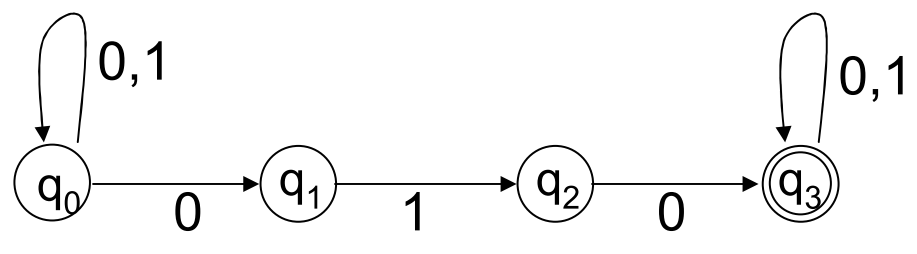
Sprachbeschreibungen und Maschinen

Formal ist ein nichtdeterministischer endlicher Automat (NEA) (ohne ε-Übergänge) ein 5-Tupel

Def: Ein (nichtdeterministischer) endlicher Automat (**NFA**) ist ein 5-Tupel (Q,Σ,δ,q_0,F), wobei

- Q eine endliche Menge von Zuständen ist,
- Σ ein endliches Alphabet
- $\delta: Q\times\Sigma\rightarrow 2^Q$ die Übergangsfunktion,
- q_0 der Startzustand und
- $F\subseteq Q$ die Menge akzeptierender Endzustände.

Bsp.:



	0	1
q_0	$\{q_0,q_1\}$	$\{q_0\}$
q_1	-	$\{q_2\}$
q_2	$\{q_3\}$	-
q_3	$\{q_3\}$	$\{q_3\}$

Sprachbeschreibungen und Maschinen

Satz: Sei N ein NFA und L = L(N). Dann gibt es einen DFA A mit L(A) = L

Beweis: Sei $N = (Q,\Sigma,\delta,q_0,F)$. Die folgende Konstruktion heißt auch "Potenzmengenkonstruktion". Um $A = (Q',\Sigma,\delta',q'_0,F')$ zu definieren setzen wir:

- $Q' = 2^Q$
- $q'_0 = \{q_0\}$
- $F' = \{R \in Q' \mid R \cap F \neq \{\}\}$
- $\delta'(R,a) = \bigcup_{r\in R} \delta(r,a) = \{q \in Q \mid \text{es gibt ein } r \in R \text{ mit } q \in \delta(r,a)\}$

Dann gilt:

$$\begin{aligned} w \in L(N) &\Leftrightarrow \delta(q_0,w) \cap F \neq \{\} \\ &\Leftrightarrow \delta(q'_0,w) \in F' \\ &\Leftrightarrow w \in L(A) \end{aligned}$$

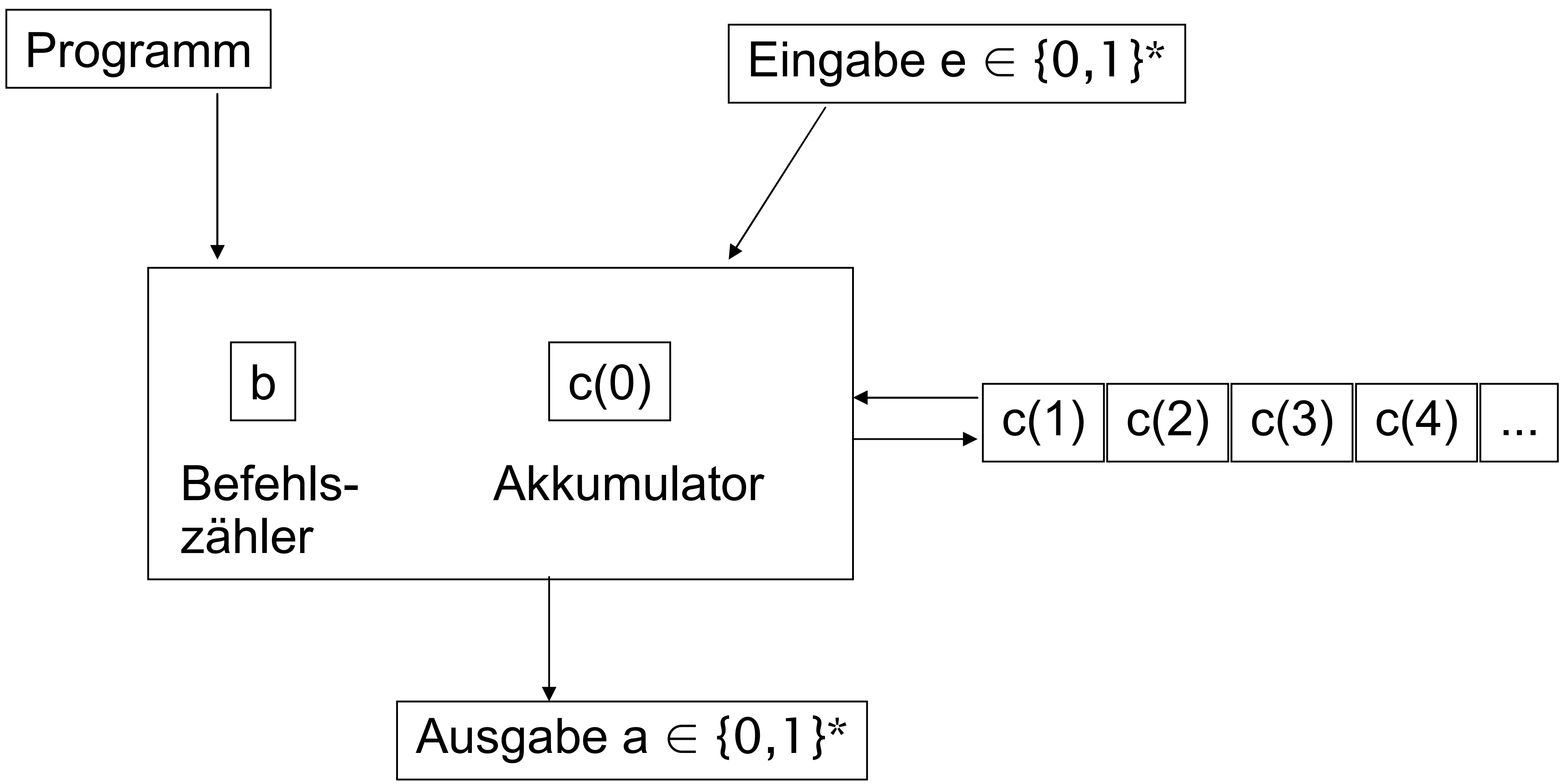
Hierbei bedeutet $\delta(q,w)$, dass die Übergangsfunktion δ mehrfach auf das Wort w angwendet wird, Buchstabe für Buchstabe und startend bei Zustand q.

