

# Automatentheorie

endliche deterministische Automaten

Prof. Dr. Franz-Karl Schmatzer  
[schmatzf@dhbw-loerrach.de](mailto:schmatzf@dhbw-loerrach.de)

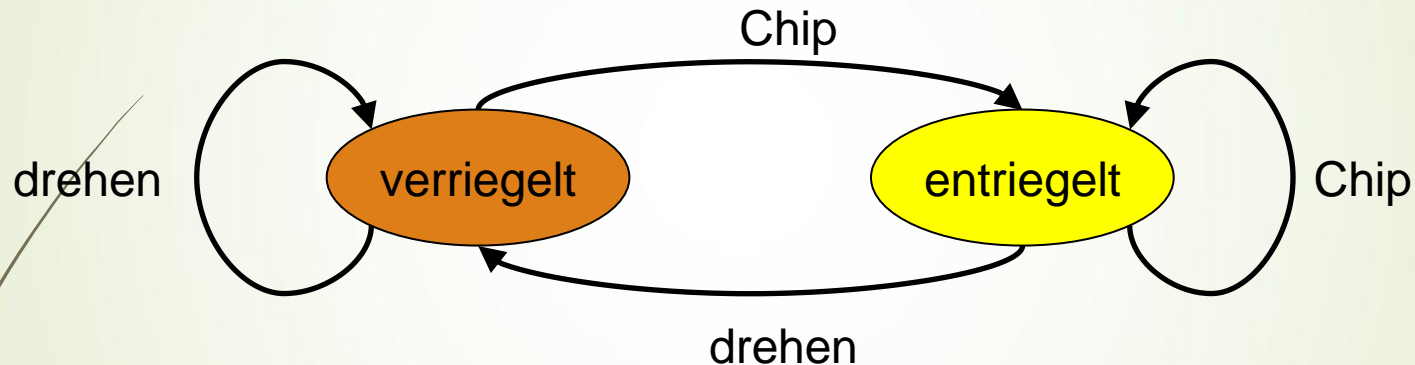
- C.Wagenknecht, M.Hielscher; Formale Sprachen, abstrakte Automaten und Compiler; 3.Aufl. Springer Vieweg 2022;
- A.V.Aho, M.S.Lam,R.Savi,J.D.Ullman, *Compiler – Prinzipien,Techniken und Werkzeuge*. 2. Aufl., Pearson Studium, 2008.
- Güting, Erwin; *Übersetzerbau –Techniken, Werkzeuge, Anwendungen*, Springer Verlag 1999
- Sipser M.; Introduction to the Theory of Computation; 2.Aufl.; Thomson Course Technology 2006
- Hopcroft, T. et al; Introduction to Automata Theory, Language, and Computation; 3. Aufl. Pearson Verlag 2006

# Agenda

- Allgemeine Einführung
- Definition eines Automaten
- Typen von deterministischen Automaten
- Aufbau von Automaten
- Modellierung von Automaten
- Sprachen eines Automaten

# Einführung

- Viele Systeme des täglichen Lebens lassen sich als Automaten auffassen.
- Betrachte dazu eine Schwimmbaddrehkreuz als Zugangskontrolle

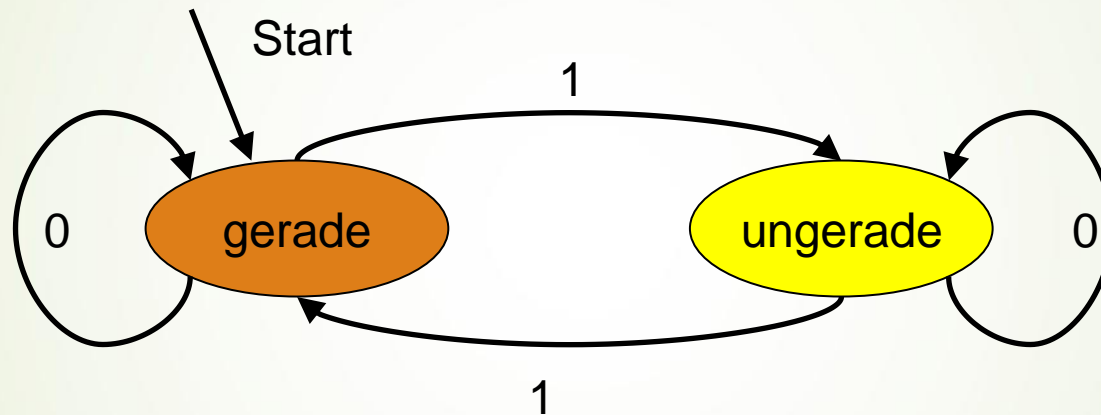


- Das Verhalten des System hängt offenbar von der Aktion und von dem Zustand des Systems ab.
- Andere Beispiele sind z.B.
  - Getränke- und Zigarettensautomaten
  - Scanner-Systeme, Paritätsprüfung

