

# 9. CoMa Übung

5.6.13

Einführung Komplexitätstheorie & Turingmaschinen

Fragestellung: Welche algorithm. Probleme von Rechnern effizient lösbar?

- keine Zeit auf unnötige Aufgaben verschwenden
- bestmögliche Algorithmen beweisen
- schwere Aspekte an Problemen identifizieren
- Probleme in gut lösbarer Form formulieren

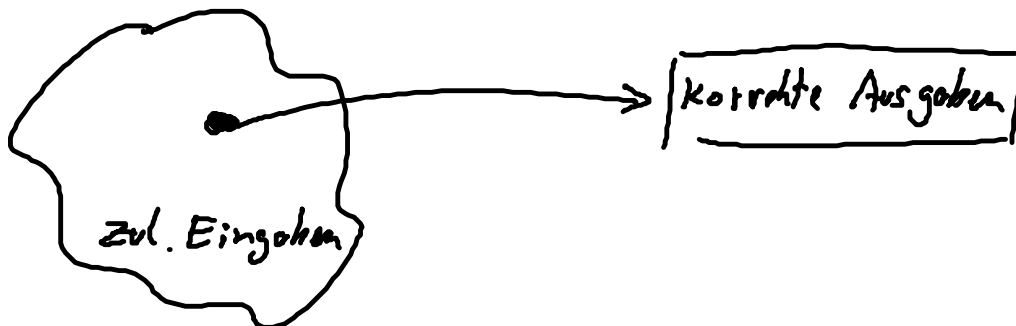
zu klären:

1. was sind algorithm. Probleme?
2. Was ist ein Rechner?
3. Was bedeutet effizient?

## 1. Algorithmische Probleme

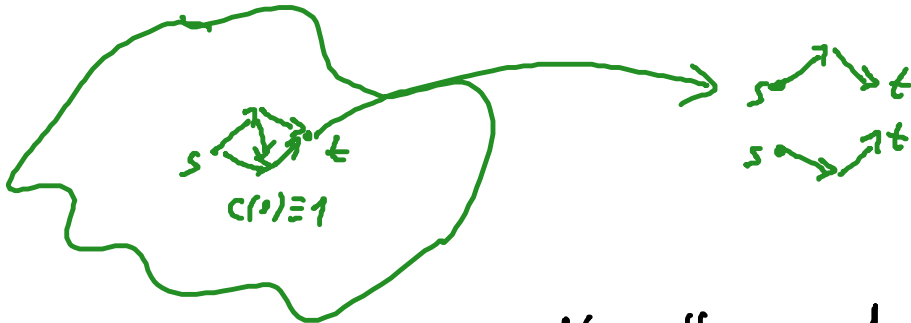
- Beschreibung der Menge der zulässigen Eingaben
- Besch. der Funktion, die einer zulässigen Eingabe die Menge der zulässigen Ausgaben zuweist

Ein / Ausgabe jeweils endliche Folge von Zeichen eines endl. Alph.



Beispiel: kürzeste Wege Problem:

- zulässige Eingabe: Menge aller Graphen  $G = (V, E)$  mit  $s, t \in V$
- korrekte Ausgabe: alle kürzesten  $s-t$  Wege



3. Effizient: es ex. ein Algorithmus dessen Laufzeit polynomial in der Eingabegröße ist

warum polynomiell: - Hardware Komplexität verdoppelt sich etwa alle 18 Monate (Moore's Law)

- bei polyn. Algorithmen: es ex.  $c > 1$   
max. lösbare Instanz um Faktor  $c$  wächst

- bei expon. Algorithmen:  
max. lösbare Instanz wächst um additive Konstante

2. Rechner: benötigen Aussage über alle Rechner

Rechnermodell: - komplex genug um alle anderen Rechner simulieren zu können

- einfach genug um Beweise zu ermöglichen

⇒ existiert so ein Modell

Churchsche Hypothese: Alle Rechnermodelle können sich gegenseitig simulieren. Menge der lösbarer Probleme nicht abhängig vom Rechnermodell.

