\hat{\delta}(\{z_0\}, aaa)$?

1. z_2
2. $\{z_2\}$
3. $\{z_0, z_2\}$
4. $\{z_0, z_1\}$
5. $\{z_0, z_1, z_2\}$

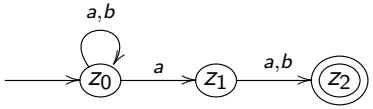
Fragen...



Was ist $\hat{\delta}(\{z_0\}, aba)$?

1. z_2
2. $\{z_2\}$
3. $\{z_0, z_2\}$
4. $\{z_0, z_1\}$
5. $\{z_0, z_1, z_2\}$

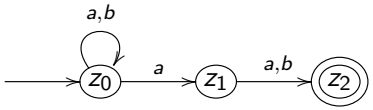
Fragen...



Gibt es eine Rechnung auf aaa, bei der A blockiert?

1. Ja!
2. Nein!

Fragen...



Welche Sprache akzeptiert dieser NFA?

1. $\{a,b\}^*\{a\}\{a,b\}^*$
2. $\{a,b\}\{a\}\{a,b\}^*$
3. $\{a,b\}^*\{a\}\{a,b\}$
4. $\{a,b\}\{a\}\{a,b\}$

Zur Nachbereitung

Zur Nachbereitung

Richtige Antworten sind:

1. 5
2. 4
3. 1
4. 3

Äquivalenz?

Pause to Ponder

Können NFAs alles, was DFAs können? Können NFAs mehr? (D.h. sind NFAs mächtiger als DFAs?) Oder andersherum?

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 ??

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 DFAs:

- 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

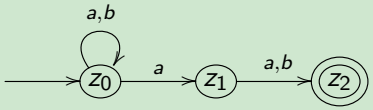
1. $Z' := 2^Z$
2. $\Sigma' := \Sigma$
3. $\delta'(M, x) := \cup_{z \in M} \delta(z, x)$ oder alternativ $\delta'(M, x) := \cup_{z \in M} \{z' \in Z \mid (z, x, z') \in K\}$
4. $z_0 := Z_{start}$
5. $Z'_{end} := \{M \in 2^Z \mid M \cap Z_{end} \neq \emptyset\}$

Der Potenzautomat - Anmerkungen

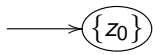
Anmerkungen zur Konstruktion:

- 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

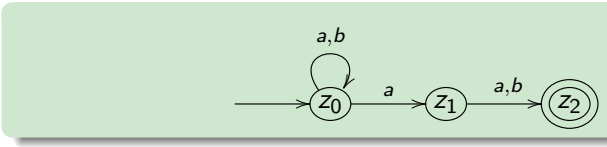


Wir bauen nur die initiale Zusammenhangskomponente und beginnen mit dem Startzustand:

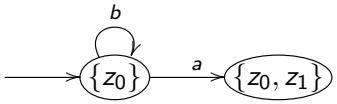


Wohin gelangen wir mit einem a? Wohin mit einem b?

Der Potenzautomat - Ein Beispiel

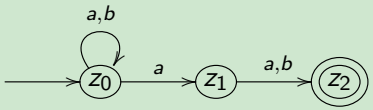


Mit a zu $\{z_0, z_1\}$, mit b zu $\{z_0\}$

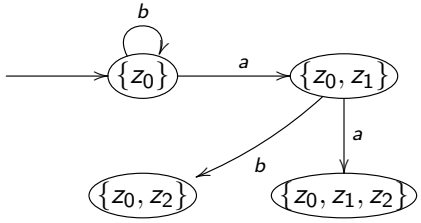


Wohin nun von $\{z_0, z_1\}$ mit a und b?

Der Potenzautomat - Ein Beispiel

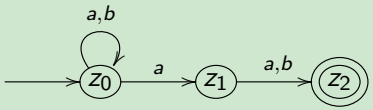


Es ist $\delta(z_0, a) = \{z_0, z_1\}$ und $\delta(z_1, a) = z_2$ also insgesamt $\delta'(\{z_0, z_1\}, a) = \{z_0, z_1, z_2\}$ und ähnlich für b:

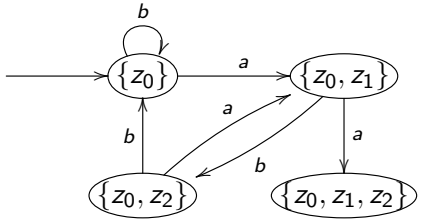


Wohin nun von $\{z_0, z_2\}$ mit a und b?