# Einführung

## Paritätsprüfung

- Beispiel für eine Paritätsprüfung



- Man kann endliche Automaten auch als die Abstraktion eines einfachen Computer auffassen, der endliche viele Zustände einnehmen kann und von einem Eingabeband liest.

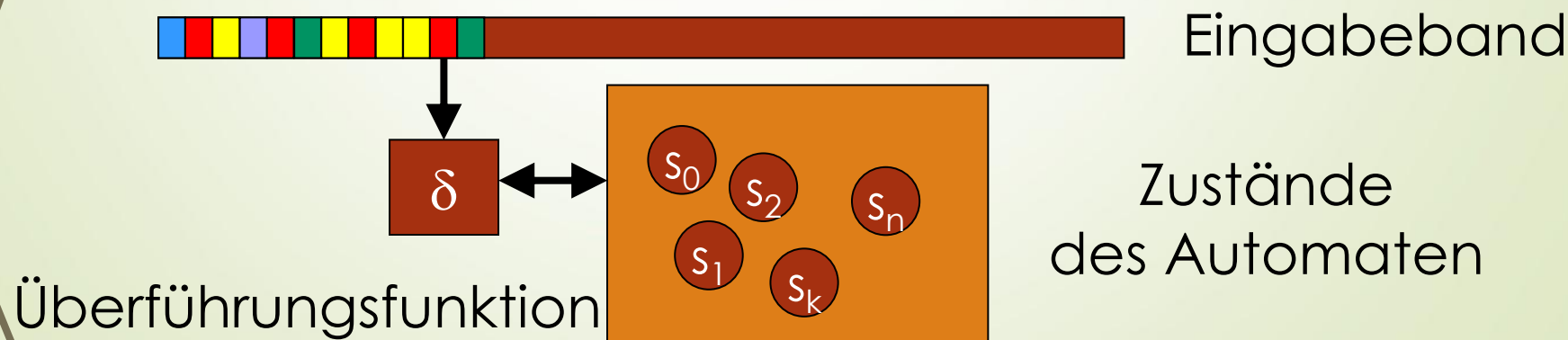
# Aufgaben

- Wo sind Ihnen endliche Automaten noch begegnet?

# Deterministischer endlicher Automat (DEA)

## Abstraktion

- Allgemeines Modell einer Maschine
  - Ein Einleseband mit Eingabezeichen,
  - eine Maschine, die endliche viele interne Zustände haben kann
  - eine Funktion, die abhängig von dem gelesenen Eingabezeichen und des momentanen Zustandes der Maschine, die Zustände der Maschine ändern kann.
- Weiter muss man noch festlegen:
  - die Startkonfiguration der Maschine
  - und die Endkonfigurationen der Maschine



# Deterministischer endlicher Automat (DEA)

## Formale Definition

➤ Sei  $A = (Q, \Sigma, \delta, q_0, F)$  ein DEA

$Q = \{q_1, \dots, q_n\}$  eine nicht leere Menge von Zuständen

$\Sigma = \{e_1, \dots, e_n\}$  eine nicht leere Menge von Zeichen, das Eingabealphabet

$\delta : Q \times \Sigma \rightarrow Q$  eine Funktion, die Überföhrungsfunktion

