

# Formale Grundlagen der Informatik 1

## Kapitel 2

### Endliche Automaten und reguläre Sprachen

Frank Heitmann

heitmann@informatik.uni-hamburg.de

13. April 2015

## Alphabet und Wörter - Zusammengefasst

Die wichtigsten Dinge zu Alphabeten und Wörtern:

- $\Sigma$  für eine Menge von Symbolen (ein Alphabet), z.B.  $\Sigma = \{a, b, c\}$  oder  $\Sigma = \{0, 1\}$ .
- $\lambda$  oder  $\epsilon$  für das leere Wort.
- Das leere Wort ist *nie in einem Alphabet  $\Sigma$ !*
- Konkatenation:
  - Von Symbolen oder Worten:  $a \cdot b = ab$ ,  $ab \cdot cd = abcd$
  - Von Wortmengen:  $\{a, ac\} \cdot \{\lambda, c\} = \{a, ac, acc\}$
- $R^2, R^3$  oder auch  $\Sigma^2, \Sigma^3$  und  $w^2, w^3$  usw. für mehrfach Hintereinanderausführen von  $\cdot$ .
- Sonderfall:  $R^0 = \{\lambda\}$  und auch  $w^0 = \lambda$  ( $w$  ein Wort)
- $R^+$  für alle Worte, die sich durch beliebiges Aneinanderreihen von Worten aus  $R$  bilden lassen.
- $R^*$  wie  $R^+$ , aber  $\lambda$  kommt noch hinzu.

## Alphabete und Wörter - Noch zwei Notationen

Noch zwei Notationen:

### Definition/Notation

Sei  $\Sigma$  ein Alphabet,  $x \in \Sigma$  ein Symbol und  $w \in \Sigma^*$  ein Wort.

- 1 Mit  $|w|$  ist die Länge von  $w$  gemeint, also z.B.  $|001| = 3$  und  $|00111| = 5$ . Außerdem ist  $|\lambda| = 0$ .
- 2 Mit  $|w|_x$  ist die Anzahl der  $x$  in  $w$  gemeint. Z.B. ist mit  $\Sigma = \{0, 1, 2\}$

$$|2202|_2 = 3, \quad |2202|_1 = 0 \quad \text{und} \quad |2202|_0 = 1.$$

## Alphabete, Wörter und Sprachen

### Definition

- 1 Betrachten wir  $\Sigma^*$  für ein Alphabet  $\Sigma$ , so ist  $\Sigma^*$  die **Menge aller endlichen Wörter** (über dem Alphabet  $\Sigma$ ).
- 2 Jede (Teil-)Menge  $L \subseteq \Sigma^*$  heißt **formale Sprache**.

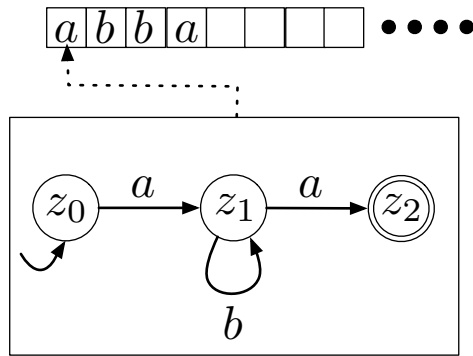
### Das Ziel

Ziel ist es nun einen Automaten zu haben, der bestimmte Wörter akzeptiert (und alle anderen nicht). Ein Automat akzeptiert damit eine formale Sprache!

### Das Ziel 2

Normalerweise ist eine Sprache  $M$  gegeben und ein Automat  $A$  wird dazu *konstruiert*. Zu zeigen ist dann stets  $L(A) = M$ .

