

# Komplexitätstheorie

## Script Kapitel 3

**Fragestellung:** Welcher Aufwand (Laufzeit, Speicherplatz) ist notwendig, um ein entscheidbares Problem zu lösen?

- formale Definition von **Komplexitätsklassen** mittels Turingmaschinen
- **nichtdeterministische** Turingmaschinen und die Klasse  $\text{NP}$
- einige beweisbar “schwierige” Probleme ( **$\text{NP}$ -Vollständigkeit**)

## Verschiedene Funktionen

Annahme: ein Programm wird berechnet, wobei die Funktion  $f$  die Anzahl der auszuführenden Operationen in Abhängigkeit von der Eingabelänge angibt. In der Tabelle ist die Zeit für die Ausführung angegeben, falls ein Computer eine Million Operationen pro Sekunde ausführt.

$f \setminus n$	5	10	50	100	200
$n^2$	0,000 025 s	0,0001 s	0,0025 s	0,01 s	0,04 s
$n^5$	0,003 125 s	0,1 s	312,5 s	3 h	89 h
$2^n$	0,000 032 s	0,001 024 s	36 a	$10^{17}$ a	$10^{47}$ a
$n^n$	0,003 125 s	3 h	$10^{71}$ a	$10^{185}$ a	$10^{447}$ a

# Definition Komplexitätsklassen

**Definition** Für eine Mehrband-Turingmaschine  $M$  mit dem Eingabealphabet  $\Sigma$  ist  $time_M$  die Funktion  $time_M : \Sigma^* \rightarrow \mathbb{N}$ , wobei  $time_M(x)$  die **Anzahl der Rechenschritte** von  $M$  bei der Abarbeitung der Eingabe  $x$  ist.

Sei  $f : \mathbb{N} \rightarrow \mathbb{N}$  eine Funktion. Die **Klasse**  $TIME(f(n))$  besteht aus allen Mengen  $A$ , die von einer Mehrband-Turingmaschine  $M$  entschieden werden können mit  $time_M(x) \leq f(|x|)$  für alle Eingaben  $x$ .

# Die Komplexitätsklasse $\mathbb{P}$

**Definition** Ein **Polynom** ist eine Funktion  $p: \mathbb{N} \rightarrow \mathbb{N}$  der Form

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_2 n^2 + a_1 n + a_0$$

wobei  $k \in \mathbb{N}$  und  $a_i \in \mathbb{N}$  für  $i = 0, 1, 2, \dots, k$  gilt.

**Definition** Die Komplexitätsklasse  $\mathbb{P}$  ist wie folgt definiert:

$$\begin{aligned} \mathbb{P} &= \{A \mid \text{es gibt eine TM } M, \text{ die } A \text{ entscheidet,} \\ &\quad \text{und ein Polynom } p \text{ mit } \textit{time}_M(x) \leq p(|x|)\} \\ &= \bigcup_{p \text{ Polynom}} \textit{TIME}(p(n)) \end{aligned}$$

## Beispiele zu $\mathbb{P}$

**Palindromproblem**, d.h. die Menge  $PAL = \{w \in \Sigma^* \mid w = w^R\}$ , wobei  $w^R$  das Wort  $w$  von rechts nach links gelesen ist.

### Eulergraph-Problem

**gegeben:** ein ungerichteter Graph  $G$

**Frage:** Gibt es einen in  $G$  einen Kreis, der jede Kante genau einmal enthält?

### Teilerfremdheitsproblem

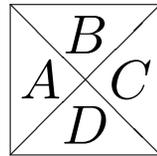
**gegeben:** natürliche Zahlen  $m, n$

**Frage:** Sind  $m$  und  $n$  teilerfremd, d.h. gilt  $ggT(m, n) = 1$ ?

**Beachte:** Hier braucht man einen Algorithmus mit polynomieller Laufzeit bezüglich  $|bin(m)| + |bin(n)|$ , d.h. bezüglich  $\log m + \log n$ .

# Das Domino-Problem

- Ein **Dominostein** ist in 4 Dreiecke aufgeteilt, die jeweils ein Symbol aus einem Alphabet  $\Sigma$  enthalten.



- Ein **Dominospiel** besteht aus endlich vielen Sorten von Dominosteinen, wobei von jeder Sorte beliebig viele Steine vorhanden sind.
- Zwei Steine dürfen benachbart sein, wenn ihre Grenzen übereinstimmen.
- Die Steine dürfen nicht gedreht werden.