$q_0 \in Q$  der Anfangszustand

$F \subseteq Q$  die nicht leere Menge von Endzustände



# DEA

## Überföhrungsfunktion und Zustandsdiagramm

### ➤ Zustandsdiagramm

➤ Startzustand

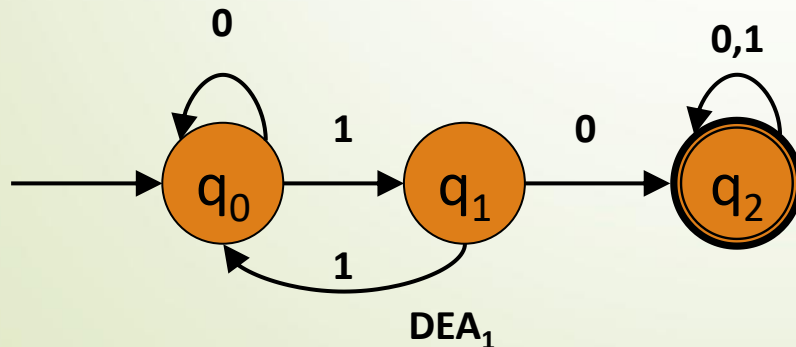
➤ Endzustände

➤ Übergänge, als Pfeile zwischen den Zuständen

### ➤ Überföhrungsfunktion

➤ Tabelle, die für jedes Eingabezeichen den Folgezustand spezifiziert

➤ Beispiel:  $DEA = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$



Überföhrungsfunktion  $\delta$  für  $DEA_1$

$\delta$	0	1
$q_0$	$q_0$	$q_1$
$q_1$	$q_2$	$q_0$
$q_2$	$q_2$	$q_2$

# Tools zu den Automaten

- Nutzen Sie die Tools der Webseite AtoCC
  - Es erlaubt Automaten und Grammatiken aufzubauen und zu simulieren
  - Zugang <https://atocc.de/cgi-bin/atocc/site.cgi?lang=de&site=main>

**AtoCC**  
"from automaton to compiler construction"

[AtoCC](#) [Papers](#) [Download](#) [Tutorials](#) [Workshops](#) [Aufgaben](#) [Impressum](#)

**AtoCC herunterladen >>>**

**AtoCC Buch >>>**

**AutoEdit Aufgabenheft >>>**

**FLACI**

Wir haben AtoCC konsequent weitergedacht und einen deutlich **überarbeiteten Nachfolger** in Form einer modernen Web-Applikation entwickelt. Wir empfehlen allen Nutzenden von AtoCC einen Blick auf [FLACI.com](https://FLACI.com) zu werfen.

AtoCC wird weiterhin auf dieser Seite angeboten werden, jedoch sind keine Weiterentwicklungen mehr geplant.

**Was ist AtoCC?**

Die Lernumgebung AtoCC unterstützt den Lernenden in der theoretischen Informatik (Automatentheorie, formale Sprachen) und deren Anwendung im Compilerbau. AtoCC befördert Aktivitäten, mit deren Hilfe beim Lehrenden ganz bestimmte geistige Techniken entwickelt werden.

AtoCC besteht aus 6 Komponenten: AutoEdit, AutoEdit Workbook, kfG Edit, TDiag, VCC und SchemeEdit. Weitere Informationen zur Architektur von AtoCC finden sich unter "Papers".

Wir freuen uns über Feedback über den Einsatz und Erfolg des Projektes, wie auch über Bug-Reports und Verbesserungsvorschläge (hierzu können Sie am schnellsten das passende Formular unter "Kontakt" verwenden). Viel Erfolg mit AtoCC wünschen,

# Toolbox FLACI I

(Formale Sprachen, abstrakte Automaten, Compiler und Interpreter)

- Zugang <https://flaci.com/home/>
- Das Tool „Abstrakte Automaten“ erlaubt die Automaten aufzubauen und zu simulieren.
- Rufen Sie das Tool „Abstrakte Automaten“ auf und melden Sie sich an dem System an.

## Formale Sprachen

Ein Alphabet  $A$  ist eine endliche, nichtleere Menge von Zeichen.

Wählen Sie eines der Beispiel-Alphabete zum Experimentieren aus.

- ☐  $A_1 = \{0, 1\}$
- ☐  $A_2 = \{a, b, c, \dots, z\}$
- ☐  $A_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- ☒  $A_4 = \{ \text{Bild, Musik, Video, Text, Star} \}$
- ☐  $A_5 = \{ \text{begin, end, for, while, do, repeat, until} \}$

Interaktives Minututorial zu den Grundbegriffen formaler Sprachen

## Reguläre Ausdrücke

Der einfachste reguläre Ausdruck  $a$  steht für ein einzelnes Wort "a", kurz:  $a$ . Es besteht aus genau einem Alphabetzeichen  $a$ . Der Ausdruck beschreibt die Sprache  $L = \{a\}$ . Reguläre Ausdrücke lassen sich verketteten:  $a$  gefolgt von  $b$ , gefolgt von  $c$  wird kurz  $abc$  geschrieben.