# Endliche Automaten - Beispiel



# Der deterministische, endliche Automat

## Definition (DFA)

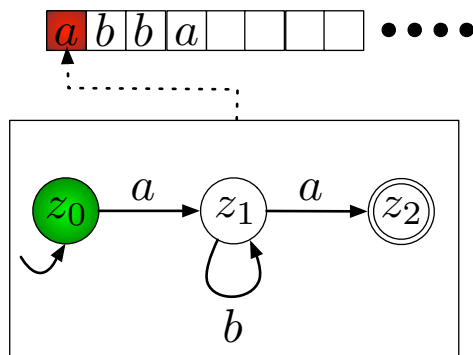
Ein **deterministischer, endlicher Automat** (DFA) ist ein 5-Tupel

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

mit:

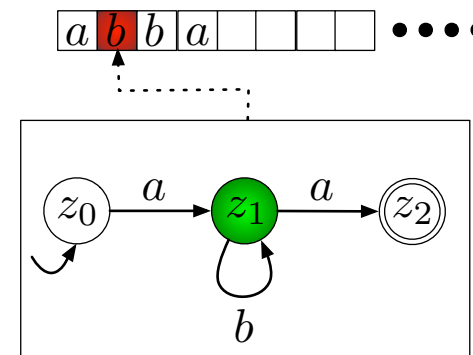
- Der endlichen Menge von *Zuständen*  $Z$ .
- Dem endlichen Alphabet  $\Sigma$  von *Eingabesymbolen*.
- Der *Überföhrungsfunktion*  $\delta : Z \times \Sigma \rightarrow Z$ .
- Dem *Startzustand*  $z_0 \in Z$ .
- Der Menge der *Endzustände*  $Z_{end} \subseteq Z$ .

# Ein Beispiel



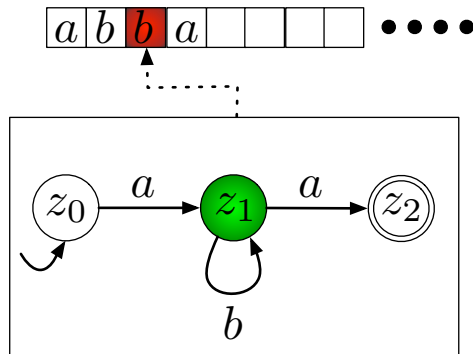
$$(z_0, abba) \text{ oder } \hat{\delta}(z_0, abba)$$

# Ein Beispiel



$$(z_0, abba) \vdash (z_1, bba) \text{ oder } \hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba)$$

## Ein Beispiel

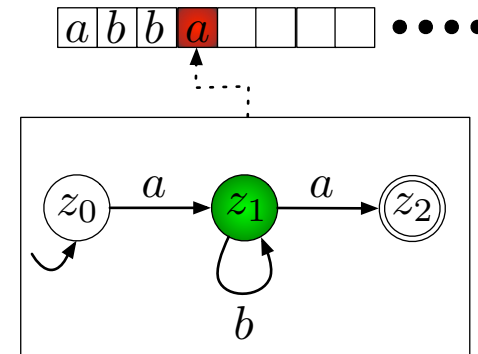


$$(z_0, abba) \vdash (z_1, bba) \vdash (z_1, ba)$$

oder

$$\hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba) = \hat{\delta}(z_1, ba)$$

## Ein Beispiel

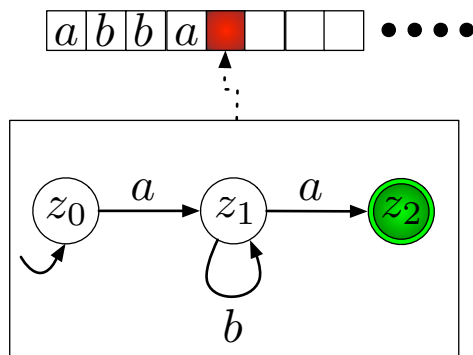


$$(z_0, abba) \vdash (z_1, bba) \vdash (z_1, ba) \vdash (z_1, a)$$

oder

$$\hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba) = \hat{\delta}(z_1, ba) = \hat{\delta}(z_1, a)$$

## Ein Beispiel



$$(z_0, abba) \vdash (z_1, bba) \vdash (z_1, ba) \vdash (z_1, a) \vdash (z_2, \lambda)$$

oder

$$\hat{\delta}(z_0, abba) = \hat{\delta}(z_1, bba) = \hat{\delta}(z_1, ba) = \hat{\delta}(z_1, a) = z_2$$

## Akzeptierte Sprache

### Definition (Akzeptierte Sprache)

Die von einem DFA  $A$  **akzeptierte Sprache** ist die Menge

$$L(A) := \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in Z_{end}\}$$

$$= \{w \in \Sigma^* \mid (z_0, w) \vdash^* (z_e, \lambda), z_e \in Z_{end}\}$$

Diese Menge wird auch als **reguläre Menge** bezeichnet. Die **Familie aller regulären Mengen** wird mit  $REG$  bezeichnet.

