arbeiten und hier kann dann ggf. auch wieder auf die Konstruktion verwiesen werden.

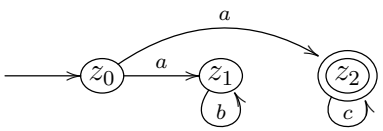
Fragen

1. Wie viele Informationen kann man in einem DFA speichern?
   a) beliebig viele
   b) endlich viele
   c) so viele, wie man Zustände hat
   d) so viele, wie man Symbole in Σ hat
2. Wie kann man $L(A) = M$ beweisen?
   a) Für jedes $w \in L(A)$ zeigen, dass $w \in M$ gilt und andersherum.
   b) Für ein $w \in L(A)$ zeigen, dass $w \in M$ gilt und andersherum.
   c) Für ein beliebiges $w \in M$ zeigen, dass $w \in L(A)$ gilt und andersherum.
   d) Für ein festes $w \in M$ zeigen, dass $w \in L(A)$ gilt und andersherum.

2.2 Nichtdeterministische endliche Automaten

Bei einem vollständigen DFA gibt es zu jedem Zustand z, in dem sich der DFA befindet, und jedem Eingabesymbol x, das er in diesem Zustand liest, genau einen Nachfolgezustand $\delta(z, x) = z'$. Ist der DFA nicht vollständig, dann gibt es höchstens einen Nachfolgezustand. Man spricht daher davon, dass der Nachfolgezustand beim DFA *determiniert* ist (darum auch die Bezeichnung als *deterministischer* endlicher Automat). Ist ein DFA A nämlich in einem Zustand z, dann gibt es stets nur genau eine Möglichkeit, was bei einem Symbol x passiert. Entweder passiert der Zustandswechsel zu einem ganz bestimmten Zustand oder der Automat blockiert.

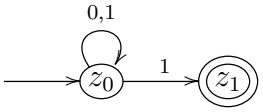
In den späten 1950er Jahren kam die Idee auf, der Übergangsfunktion des DFA mehr Freiheiten einzuräumen. Statt nur einen Nachfolgezustand zu haben, darf der neue Automat nun beim Lesen eines Buchstaben mehrere Nachfolgezustände haben.



Im abgebildeten Automaten gibt es in z_0 zwei Möglichkeiten den Zustand beim Lesen von a zu wechseln. Die Idee ist, dass *nichtdeterministisch* ein Nachfolgezustand gewählt wird. Man interpretiert dies so, dass der Automat sowohl

in z_1 als auch in z_2 ist und quasi in der Rückschau das bessere ausgewählt wird. Akzeptiert wird daher, informal ausgedrückt, sobald es mindestens eine Rechnung gibt, die den Automaten vom Start- in einen Endzustand bringt. Endliche Automaten, die wie der abgebildete Automaten mehrere gleichbeschriftete Kanten aus einem Zustand heraus haben, also nichtdeterministisch sind, werden als NFA (nichtdeterministischer endlicher Automat) bezeichnet.

Nichtdeterminismus einzuführen war eine wegweisende Idee, die uns an vielen Stellen in der Informatik begegnet. Hier ist sie insb. deswegen nützlich, weil sie uns oft erlaubt, auf einfache Weise Automaten zu entwerfen, die als DFAs viel komplizierter sind. Will man z.B. einen Automaten für alle Worte, die auf 1 enden, so kann man dafür schnell den abgebildeten Automaten konstruieren.



Der DFA ist in diesem Fall nicht viel komplexer, aber man versuche dies einmal zu verallgemeinern und z.B. einen DFA zu entwerfen, der jene Worte akzeptiert, dessen fünftletzter Buchstabe 1 ist. Dies ist ein recht großer und schwer zu lesender DFA, mit Nichtdeterminismus aber sehr schnell konstruiert.

Zudem werden wir Nichtdeterminismus bei späteren Automatenmodellen benötigen und insb. bei der Turingmaschine und der damit zusammenhängenden Komplexitätstheorie wird dieses Konzept unglaublich hilfreich sein. Mit ihm lassen sich einige der ganz großen offenen Fragen der Informatik formulieren.

2.2.1 Definition des NFA

Den nichtdeterministischen endlichen Automaten (kurz: NFA) können wir formal fast wie den DFA einführen. Wir müssen nur bei der Übergangsfunktion etwas anders machen, um mehrere Nachfolgezustände ausdrücken zu können. Dies gelingt indem wir die Zielmenge von Z zu 2^Z ändern. Damit ist $\delta(z, x)$ dann eine *Menge* von Zuständen. In dieser Menge sind dann gerade jene Zustände, die von z aus mit x erreichbar sind.