Sprachbeschreibungen und Maschinen

- Die Frage, ob ein $w \in \Sigma^*$ ein Wort aus einer Sprache $L \subseteq \Sigma^*$ ist, kann unterschiedlich schwierig zu lösen sein
  - **Bsp 2.: In einem sehr komplizierten Fall ist sie nicht entscheidbar**:
    - **geg**: Codierung einer Random Access Machine (RAM, das entspricht in etwa einem herkömmlicher Computer mit unendlich viel Speicher), sowie ein $w \in \Sigma^*$
      Frage: Hält die RAM bei Eingabe w?

“nicht entscheidbar” heisst: es gibt keinen Algorithmus, der alle Instanzen das Problem lösen kann. (faszinierende Nebeneffekte, Busy Beaver)

Random Access Maschinen



Random Access Maschinen

Ein/Ausgabe

read	c(0) := head(e); e := e \ head(e); b := b+1;	falls |e| > 0
	c(0) := EOF; b := b + 1;	sonst
write	a := a c(0); b := b + 1;	

Arithmetik

add x	c(0) := c(0) + c(x); b := b + 1;	
sub x	c(0) := c(0) – c(x); b := b + 1;	falls c(x) < c(0)
	c(0) := 0; b := b + 1;	sonst
c *add* x	c(0) := c(0) + x; b := b + 1;	
c *sub* x	analog: Operation mit Konstante c	

Random Access Maschinen

Sprünge

goto j — b := j;

if (c(0) R i) *then goto* j; b := $\begin{cases} j & \text{falls } c(0)\ R\ i \\ b+1 & \text{sonst} \end{cases}$, $R \in \{<,>,=,\leq,\geq\}$

end — Programm hält

Speicherzugriffe

direkt

load x — c(0) := c(x); b := b + 1;

store x — c(i) := c(0); b := b + 1;

indirekt

iload x — c(0) := c(c(x)); b := b + 1;

istore x — c(c(i)) := c(0); b := b + 1;