### Wichtige Anmerkung

$A$  akzeptiert ein Wort  $w$  genau dann, wenn  $w$  bis zum Ende gelesen werden kann **und** er dann in einem Endzustand ist.

## Eine Technik

## Wichtiges Vorgehen

Ermittelt man für einen DFA  $A$  seine akzeptierte Sprache  $M$  bzw. konstruiert man zu einer Sprache  $M$  einen DFA  $A$ , so ist  $L(A) = M$  zunächst eine Behauptung, die zu zeigen ist!

## Wichtiges Vorgehen

Hierzu sind dann **zwei Richtungen** zu zeigen:

- $L(A) \subseteq M$ . Jedes Wort, das der Automat akzeptiert ist tatsächlich in  $M$ . Bei der Argumentation geht man von einem Wort  $w$  aus, das der Automat akzeptiert ( $w \in L(A)$ ) und zeigt, dass dann auch  $w \in M$  gilt.
- $M \subseteq L(A)$ . Jedes Wort aus  $M$  wird auch von dem Automaten akzeptiert. Man geht von einem (beliebigen!) Wort aus  $M$  aus ("Sei  $w \in M$ , dann ...") und zeigt, dass dieses auch von  $A$  akzeptiert wird, also in  $L(A)$  ist.

## Zusammenfassung - Begriffe

Begriffe bisher (endliche Automaten):

- DFA
- Zustände, Startzustand, Endzustände
- Überföhrungsfunktion  $\delta : Z \times \Sigma \rightarrow Z$
- erweitere Überföhrungsfunktion  $\hat{\delta} : Z \times \Sigma^* \rightarrow Z$
- Konfiguration, Konfigurationsübergang
- Rechnung, Erfolgsrechnung
- akzeptierte Sprache
- reguläre Menge, REG
- vollständig, initial zusammenhängend

## Wichtige Anmerkung

Alle diese Begriffe sind wichtig (auf dieser Grundlage bauen wir auf). Daher die fleissig lernen!

## Noch zwei Begriffe

## Definition

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

- 1 **vollständig**, wenn zu jedem  $(z, x) \in Z \times \Sigma$  ein  $z'$  existiert mit  $\delta(z, x) = z'$ .
- 2 **initial zusammenhängend**, wenn es zu jedem  $z \in Z$  ein Wort  $w$  gibt mit  $\hat{\delta}(z_0, w) = z$ .

## Hinweis

**vollständig** = jede Eingabe kann gelesen werden

**initial zusammenhängend** = jeder Zustand erreichbar

## Fragen...

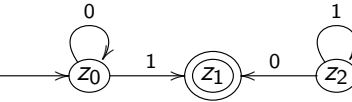
Sei  $R = \{a, b\}$  und  $S = R^* \cdot \{a\} \cdot R^*$ . Was ist  $R^* \setminus S$ ?

- 1  $R^*$
- 2  $\emptyset$
- 3  $\{b\}^*$
- 4  $\{w \in \{a, b\}^* \mid |w|_a \geq 2\} \cup \{\lambda\}$

## Fragen...

Sei  $A$  ein Automat und  $M$  eine Wortmenge. Wenn man  $L(A) \subseteq M$  zeigt, dann ...

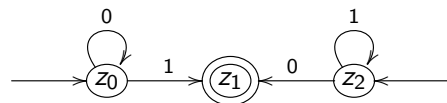
- 1 ist man fertig, wenn man  $M$  nicht zu groß gewählt hat.
- 2 weiß man, dass jedes Wort, das in  $M$  ist, vom Automaten akzeptiert wird.
- 3 weiß man, dass der Automat teilweise stimmt.
- 4 ...



Ist dieser DFA initial zusammenhängend?

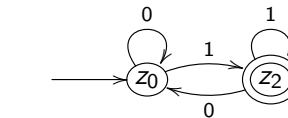
- 1 Ja!
- 2 Nein!