Wie ändert sich die beschriebene Sprache, wenn man den regulären Ausdruck  $a$  zu  $ab$  oder  $abc$  verändert?

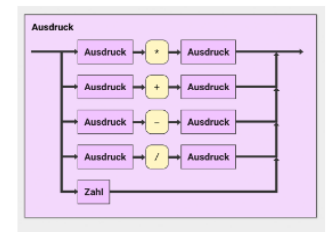
Regulärer Ausdruck:  $a$

Ergebnis:  $ab$

Syntax Diagramm:  $a \rightarrow ab$

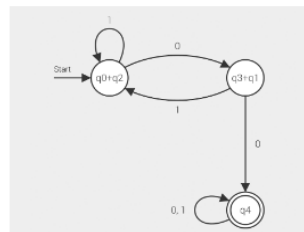
Interaktives Minututorial zu regulären Ausdrücken

## Formale Grammatiken



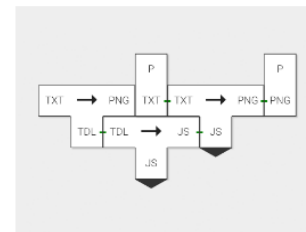
Kontextfreie Grammatiken entwickeln, transformieren und konvertieren

## Abstrakte Automaten



Abstrakte Automaten konstruieren, simulieren, transformieren und konvertieren

## Compiler und Interpreter



Modellieren von Übersetzungsprozessen und Entwicklung von Compilern und Interpretern

# Toolbox FLACI II

## Einführendes Modell

- Drücken Sie den + Button unten rechts.
- Füllen Sie das Popup-Window wie folgt aus.

**Name:** Beispiel 1

**Beschreibung:** Einführendes Modell

**Typ:** DEA

### Neuen Automaten erstellen:

Name

Beispiel 1

Beschreibung

Einführendes Modell

Typ

- ☒ **DEA:** deterministischer endlicher Automat
- ☐ **NEA:** nichtdeterministischer endlicher Automat
- ☐ **MEALY:** Mealy-Maschine
- ☐ **MOORE:** Moore-Maschine
- ☐ **DKA:** deterministischer Kellerautomat
- ☐ **NKA:** nichtdeterministischer Kellerautomat
- ☐ **TM:** deterministische Turingmaschine

ABBRECHEN

SPEICHERN

# Toolbox FLACI III

## Einführendes Modell

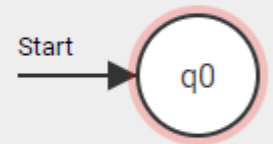
- Nach der Bestätigung erscheint der Startzustand q0
- Wählen Sie das 5 Alphabet und geben Sie die Zustände 0 und 1 ein.
- Nun erstellen Sie einen weiteren Zustand und einen Übergang mit dem Zeichen 0.
- Kennzeichnen Sie den neuen Zustand als Endzustand.
- Fügen Sie ein Übergang mit dem Zeichen 1 vom Endzustand zu dem Startzustand ein.
- Drücken Sie den Reiter „überprüfen“.

### Eingabealphabet:

- ☐  $\Sigma_1 = \{ a, b, c \}$
- ☐  $\Sigma_2 = \{ a, b, c, \dots, z \}$
- ☐  $\Sigma_3 = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- ☐  $\Sigma_4 = \{ \heartsuit, \spadesuit, \clubsuit, \diamondsuit, \times, \div \}$
- ☒  $\Sigma_5 = \{ 0, 1 \}$

Alphabetszeichen (mit Komma getrennt eingeben)