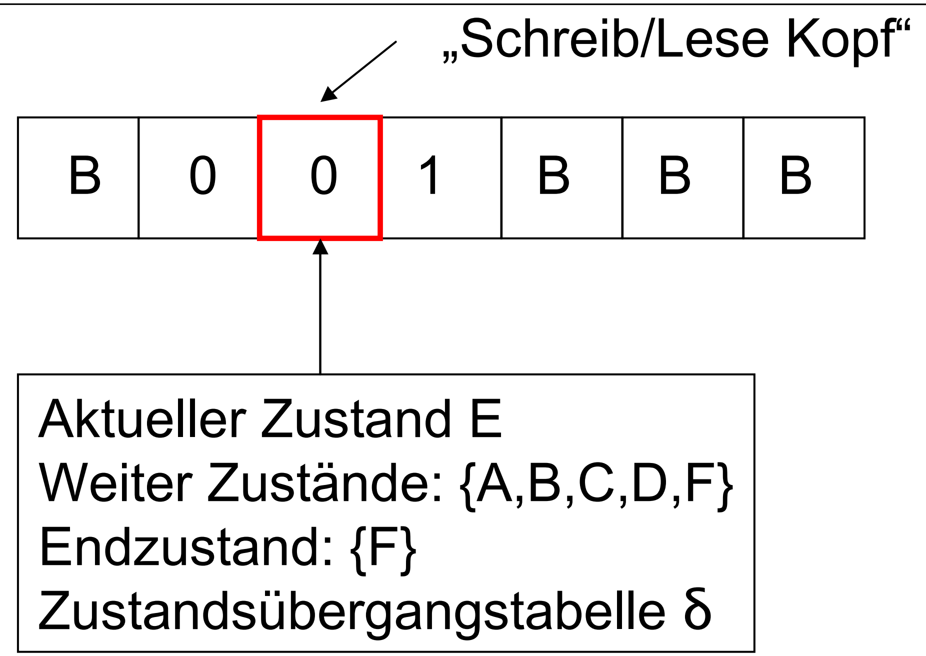
TECHNISCHE UNIVERSITÄT DARMSTADT

Turingmaschinen

- **Formal ist eine (1-Band) Turingmaschine ein 6-Tupel**

- **Def:** Eine (deterministischer 1-Band) Turingmaschine ist ein 6-Tupel $(Q,\Sigma,\Gamma,\delta,q_0,F)$, wobei
- Q eine endliche Menge von Zuständen ist,
- Σ ein endliches Alphabet
- $\Gamma := \Sigma \cup \{B\}$, B das so genannte Blank-Symbol
- $\delta: Q \times \Gamma \to Q \times \Gamma \times \{R,N,L\}$ die (partielle) Übergangsfunktion,
- q_0 der Startzustand und
- $F \subseteq Q$ die Menge akzeptierender Endzustände.

Turingmaschinen



„Programm“: Falls die Turingmaschine im Zustand q ist und das Zeichen a liest,
dann gehe in den Zustand q‘, überschreibe a durch a‘, und bewege den Kopf nach rechts, links oder gar nicht.

Schreibweise: $\delta(q,a) = (q',a',R)$

Turingmaschinen

- **Eine Mehrband- Turingmaschine ist ein 6-Tupel**

- **Def:** Eine (deterministischer 1-Band) Turingmaschine ist ein 6-Tupel $(Q,\Sigma,\Gamma,\delta,q_0,F)$, wobei
- Q eine endliche Menge von Zuständen ist,
- Σ ein endliches Alphabet
- $\Gamma := \Sigma \cup \{B\}$, B das so genannte Blank-Symbol
- δ: $Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{R,N,L\}^k$ die (partielle) Übergangsfunktion,
- q_0 der Startzustand und
- $F \subseteq Q$ die Menge akzeptierender Endzustände.

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Theorem 1: RAM und Turingmaschine können sich gegenseitig simulieren.

Theorem 2:

Church-Turing Hypothese:

Die von jeglicher Maschine berechenbaren
Funktionen sind genau die,
die von Turingmaschinen berechenbar sind.

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Def.: Zeit- und Platzkomplexität

Sei M eine determistische Turingmaschine (DTM), die für jede Eingabe hält.
Zeitkomplexität: $T_M(x)$:= #Schritte, die M mit Eingabe x ausführt.
Platzkomplexität: $S_M(x)$:= # verschiedener Speicherzellen, die der Kopf von M bei Eingabe x besucht.

$T_M(n) = \max\{T_M(x) \mid x \in \Sigma^{\leq n}\}$

Seien t,s: $\mathbb{N} \to \mathbb{N}$ Abbildungen. Dann ist M t(n)-zeitbeschränkt und s(n)-platzbeschränkt, falls

$$T_M(n) \leq t(n) \text{ und } S_M(n) \leq s(n) \text{ für alle } n \in \mathbb{N}.$$

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Wir sagen:
„asympototisch wächst f nicht stärker als g“, genau dann, wenn

$$\exists\, k > 0, n_0 > 0\ \forall n > n_0 : f(n) \leq k*g(n)$$

Man schreibt zudem auch $f(n) \in O(g(n))$, d.h.
$f(n) \in \{ h: \mathbb{N}\rightarrow\mathbb{N} \mid \exists\, k > 0, n_0 > 0\ \forall n > n_0 : f(n) \leq k*g(n) \}$

Satz: Jede t(n)-zeit, und s(n)-platzbeschränkte k-Band-DTM kann durch eine $O(t(n)*s(n))$ zeit- und $O(s(n))$ platzbeschränkte 1-Band DTM simuliert werden.
Satz: Jede t(n)-zeitbeschränkte RAM kann durch eine $O(t(n)^3)$ zeitbeschränkte DTM simuliert werden.

(ohne Beweis)

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Eine Turingmaschine heißt Zähler, wenn sie, gestartet mit bin(p), $p \in \mathbb{N}$,

bin(p-1), bin(p-2),...bin(1),bin(0)

hintereinander, immer auf dem gleichen Bandbereich erzeugt
und dann stoppt.

Satz: Es gibt einen $O(n)$ platz- und $O(2^n)$ zeitbeschränkten Zähler.

Beweis: gemeinsam in Übung