## Fragen...



Ist dieser DFA initial zusammenhängend?

- 1 Ja!
- 2 Nein!



Welche Sprache akzeptiert dieser DFA  $A$ ?

- 1  $L(A) = \{0, 1\}^*$
- 2  $L(A) = \{0\}^* \cdot \{1\} \cdot \{1\}^*$
- 3  $L(A) = \{0, 1\}^* \cdot \{1\}$
- 4  $L(A) = (\{0\}^* \cdot \{1\} \cdot \{1\}^* \cdot \{0\})^*$

## Zur Nachbereitung

### Zur Nachbereitung

Richtige Antworten sind:

- ① 3
- ② 4 (jedes Wort, das vom Automaten akzeptiert wird, ist in  $M$ )
- ③ 2
- ④ 2
- ⑤ 3

## Konstruktion von DFAs 1

### Technik

#### Techniken zum Konstruieren von DFAs

Methode 1: Was will ich speichern?  
(Und wie sehen dann die Zustände aus?)

## Die Aufgabe...

Es ist nun also üblicherweise

- eine **Sprache gegeben** (bzw. wir wissen, was wir brauchen; z.B. wollen wir die Wörter erkennen, die auf 0 enden)

Wir wollen dazu

- einen **Automaten konstruieren**

Wie kann man das machen?

Es gibt keine Vorschrift/kein Rezept dafür - aber Techniken.

## Konstruktion von DFAs 1 - Beispiel

### Beispiel

$$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$$

Wie konstruieren wir hierfür einen DFA?

Tipp: Was wollen wir uns merken?

### Definition/Notation (zur Wiederholung)

Sei  $\Sigma$  ein Alphabet,  $x \in \Sigma$  ein Symbol und  $w \in \Sigma^*$  ein Wort. Mit  $|w|_x$  ist dann die Anzahl der  $x$  in  $w$  gemeint. Z.B. ist mit  $\Sigma = \{0, 1, 2\}$

$$|2202|_2 = 3, \quad |2202|_1 = 0 \quad \text{und} \quad |2202|_0 = 1.$$

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$

### Die Idee

Wir merken uns im Zustand, ob die Anzahl der bisher gelesenen 0en gerade (g) oder ungerade (u) ist. Ebenso für die 1en. Das sind **zwei** Informationen, die **jeweils genau einen von zwei Werten annehmen** können (führt zu vier Zuständen) und die **wir leicht mit jedem gelesenen Symbol aktualisieren können**.

### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

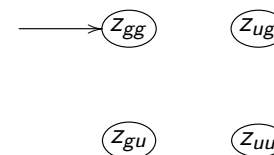
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

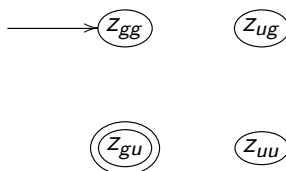
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

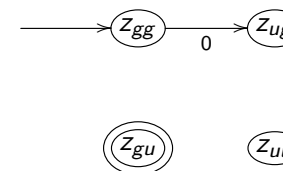
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

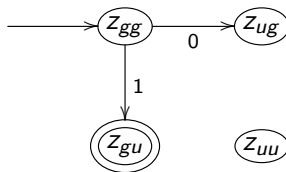
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

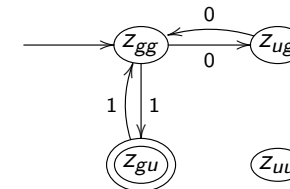
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

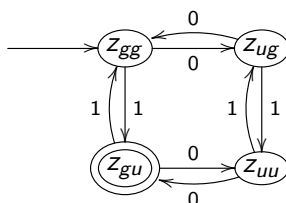
$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Konstruktion von DFAs 1 - Beispiel

### Das Problem

$$M := \{w \in \{0,1\}^* \mid |w|_0 \text{ ist gerade und } |w|_1 \text{ ist ungerade}\}$$



### Die Durchführung

Wir notieren  $gg, gu, ug, uu$  für die Zustände.