- nicht wirklich beweisbar, wird als wahr angenommen

- simulation muss effizient sein

- aus reichend für Berechenbarkeitstheorie

erweiterte Hypothese: gegenseitige Simulation mit polyn. Mehr Aufwand

Ein Schub: Kosten für Rechenoperationen:

- Uniforme Kosten: Operationen kosten gleich viel, jeweils 1  
Problem: Addition großer Zahlen gleich teuer  
wie die von kleinen
- logarithm Kosten: Rechenop./Vergleiche haben Kosten von  $\log(n)$  bei Zahl  $n$ , (entspricht Kodierungslänge)  
→ fairer bei Zahlen versch. Länge

Rechnermodell: Turingmaschine

- besteht:
- endlosem Band (enthält Zeichen aus Alph.  $\Sigma$ )
  - Zustand aus Menge  $Q$
  - lese/schreibkopf (liest/schreibt Band, bewegt sich, ändert Zustand der Maschine)

Alphabete & Sprachen

$\Sigma$  ist Alphabet  $\hat{=}$  Menge von versch. Zeichen, Bsp.  $\Sigma = \{0,1\}$

Kleensche Hülle  $\Sigma^*$  (über Alphabet  $\Sigma$ ):

- enthält leere Wort  $\epsilon$
- für  $w \in \Sigma^*$  und  $a \in \Sigma$  ist  $wa \in \Sigma^*$

positive Hülle  $\Sigma^+ := \Sigma^* \setminus \{\epsilon\}$  wa heißt Konkatination

Sprache über Alphabet  $\Sigma$ : Teilmenge  $L \subseteq \Sigma^*$

- kann leer, unendlich oder endlich sein
- Elemente  $w \in L$  heißen Worte

- Mengenoperationen auf Sprachen möglich:

- Konkatination:  $L_1 L_2 := \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
- Potenzen:  $L^i := L^{i-1} L$ ,  $L^0 := \{\epsilon\}$
- Kleensche Hülle:  $L^+ := \bigcup_{i=0}^{\infty} L^i$

# Deterministische Turingmaschine:

formal:  $(Q, \Sigma, \Gamma, B, q_0, \delta, F)$

$Q$ : endl. Zustandsmenge

$\Sigma$ : Alphabet

$\Gamma$ : Bandalphabet,  $\Gamma = \Sigma \cup \{B\}$

$B$ : leerzeichen

$q_0$ : startzustand,  $q_0 \in Q$

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$

$F \subseteq Q$ : Menge der akzept. Zustände  
links, neutral, rechts

lese zeichn + akt. Zustand  
 $\rightarrow$  schreibe zeichn, ändere Zustand, bewege Band



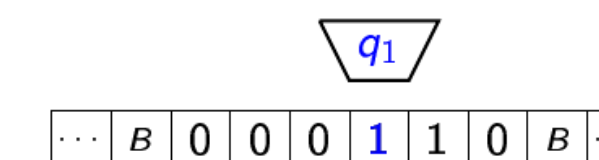
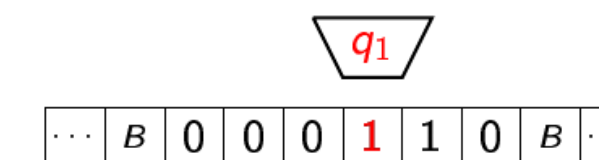
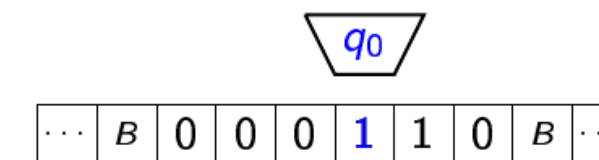
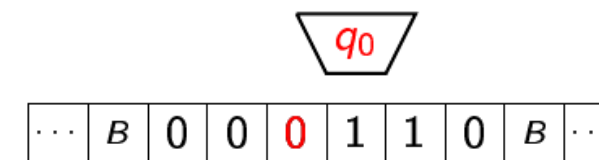
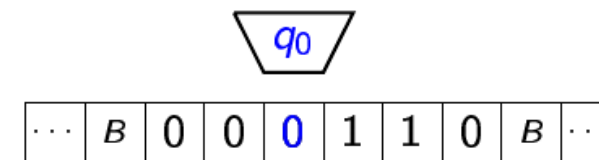
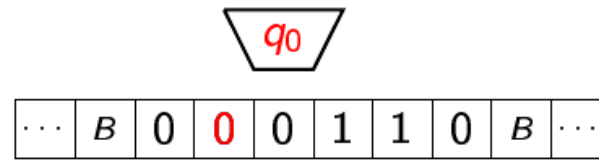
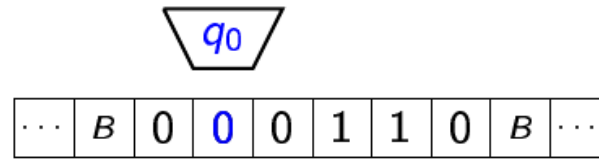
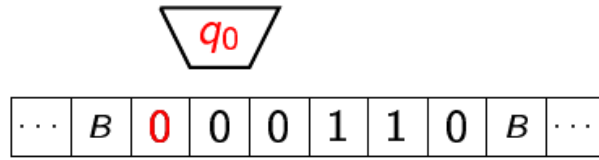
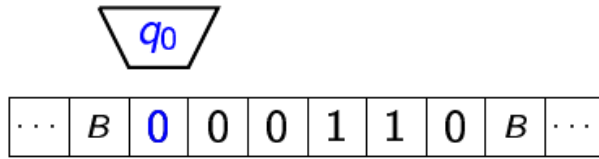
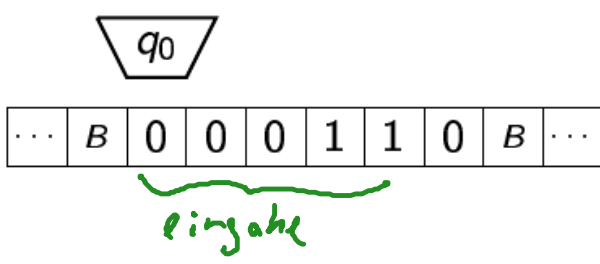
- Verhalten:
- Zustand  $q_0$  + gelesenes Zeichen  $\epsilon$  per  $\delta$  abbilden
  - Zustand. ändern, Zeichen schreiben, kopf. bewegen
  - Stopp: bei  $\delta(q, \epsilon) = \delta(q, \epsilon, N)$  keine Änderung
  - nach Stopp:
    - Rechenergebnis ablesen (auf Band)
    - Entscheidungsproblem laut  $q$  ablesen
- $q \in F \Rightarrow$  Antwort ist Ja