0,1



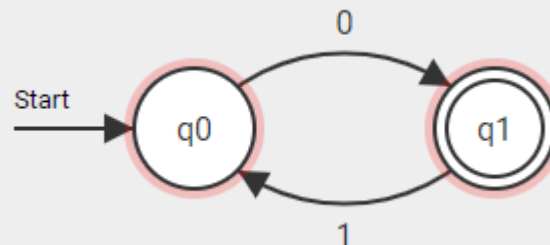
Name

q1

☒ Endzustand

☐ Startzustand

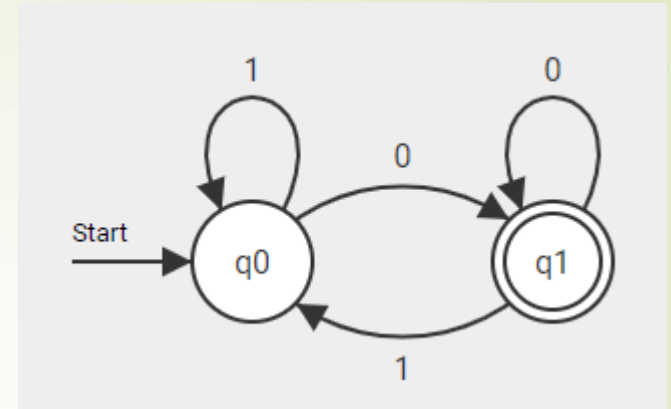
ZUSTAND ENTFERNEN



# Toolbox FLACI IV

## Einführendes Modell

- Der Automat ist nicht vollständig.
- Fügen Sie noch die fehlenden Übergänge bei q0 und q1 ein.
- Drücken Sie „überprüfen“
- Nun Drücken Sie auf „Simulation“ und geben das Wort „01011“
- Und drücken Sie auf „Simulation starten“.
- Exportieren Sie die Datei.



Eingabewort  
0 1 0 1 1 +

Simulation starten

Langsam Schnell

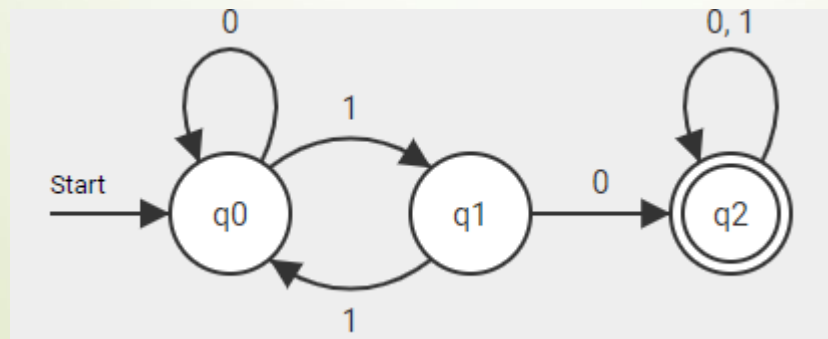
? Letzte Eingabewörter:  
x 0 1 0 1 1

Konfigurationsfolge(n) für: 0 1 0 1 1

K1	0	1	2	3	4	5
▶	q0	q1	q0	q1	q0	q0
	01011	1011	011	11	1	

# Aufgabe1 FLACI

- Erstellen Sie den unten angegebenen Automaten
  - In FLACI drücken sie auf den „+“-Button unten rechts.
  - Geben Sie einen Namen „Beispiel 2“ und kurze Beschreibung an.
  - Definieren Sie den Automatentyp (DEA)
  - Drücken Sie okay.
  - Geben Sie das Alphabet ein  $\{0,1\}$  und Erstellen Sie den Automaten
    - Legen Sie zunächst die 3 Zustände wie unten angegeben an.
    - Dann fügen Sie die Übergänge hinzu. Wenn Sie auf die Übergänge klicken können Sie die Zeichen auch nachträglich eingeben.
    - Checken Sie den Automaten. Er sollte korrekt sein.



# Aufgabe 1 FLACI

## Übergangstabelle

- Lassen Sie sich die Übergangstabelle anzeigen

$\delta$	0	1
q0	q0	q1
q1	q2	q0
q2	q2	q2

- Führen Sie ein Simulation mit verschiedene Worten durch.
  - Welche Worte landen am Schluss im Endzustand?