$gu \hat{=}$  gerade Anzahl 0en, ungerade Anzahl 1en

$ug \hat{=}$  ungerade Anzahl 0en, gerade Anzahl 1en ...

## Methode 1 - Anmerkungen

### Bemerkung

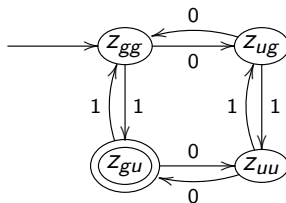
Bisweilen (nicht hier, aber manchmal) wird mit dieser Methode ein nicht erreichbarer Teil mitkonstruiert. Dieser kann dann weggelassen werden. Es genügt dann sich auf die initiale Zusammenhangskomponente einzuschränken.

### Wichtige Anmerkung

Egal, wie man den Automaten konstruiert, danach ist stets ein Beweis von  $L(A) = M$  nötig! Ist die Konstruktion "gut" (d.h. liegt ihr eine eingängige Idee zugrunde), so ist der Beweis aber oft sehr viel einfacher.

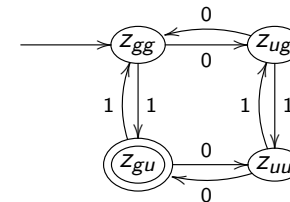


## Beweis der Korrektheit



- Sei  $w \in M$ , dann kann  $w$  nach Konstruktion (der Automat ist vollständig) zu Ende gelesen werden. Nach Konstruktion der Zustände und der Zustandsübergänge enden wir dann in  $z_{gu}$  (da  $w$  eine gerade Anzahl von 0en und eine ungerade Anzahl von 1en enthält) und akzeptieren. Also ist auch  $w \in L(A)$ .

## Beweis der Korrektheit



- Sei  $w \in L(A)$ . Da  $w$  akzeptiert wird, muss  $A$  in  $z_{gu}$  enden. Dies ist aber nach Konstruktion gleichbedeutend damit, dass  $w$  gerade viele 0en und ungerade viele 1en enthält. Damit gilt auch  $w \in M$ .

## Beweis der Korrektheit

### Anmerkung

Man könnte noch die Zustände und Zustandsübergänge durchgehen und argumentieren, dass sie wirklich das gewünschte leisten (z.B. das der Wechsel von  $z_{gg}$  nach  $z_{gu}$  eine 1 erfordert und das dann ja wirklich ungerade viele 1en gelesen wurden, wenn vorher gerade viele gelesen wurden). Bei kleineren Beispielen ist das aber meist klar. **Es ist dann aber nötig vorher gut die Konstruktion zu erklären! Und aus dieser Konstruktion müssen sich Dinge ableiten lassen, die gebraucht werden, sonst macht die Formulierung "nach Konstruktion" keinen Sinn!**

## Konstruktion von DFAs 2

### Technik

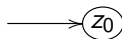
**Techniken zum Konstruieren von DFAs**  
Methode 2: On-the-fly-Konstruktion ...

## Konstruktion von DFAs 2 - Beispiel

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

Wir starten mit dem Startzustand:



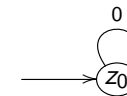
Was passiert bei einer 0 ?

## Konstruktion von DFAs 2 - Beispiel

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

Bei 0: Keine neuen Informationen. Wir bleiben in  $z_0$ :



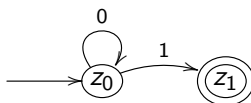
Was passiert bei einer 1 ?

## Konstruktion von DFAs 2 - Beispiel

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

Bei 1: Vielleicht endet das Wort hier. Dann sind wir fertig! Also brauchen wir einen neuen Zustand, in dem wir akzeptieren!



Mit  $z_0$  sind wir fertig (keine weiteren Symbole in  $\Sigma$ ).

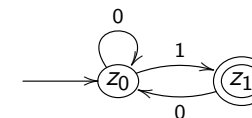
Aber wir haben einen neuen Zustand! Was passiert hier bei 0?

## Konstruktion von DFAs 2 - Beispiel

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

Bei 0: Wir wollen nicht mehr akzeptieren und könnten einen neuen Zustand einführen. **Aber** wir haben wieder genau so viele Infos wie in  $z_0$ , also gehen wir einfach dahin!