$\sim$  Entscheidungsproblem: ist Wort  $w \in L$ ,  $L$  ist Sprache

## Beispiel:

Übergänge:

$\delta$	0	1	B
$q_0$	$(q_0, 0, R)$	$(q_1, 1, N)$	$(q_0, B, N)$
$q_1$	$(q_1, 0, N)$	$(q_1, 1, N)$	$(q_1, B, N)$



- = Zustand + Zeichen lesen
- = Zustand + Zeichen schreiben

$\delta(q_1, 1) = (q_1, 1, N)$   
 $\Rightarrow$  Abbruch

TM endet in  $q_0 \Rightarrow$  Eingabe enthält keine 1  
 $q_1 \Rightarrow$  Eingabe enthält 1

# Entwurf einer TM:

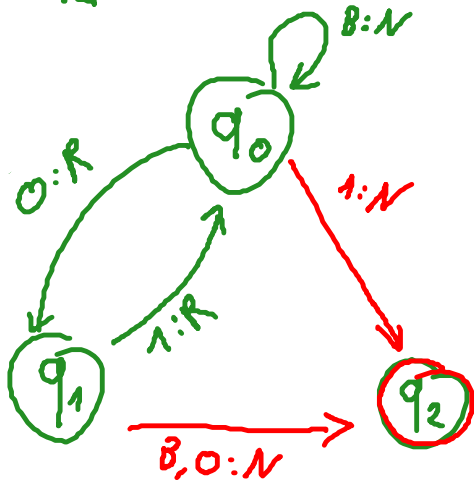
Ziel: erkenne ob Eingabewort aus Sprache  $\{(01)^n \mid n \geq 0\} =: L$

bsp:  $\left. \begin{matrix} 01 \\ 0101 \\ \epsilon \\ 010101 \end{matrix} \right\} \text{ok}$        $\left. \begin{matrix} 010 \\ 101 \\ 1 \end{matrix} \right\} \text{nicht ok}$

Idee: weiterer Zustand für zuletzt gelesenes Zeichen:

$\delta$	0	1	B
$q_0$	$(q_1, 0, R)$	$(q_2, 1, N)$	$(q_0, B, N)$
$q_1$	$(q_2, 0, N)$	$(q_0, 1, R)$	$(q_2, B, N)$
$q_2$	—	—	—

$q_0$  = start bzw. als letztes 1 gelesen (erwarte 0)  
 $q_1$  = letztes Zeichen war 0 (erwarte 1)  
 $q_2$  = ungültiger Zustand