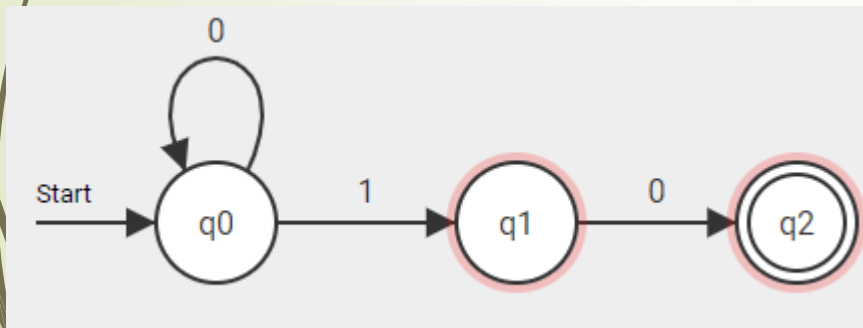


# DEA

## vollständiger/nicht vollständiger Automat

- Man nennt einen Automaten
  - vollständig, wenn die Überföhrungsfunktion  $\delta$  für jedes Eingabezeichen einen Folgezustand spezifiziert (totale Funktion).
  - nicht vollständig, wenn die Überföhrungsfunktion  $\delta$  nur für die erlaubten Übergänge einen Folgezustand spezifiziert (partielle Funktion).
- Beispiel:  $DEA_2 = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$  (Nicht vollständiger Automat)

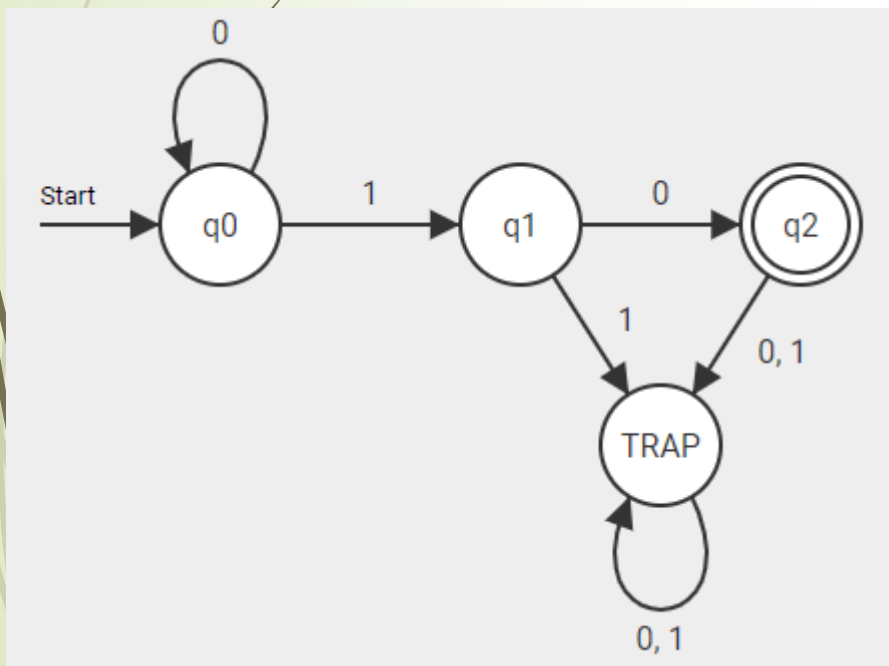
Überföhrungsfunktion  $\delta$  für  $DEA_2$



$\delta$	0	1
$q_0$	$q_0$	$q_1$
$q_1$	$q_2$	-
$q_2$	-	-

## Äquivalenz vollständiger und nicht vollständiger Automat

- Jeder nicht vollständiger Automat lässt sich in einen vollständigen Automaten umwandeln, durch Einfügen eines neuen Zustandes „TRAP“.
- Beispiel umwandeln von  $DEA_2$  in einen vollständigen Automaten  $DEA_3$
- $DEA_3 = (\{q_0, q_1, q_2, TRAP\}, \{0,1\}, \delta, q_0, \{q_2\})$



Überföhrungsfunktion  $\delta$  für  $DEA_3$

$\delta$	0	1
$q_0$	$q_0$	$q_1$
$q_1$	$q_2$	TRAP
$q_2$	TRAP	TRAP
TRAP	TRAP	TRAP

# Sprache eines DEAs

## Konfigurationsübergänge

- Eine Maschine liest Schritt für Schritt die Zeichen  $e_i \in \Sigma$  vom Band und ändert abhängig von der Überföhrungsfunktion seinen internen Zustand.
- Man charakterisiert Zustände  $k$  der Maschine durch das Tupel  $k = (q, v)$  mit  $q \in Q$  und  $v \in \Sigma^*$ . Dabei ist  $v$  das noch zu verarbeitende Wort aus  $\Sigma^*$ .
- Ein Zustandsübergang von  $k$  nach  $k'$  findet beim Einlesen eines Zeichens  $e \in \Sigma$  statt, wenn es dazu einen Übergang gibt:  
d.h  $k = (q, ew)$  geht in  $k' = (q', w)$ , wenn  
die Überföhrungsfunktion  $\delta(q, e) = q'$  existiert.
- Man schreibt formal:  $(q, ew) \rightarrow (q', w)$