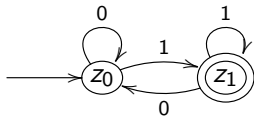
Und was passiert bei 1?

## Konstruktion von DFAs 2 - Beispiel

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

Bei 1 wollen wir weiterhin akzeptieren und bleiben einfach in  $z_1$ .



Und sind fertig, da wir in allen Zuständen alle Symbole lesen können!

## Methode 2 - Anmerkungen

### Bemerkung

Da man sich für jeden Zustand zu jeder Eingabe überlegt, was passiert, wird der konstruierte Automat i.A. vollständig und initial zusammenhängend sein.

### Wichtige Anmerkung

Kritisch bei dieser Konstruktionsmethode ist irgendwann zu merken, wann bereits vorhandene Zustände benutzt werden können – und im besten Fall auch, wofür sie stehen; bspw. kann man oben  $z_x$  mit "letztes gelesenes Symbol ist  $x$ " identifizieren ( $x \in \{0, 1\}$ ).

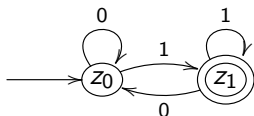
### Wichtige Anmerkung

Nochmal: Egal, wie man den Automaten konstruiert, danach ist stets ein Beweis von  $L(A) = M$  nötig! Ist die Konstruktion "gut" (d.h. liegt ihr eine eingängige Idee zugrunde), so ist der Beweis aber oft sehr viel einfacher.

## Beweis der Korrektheit

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$

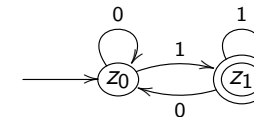


- Sei  $w = v1 \in M$ . Da der Automat vollständig ist, kann  $w$  auf jeden Fall ganz gelesen werden. Da  $w$  auf 1 endet, muss der Automat mit dieser letzten 1 nach Konstruktion in den Zustand  $z_1$  wechseln (da jeder Zustand eine 1-Kante zu  $z_1$  hat). Mit  $z_1 \in Z_{end}$  folgt  $w \in L(A)$ .

## Beweis der Korrektheit

### Beispiel

$$M := \{0, 1\}^* \{1\} = \{w \in \Sigma^* \mid \exists v \in \Sigma^* : w = v1\}$$



- Sei  $w \in L(A)$ . Dann muss nach Konstruktion das Wort auf 1 enden, da  $z_1$  der einzige Endzustand ist und alle Kanten nach  $z_1$  mit 1 beschriftet sind. Dann ist aber sofort auch  $w \in M$ .

## Intermezzo

## Intermezzo ...

## Wortsuche

Gegeben ein Text  $T$  der Länge  $n$  und ein Muster (ein gesuchtes Wort)  $w$  der Länge  $m$ . Wie findet ihr  $w$  in  $T$ ?

## Naive Wortsuche vs. Wortsuche mit Automaten

## Die naive Wortsuche

- dauert  $O(n \cdot m)$  Schritte.
- Verbesserungen sind möglich (z.B. mit booleschen Flags welche Buchstaben man schon entdeckt hat), aber dies ist fehleranfällig und im Grunde baut man Automaten nach.

## Die Wortsuche mit Automaten

- braucht nur  $O(n)$  Schritte!
- Vorher muss der Automat aus dem Pattern erzeugt werden.
- Zur Info: Dies kann man in  $O(m \cdot |\Sigma|)$  machen.
- Wenig fehleranfällig. Methoden für den Automaten müssen nur einmal geschrieben werden!

## Deswegen ...

## Vorgehen:

- Welche Sprache wird gebraucht? (Z.B. Ziffernfolgen erkennen oder Schlüsselwörter oder Wortsuche in Texten)
- Automat bauen
- $L(A) = M$  zeigen
- Automat in Code oder Tabelle gießen
- Für schnelle Worterkennung nutzen

## Deswegen...

Deswegen später auch die ganzen Konstruktionen. Man könnte jetzt/dann auch ein Tool schreiben, in dem

- jemand ein gesuchtes Wort oder einen regulären Ausdruck eingibt
- dieser wird in einen Automaten umgewandelt (automatisch)
- bzw. in Code (so dass wir eine Methode für  $\delta$  haben)
- der dann für eine effiziente Wortsuche benutzt werden kann

Anwendungen z.B.:

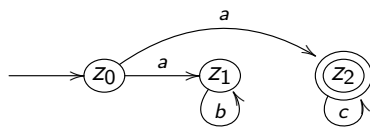
- Suchen in Texten
- Suchen nach Schlüsselwörtern beim Kompilieren
- Suchen in DNA-Sequenzen
- ...