- Ein rechteckiger **Rahmen** für ein Dominospiel besteht aus Steinen mit **einem** Symbol aus  $\Sigma$ , z.B.:

<i>A</i>	<i>B</i>	<i>B</i>	<i>A</i>	<i>B</i>	<i>B</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>B</i>
<i>C</i>									<i>A</i>
<i>C</i>									<i>C</i>
<i>A</i>									<i>A</i>
<i>C</i>									<i>C</i>
<i>A</i>	<i>B</i>	<i>B</i>	<i>A</i>	<i>B</i>	<i>B</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>A</i>

## Domino-Problem

**gegeben:** Domino-Spiel  $\Pi$ , Rahmen  $R$

**Frage:** Kann man den Rahmen  $R$  mit Steinen aus  $\Pi$  auslegen?

# Bemerkungen zum Domino-Problem

- Ein gegebenes Domino-Problem  $(\Pi, R)$  kann man durch systematisches Probieren lösen. Eine bessere Lösung ist im allgemeinen Fall nicht zu sehen.
- Für ein zweidimensionales Bild über  $\Pi$  kann man in polynomieller Zeit entscheiden, ob es eine Lösung des Domino-Problems ist.
- **Nichtdeterministischer Algorithmus**  $A$  mit polynomieller Laufzeit:
  1. Rate ein Bild  $B$  über  $\Pi$ .
  2. Entscheide, ob  $B$  eine Lösung für  $(\Pi, R)$  ist.
- Gibt es eine Lösung, so gibt es einen Lauf von  $A$  mit Ausgabe "JA". Anderenfalls liefert  $A$  immer die Ausgabe "NEIN".
- Formalisierung nichtdeterministischer Algorithmen durch **nichtdeterministische Turingmaschinen**.

# Nichtdeterministische Turingmaschinen

Eine nichtdeterministische Turingmaschine (NTM) ist gegeben durch ein 7-Tupel  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ . Hierbei sind

- $Z$  eine endliche Menge (Zustandsmenge),
- $\Sigma$  ein Alphabet (Eingabealphabet),
- $\Gamma$  ein Alphabet (Bandalphabet) mit  $\Sigma \subseteq \Gamma$ ,
- $\delta: Z \times \Gamma \rightarrow 2^{Z \times \Gamma \times \{L, R, N\}}$  eine Funktion (Überföhrungsfunktion),
- $z_0 \in Z$  (Anfangszustand),
- $\square \in \Gamma \setminus \Sigma$  (Leerzeichen, Blank),
- $E \subseteq Z$  (Menge der Endzustände).

# Konfigurationen von NTM

- Konfigurationen sind formal definiert wie bei TM.
- Nachfolgerrelation  $\vdash$  ist ähnlich definiert wie bei TM, **aber:**  
Eine Konfiguration kann mehrere Nachfolgekonfigurationen haben.

**Beispiel:** Es sei  $\delta(z, a) = \{(z_1, a, R), (z_2, b, L), (z_3, c, N)\}$ .

Dann gelten für die Konfiguration  $aczabb$  folgende Nachfolgerrelationen:

$$aczabb \vdash acaz_1bb, \quad aczabb \vdash azcbbb, \quad aczabb \vdash acz_3cbb.$$

# Akzeptierte Mengen von NTM

**Definition** Sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$  eine nichtdeterministische Turingmaschine. Die von  $M$  **akzeptierte Menge**  $T(M)$  ist definiert durch

$$T(M) = \{x \in \Sigma^* \mid z_0x \vdash^* wzy \text{ für } z \in E, w, y \in \Gamma^*\}.$$

$T(M)$  ist also die Menge aller Eingaben, für die  $M$  bei geschickter Wahl der Nachfolgekonfigurationen stoppt.

## Beispiel einer NTM

$$L = \{a^{n_1}ba^{n_2}b \cdots a^{n_{k-1}}ba^{n_k} \mid k \in \mathbb{N}, n_i \geq 0 \text{ für } 1 \leq i \leq k, \\ \text{es gibt } i, j \text{ mit } i < j \text{ und } n_i = n_j \geq 1\}$$

Arbeitsweise einer NTM, die  $L$  akzeptiert:

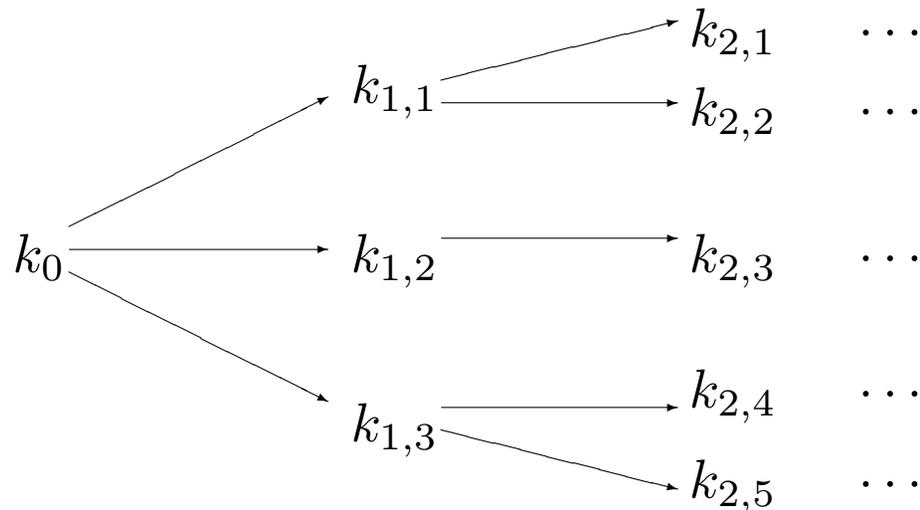
- Wähle nichtdeterministisch zwei Blöcke von  $a$ 's.
- Falls die Blöcke gleiche Länge haben, so akzeptiere; anderenfalls halte nicht.