# Sprache eines DEAs

## Konfigurationsübergänge

- Die Abarbeitung eines Wortes  $w = e_1 e_2 \dots e_r$  durch einen endlichen Automaten kann man daher als eine Folge von Konfigurationsübergänge ansehen.

$$(q_0, e_1 e_2 \dots e_r) \rightarrow (q_1, e_2 \dots e_r) \rightarrow \dots \rightarrow (q_r, \varepsilon)$$

$$\text{mit } \delta(q_i, e_i) = q_{i+1}$$

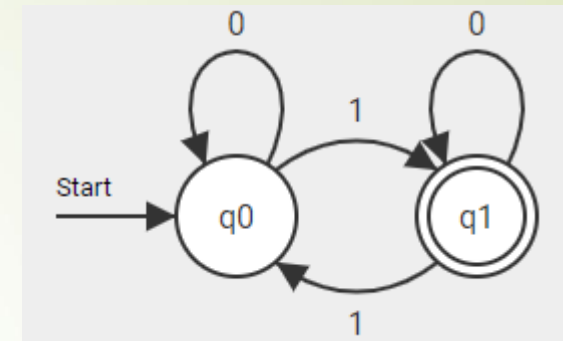
Kurze Notation dafür:

$$(q_0, e_1 e_2 \dots e_r) \rightarrow^* (q_r, \varepsilon)$$

(endliche Folge von Zustandsübergänge,  
die von  $q_0$  nach  $q_r$  führt)

# Aufgabe 2

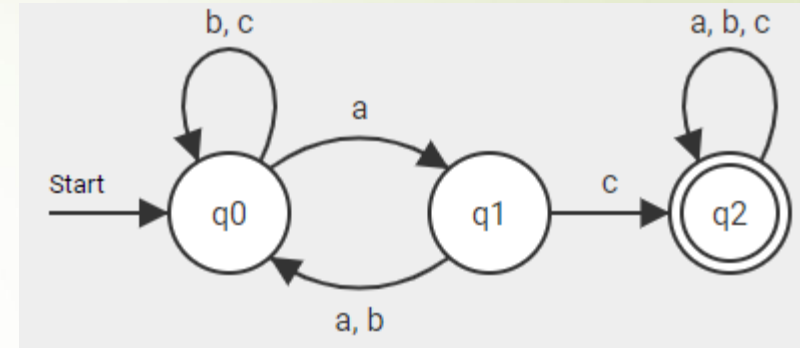
## Sprache eines Automaten I



- Ein Automat über dem Alphabet  $\{0,1\}$  ist gegeben.
- In Input geben Sie das Wort: 0110100 ein und drücken „Start Simulation“.
- Geben Sie verschiedene Worte ein. Auch Worte die der Automat nicht akzeptiert.
- Beobachte Sie das Verhalten
- Was ist das kleinste Wort, was der Automat akzeptiert?
- Was für eine Struktur haben die Worte der Sprache?

# Aufgabe 2

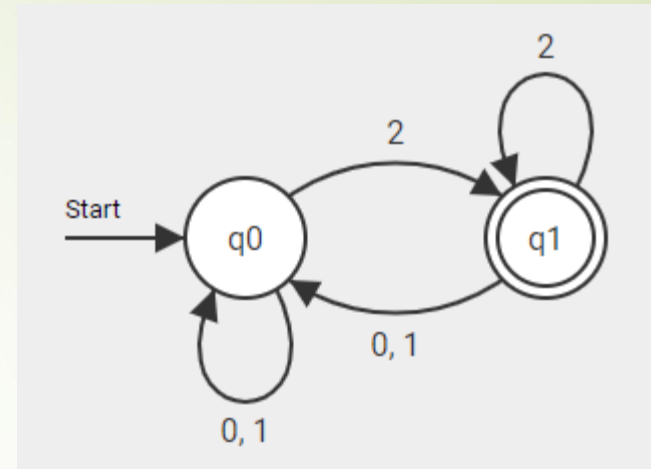
## Sprache eines Automaten II



- Ein Automat über dem Alphabet  $\{a,b,c\}$  ist gegeben
- Als Input geben Sie das Wort: bccacb ein und drücken „Start Simulation“.
- Geben Sie verschiedene Worte ein. Auch Worte die der Automat nicht akzeptiert.
- Beobachte Sie das Verhalten
- Was ist das kleinste Wort, was der Automat akzeptiert?
- Was für eine Struktur haben die Worte der Sprache?