## Auf zu neuen Ufern...

Und jetzt was neues...

## 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...



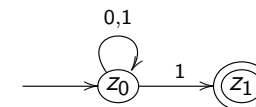
Nach Lesen von  $a$  ist der Automat *nichtdeterministisch* sowohl in  $z_1$  als auch in  $z_2$ .

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

## Sinn?

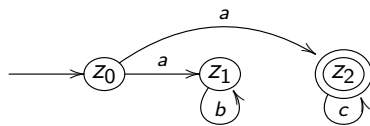
ABER WARUM?!?

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



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

## Die Überföhrungsfunktion



Wie würdet ihr die Überföhrungsfunktion definieren?

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

## 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  $\Sigma$  von *Eingabesymbolen*.
- Der *Überföhrungsfunktion*  $\delta : Z \times \Sigma \rightarrow 2^Z$ .
- 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:

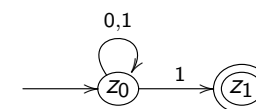
- Der endlichen Menge von *Zuständen*  $Z$ .
- Dem endlichen Alphabet  $\Sigma$  von *Eingabesymbolen*.
- Der **Zustandsübergangsrelation**  $K \subseteq Z \times \Sigma \times Z$
- Der Menge der *Startzustände*  $Z_{start} \subseteq Z$ .
- Der Menge der *Endzustände*  $Z_{end} \subseteq Z$ .

## Überföhrungsfunktion und Rechnung

### Definition (Erweiterte Überföhrungsfunktion)

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

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



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

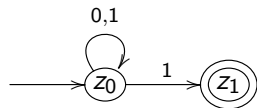
## Überföhrungsfunktion und Rechnung

### Anmerkung

Man kann die erweiterte Überföhrungsfunktion 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.



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

## Überföhrungsfunktion 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.

## Überföhrungsfunktion 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  $(z', \lambda)$  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  $(z_e, \lambda)$  mit  $z_e \in Z_{end}$  endet, dann wird  $w$  akzeptiert.

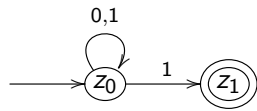
## Akzeptierte Sprache

### Definition (Akzeptierte Sprache)

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

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

## Ein Beispiel

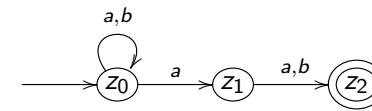


Alle möglichen Rechnungen auf dem Wort 011:

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

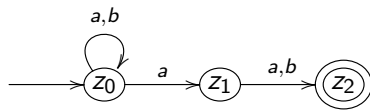
## Fragen...



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

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

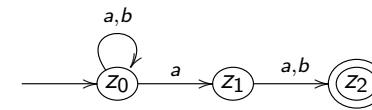
## Fragen...



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...

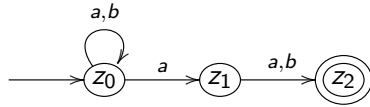


Gibt es eine Rechnung auf  $aaa$ , bei der  $A$  blockiert?

- 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  $\{a, b\} \{a\} \{a, b\}$

## 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 ( $\delta_N$  ist die Überföhrungsfunktion des NFA,  $\delta_D$  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 ??

## Zur Nachbereitung

### Zur Nachbereitung

Richtige Antworten sind:

- 1 5
- 2 4
- 3 1
- 4 3

## 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.

Hier machen wir morgen weiter ...

## Zusammenfassung

Wir haben heute:

- Zwei **Konstruktionsmethoden für DFAs** kennengelernt.
- Gesehen, wo DFAs (u.a.) sinnvoll benutzt werden können.
- **NFAs** kennengelernt.