$q_2: w \notin L$   
 $q_0: w \in L$

Ablauf: Eingabe: 010

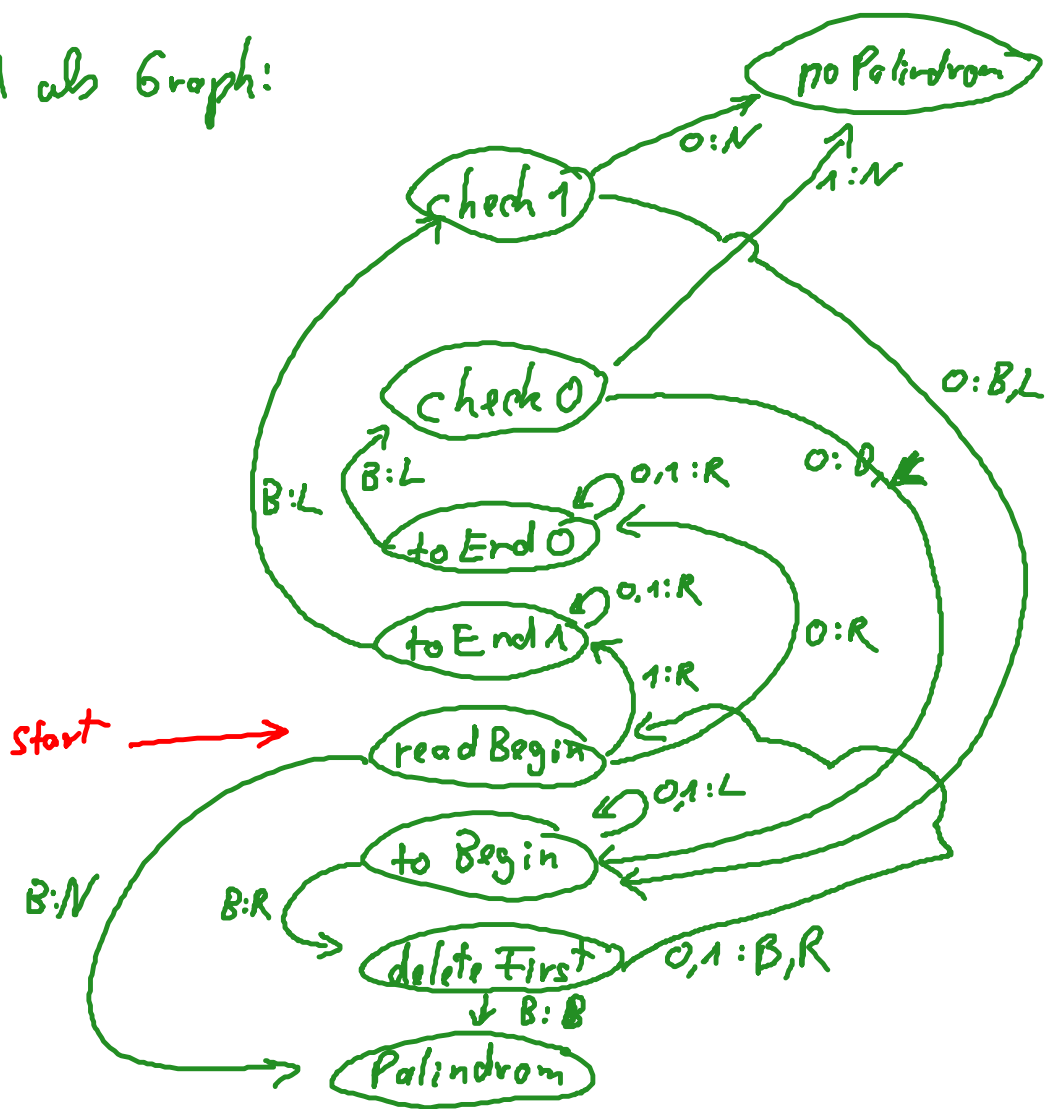
$q_0 010 \quad (q_0, 0) \rightarrow (q_1, 0, R)$   
 $\vdash 0q_110$   
 $\vdash 01q_00$   
 $\vdash 010q_1 \quad \text{Es } (q_1, B) \rightarrow (q_2, B, N)$   
 $\vdash 010q_2 \Rightarrow \text{Abbruch, } 010 \text{ ist nicht } \in L$

## Entwurf einer TM zum Erkennen von Palindromen:

- Idee: - Merke Zeichen am Anfang  
 - vergleiche mit Ende  
 - Zeichen gleich?  $\rightarrow$  lösche beide und gehe zu Anfang  
 $\rightarrow$  rekursiver Start

ungleich: Abbruch

TM als Graph:



$q_0 = \text{read Begin}$   
 $F = \{ \text{Palindrom} \}$