Zusätzlich wollen wir auch eine Menge von Startzuständen zulassen. Dies beschreibt, dass wir auch nach dem Lesen von null Symbolen nichtdeterministisch in mehreren Zuständen sein können, das also bereits der Anfang nichtdeterminiert ist.

Formal können wir dies nun wie folgt exakt ausdrücken.

Definition 2.2.1 (Nichtdeterministischer endlicher Automat (NFA))**.** *Ein* nichtdeterministischer, endlicher Automat *(NFA) ist ein* 5*-Tupel*

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

mit:

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

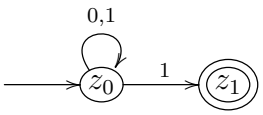
Die graphische Darstellung ist wie beim DFA.

Alternativ kann statt der Überführungsfunktion δ auch mit der Zustandsübergangsrelation

$$K \subseteq Z \times \Sigma \times Z$$

arbeiten. Ein Tupel (z, x, z') drückt dann aus, das es von z eine Kante nach z' gibt, die mit x beschriftet ist. Es ist dann bei einem NFA möglich weitere Kanten mit gleichem Ursprung und gleichem Symbol zu haben, also z.B. zwei Kanten (z, x, z') und (z, x, z'').

Im abgebildeten Automaten ist dann $Z = \{z_0, z_1\}$, $\Sigma = \{0, 1\}$ und $Z_{end} = \{z_1\}$ wie beim DFA.



Da dies ein NFA ist, haben wir aber, obwohl es in diesem Fall wie beim DFA nur ein Startzustand ist, eine Startzustandsmenge, also ist $Z_{start} = \{z_0\}$. Die Überführungsfunktion ist durch $\delta(z_0, 0) = \{z_0\}$ und $\delta(z_0, 1) = \{z_0, z_1\}$ gegeben. Außerdem kann man $\delta(z_1, 0) = \delta(z_1, 1) = \emptyset$ setzen.

Als Zustandsübergangsrelation ist K gegeben durch $K = \{(z_0, 0, z_0), (z_0, 1, z_0), (z_0, 1, z_1)\}$.

Da wir die Überführungsfunktion angepasst haben, müssen wir auch die erweiterte Überführungsfunktion und letztendlich die akzeptierte Sprache anpassen. Für die erweiterte Überführungsfunktion definieren wir folgendes.

Definition 2.2.2 (Erweiterte Überführungsfunktion)**.** *Sei* $A = (Z, \Sigma, \delta, Z_{start}, Z_{end})$ *ein NFA. 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*

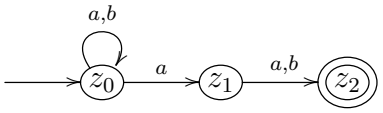
$$\begin{aligned} \hat{\delta}(Z', \lambda) &:= Z' \\ \hat{\delta}(Z', xw) &:= \cup_{z \in Z'} \hat{\delta}(\delta(z, x), w) \end{aligned}$$

oder alternativ durch

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

Wir werden meistens mit der zweiten Alternative arbeiten. Hier sammeln sich die nichtdeterministisch erreichten Zustände in der Menge im ersten Argument.

Sei der folgende Automat gegeben. Wir haben einen Startzustand und interessieren uns für das Wort aba.



Damit ergibt sich

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

Man beachte, wie sich im ersten Argument die Menge der erreichten Zustände sammelt. Um z.B. $\hat{\delta}(\{z_0, z_1\}, ba) = \hat{\delta}(\{z_0\} \cup \{z_2\}, a)$ zu sehen, muss man die Nachfolgezustände von $\{z_0, z_1\}$ beim Lesen des nächsten Buchstabens (also von b) ermitteln. Dazu nimmt man die Menge der Zustände, die von z_0 aus durch b erreicht werden (also $\delta(z_0, b) = \{z_0\}$) und vereinigt diese mit der Menge der Zustande, die von z_1 durch b aus erreichbar sind (also mit $\delta(z_1, b) = \{z_2\}$). Damit ergibt sich obiges.

Bevor wir zur akzeptierten Sprache kommen, können wir noch wie beim DFA die Konfiguration, Konfigurationsübergänge, sowie Rechnungen und Erfolgsrechnungen definieren.