Der Potenzautomat - Ein Beispiel

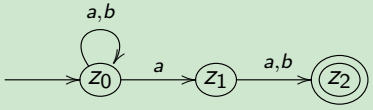


Es ist $\delta(z_0, b) = \{z_0\}$ und $\delta(z_2, b) = \emptyset$ also insgesamt $\delta'(\{z_0, z_2\}, b) = \{z_0\}$ und ähnlich für a:

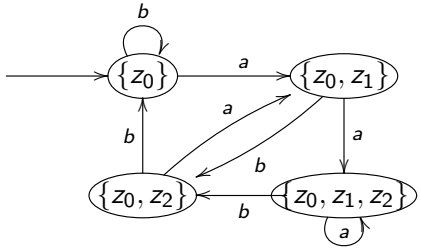


Und von $\{z_0, z_1, z_2\}$?

Der Potenzautomat - Ein Beispiel

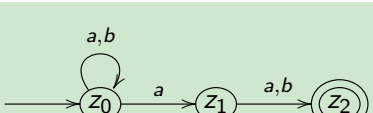


Es ist $\delta(z_0, b) = \{z_0\}$, $\delta(z_1, b) = \{z_2\}$ und $\delta(z_2, b) = \emptyset$ also insgesamt $\delta'(\{z_0, z_1, z_2\}, b) = \{z_0, z_2\}$ und ähnlich für a:

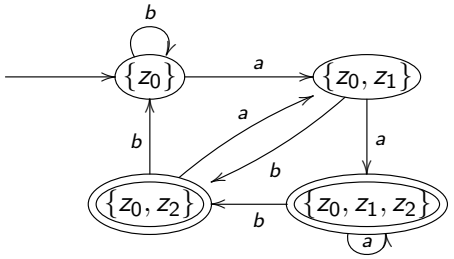


Endzustände?

Der Potenzautomat - Ein Beispiel



Und fertig!



Auf einen Blick

Überführungsfunktion und akzeptierte Sprache

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

$$L(A) := \{w \in \Sigma^* \mid \hat{\delta}(Z_{start},w) \cap Z_{end} \neq \emptyset\}$$

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 - 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.

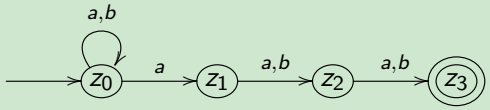
Eine letzte Anmerkung dazu...

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

In der Konstruktion im Beweis wurde $Z' := 2^Z$ gesetzt. Im Beispiel haben wir nur die initiale Zusammenhangskomponente konstruiert. Dies genügt auch stets. Man könnte die Konstruktion im Beweis so umformulieren, dass man sie algorithmisch formuliert und bei $z_0 := Z_{start}$ beginnt und dann (so wie im Beispiel) nach und nach weitere, erreichbare Zustände (also insgesamt die initiale Zusammenhangskomponente) generiert. Der Korrektheitsbeweis geht dann ganz genauso. In der worst-case Abschätzung der vorherigen Folie würde man dann nicht sagen, dass man nach Konstruktion exponentiell viele Zustände hat, sondern dass im schlimmsten Fall exponentiell viele konstruiert werden müssen (was mit dem gleichen Beispiel belegt werden kann). Die Aussagen bleiben also die gleichen. Die Variante mit der initialen Zusammenhangskomponente ist aber schneller und platzsparender.

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.

Zusammenfassung

Wir haben gestern und heute

- Wiederholt (Begriffe bei DFAs, Konstruktionsmethoden)
- **NFAs** kennengelernt
- Die **Potenzautomatenkonstruktion** kennengelernt und
- ihre Korrektheit bewiesen und
- dabei die Idee einer **Induktion über die Wortlänge** kennengelernt.

Zusammenfassung

Zusammenfassung

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.

Zusammenfassung

Was ihr bis zur nächsten Woche kennen solltet:

- Was DFAs sind und wie die funktionieren
- Was NFAs sind und wie die funktionieren
- Wissen, dass DFA und NFA im Prinzip "das Gleiche" sind.
- Konstruktionsmethoden (zu einer Sprache M einen Automaten A machen)
- und der Beweis $L(A) = M$.
- Potenzautomatenkonstruktion (die Konstruktion)