# Aufgabe 2

## Sprache eines Automaten III



- Ein Automat über dem Alphabet  $\{0,1,2\}$  ist gegeben
- Als Input geben Sie das Wort: 0101221022 ein und drücken „Start Simulation“.
- Geben Sie verschiedene Worte ein. Auch Worte die der Automat nicht akzeptiert.
- Beobachte Sie das Verhalten.
- Was ist das kleinste Wort, was der Automat akzeptiert?
- Was für eine Struktur haben die Worte der Sprache?

# Sprache eines DEAs

## Die Sprache $L(A)$

- Wörter  $w$  werden akzeptiert, wenn ausgehend vom Startzustand  $q_0$  die Maschine nach der Bearbeitung aller Zeichen sich in einem der Endzustände  $q_F$  befindet.

$$(q_0, w) \rightarrow^* (q_F, \varepsilon)$$

- Die Menge aller Wörter  $w$ , die eine Maschine  $A = (Q, \Sigma, \delta, s_0, F)$  akzeptiert, bezeichnet man als die Sprache  $L(A)$  der Maschine:

$$L(A) = \{w \in \Sigma^* \mid (q_0, w) \rightarrow^* (q_F, \varepsilon), q_F \in F\}$$

- Eine **Sprache**  $L \subseteq \Sigma^*$  heißt **regulär**, falls es einen **endlichen Automaten** gibt, der  $L$  akzeptiert, d.h. für den  $L = L(A)$  gilt.



# Sprache eines DEAs

## Beispiel Arbeitsweise und $L(A)$

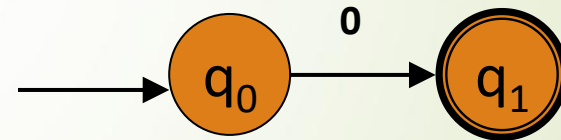
- Der Automat  $DEA_2$  akzeptiert alle Wörter  $w$  die, entweder
  - 10 oder
  - 0..010 lauten.
- $L(DEA_2) = \{10, 010, 0010, 0...10\} = 0^*10$
- Notation  $0^* = \{\epsilon, 0, 00, 000, ..., 0000000...\}$
- Konstruieren Sie zuerst einen unvollständigen Automaten
- Überführen Sie ihn in einen vollständigen Automaten
- Geben Sie die Überföhrungsfunktion an.
- Leiten Sie die 4 Worte 010, 0101, 10110, 10 ab.

# Einfache Automatenkonstruktion

- Wie konstruiert man endliche deterministische Automaten
- Man geht vom kleinsten Wort aus, was der Automat akzeptieren soll (Rumpf-Automat)
- Man erweitert den Automat durch geschickte Wahl der Übergänge, dass er alle Worte der Sprache versteht.

- Beispiel:

- Alle Wort aus dem Alphabet  $\{0,1\}$  die am Anfang eine 0 haben sollen.
- Kleinstes Wort ist  $w = „0“$



- Nun betrachtet man jeden Zustand und versucht die fehlenden Übergänge zu ergänzen, so dass nur Worte der Sprache erkannt werden.

# Aufgabe 3 DEA-Konstruktionen

- Das Alphabet sei  $\Sigma = \{0,1\}$ 
  - 1) Konstruieren Sie einen deterministischen Automaten, der alle Worte erkennt, die am Schluss eine „0“ haben.
  - 2) Konstruieren Sie einen deterministischen Automaten, der die Zeichenkette „00“ im Wort erkennt.
  - 3) Konstruieren Sie einen deterministischen Automaten, der alle Worte erkennt, die am Schluss die Zeichenfolge „00“ haben.
  - 4) Konstruieren Sie einen deterministischen Automaten, der alle Worte erkennt, die an der zweitletzten Stelle eine „0“ haben.