Definition 2.2.3 (Konfiguration und Rechnung)**.** *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*

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

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

*3. 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.*

*4. Endet eine Rechnung in (z', λ) und ist $z' \in Z_{end}$, so ist dies eine **Erfolgsrechnung**.*

Man beachte wie gering die Unterschiede zum DFA sind. Beim DFA wird beim Konfigurationsübergang $z' = \delta(z, x)$ gefordert, hier wird $z' \in \delta(z, x)$ gefordert. Bei der Rechnung wird gefordert, dass z_0 ein Startzustand ist (wie beim DFA auch, aber hier könnte man mehrere zur Auswahl haben).

Man beachte insbesondere aber auch, dass anders als beim DFA hier bei der Konfiguration Informationen verloren gehen! Ein NFA kann sozusagen in mehreren Konfigurationen gleichzeitig sein. Der Sinn hier ist, dass man über bestimmte Rechnungen ähnlich wie beim DFA sprechen will.

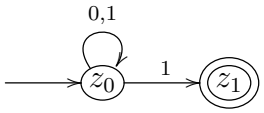
Wir können nun die akzeptierte Sprache definieren.

Definition 2.2.4 (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 NFA A akzeptiert also ein Eingabewort w dann, wenn auf w eine Erfolgsrechnung existiert, d.h. wenn es eine Rechnung gibt, die in (z_0, w) mit $z_0 \in Z_{start}$ beginnt und in (z_e, λ) mit $z_e \in Z_{end}$ endet. Die Notation $\hat{\delta}(Z_{start}, w) \cap Z_{end} \neq \emptyset$ drückt das gleiche aus, nutzt aber die erweiterte Überführungsfunktion.

Betrachten wir noch einmal den folgenden Automaten A.



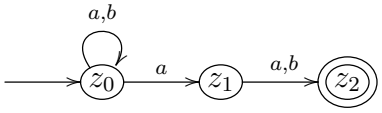
Wir betrachten das Wort $w = 011$. Auf diesem Wort hat der Automat drei Rechnungen:

1. $(z_0, 011) \vdash (z_0, 11) \vdash (z_1, 1)$ – blockiert
2. $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_0, \lambda)$ – lehnt ab
3. $(z_0, 011) \vdash (z_0, 11) \vdash (z_0, 1) \vdash (z_1, \lambda)$ – akzeptiert

Im ersten Fall blockiert der Automat, da es in z_1 keine 1-Kante gibt. Im zweiten Fall lehnt der Automat das Wort ab, da die Rechnung in z_0 endet und dies kein Endzustand ist. Im dritten Fall ist das Wort zu Ende gelesen und der Automat in einem Endzustand, also wird das Wort akzeptiert. Wegen dieses dritten Falles wird das Wort auch ganz allgemein akzeptiert, soll heißen: es gilt $001 \in L(A)$. Man mache sich noch einmal klar, dass ein NFA ein Wort akzeptiert, *sobald es irgendeine Rechnung gibt, die ihn von einem Start- zu einem Endzustand bringt*. Ein NFA muss, um ein Wort zu akzeptieren, dieses wie ein DFA zu Ende gelesen haben und er muss dann in einem Endzustand sein. Er kann aber auf einem Wort mehrere Rechnungen haben und es genügt, wenn mindestens eine davon eine Erfolgsrechnung ist.

Fragen

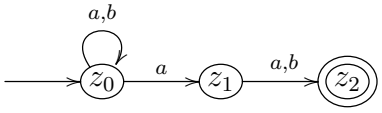
1. Sei folgender NFA gegeben



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

   a) z_2
   b) $\{z_2\}$
   c) $\{z_0, z_2\}$
   d) $\{z_1, z_2\}$
   e) $\{z_0, z_1\}$
   f) $\{z_0, z_1, z_2\}$

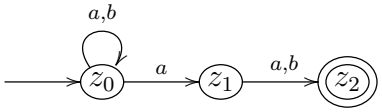
2. Sei folgender NFA gegeben



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

   a) z_2
   b) $\{z_2\}$
   c) $\{z_0, z_2\}$
   d) $\{z_1, z_2\}$
   e) $\{z_0, z_1\}$
   f) $\{z_0, z_1, z_2\}$

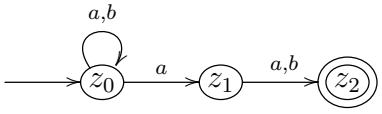
3. Sei folgender NFA gegeben



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

   a) Ja!
   b) Nein!

4. Sei folgender NFA gegeben



   Welche Sprache akzeptiert dieser NFA?

   a) $\{a, b\}^*\{a\}\{a, b\}^*$
   b) $\{a, b\}\{a\}\{a, b\}^*$
   c) $\{a, b\}^*\{a\}\{a, b\}$
   d) $\{a, b\}\{a\}\{a, b\}$