# Graphische Interpretation von TM und NTM

**TM:** Es gibt für jede Eingabe **genau einen** Berechnungspfad:

$$k_0 \vdash k_1 \vdash k_2 \vdash \dots$$

**NTM:** Berechnungspfade verzweigen sich (**Baum**)



Akzeptiert wird, falls **eine** Endkonfiguration erreichbar ist.

# Äquivalenz von TM und NTM

## Satz

Für jede NTM  $M$  existiert eine TM  $M'$  mit  $T(M') = T(M)$

## Beweisidee (Dovetailing)

- Bestimme für wachsendes  $n$  alle Konfigurationen, die in  $n$  Schritten erreichbar sind.
- Wird eine Endkonfiguration erreicht, so stoppe.

# Äquivalenz von TM und NTM – Fortsetzung

Simulation von  $M$  durch folgende 2-Band-TM:

- Band 1 enthält alle nach  $n$  Schritten von  $M$  erreichbaren Konfigurationen (zuerst also die Startkonfiguration).
- Ist eine der Konfigurationen auf Band 1 eine Endkonfiguration von  $M$ , so stoppe.
- Anderenfalls schreibe für jede Konfiguration auf Band 1 alle Nachfolgekongfigurationen auf Band 2. Lösche dabei Band 1.
- Verschiebe den Inhalt von Band 2 nach Band 1.  
Band 1 enthält nun alle nach  $n + 1$  Schritten erreichbaren Konfigurationen.

# Komplexität nichtdeterministischer Turingmaschinen

**Definition** Sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$  eine nichtdeterministische Turingmaschine;  $time_M(x)$  ist definiert durch

$$time_M(x) = \begin{cases} \text{Minimum der Längen aller akzeptierenden} \\ \text{Rechnungen von } M \text{ auf } x & \text{falls } x \in T(M), \\ 0 & \text{falls } x \notin T(M). \end{cases}$$

**Definition** Sei  $f : \mathbb{N} \rightarrow \mathbb{N}$  eine Funktion. Die Klasse  $NTIME(f(n))$  besteht aus allen Mengen  $A$ , die von einer nichtdeterministischen Mehrband-Turingmaschine  $M$  erkannt werden können mit  $time_M(x) \leq f(|x|)$ .

# Die Komplexitätsklasse $\text{NP}$

**Definition** Die Komplexitätsklasse  $\text{NP}$  ist wie folgt definiert:

$$\begin{aligned}\text{NP} &= \{A \mid \text{es gibt eine NTM } M \text{ und ein Polynom } p \text{ mit } T(M) = A \\ &\quad \text{und } \textit{time}_M(x) \leq p(|x|)\} \\ &= \bigcup_{p \text{ Polynom}} \text{NTIME}(p(n))\end{aligned}$$

**Beispiel:** Das Domino-Problem ist in  $\text{NP}$ .

# Die P-NP-Problematik

Gilt  $P = NP$ ?

# Polynomiale Reduzierbarkeit

**Definition** Seien  $A, B \subseteq \Sigma^*$  Mengen. Dann heißt  $A$  auf  $B$  **polynomial reduzierbar** (in Zeichen  $A \leq_p B$ ) falls es eine totale und mit polynomialer Komplexität berechenbare Funktion  $f : \Sigma^* \rightarrow \Sigma^*$  gibt, so dass für alle  $x \in \Sigma^*$  gilt:

$$x \in A \iff f(x) \in B.$$

## Lemma

- (i) Gilt  $A \leq_p B$  und  $B \in \text{NP}$ , so ist auch  $A \in \text{NP}$ .
- (ii) Gilt  $A \leq_p B$  und  $B \in \mathbb{P}$ , so ist auch  $A \in \mathbb{P}$ .

# NP-vollständige Mengen (Probleme)

Wir wollen jetzt die im gewissen Sinne “schwierigsten Probleme” aus  $\text{NP}$  charakterisieren und führen dazu folgenden Begriff ein.

**Definition** Eine Menge  $A$  heißt **NP-vollständig** genau dann, wenn

- (i)  $A \in \text{NP}$  und
- (ii) für alle Mengen  $A' \in \text{NP}$   $A' \leq_p A$  gilt.

# Folgerungen

Es gelten offensichtlich folgende Sätze.

## Satz

Sei  $A$  NP-vollständig und  $B \in \text{NP}$ .  
Dann folgt aus  $A \leq_p B$  die NP-Vollständigkeit von  $B$ .

## Satz

Sei  $A$  NP-vollständig; dann gilt  $A \in \mathbb{P} \Leftrightarrow \mathbb{P} = \text{NP}$ .

# NP-Vollständigkeit des Domino-Problems

**Satz:** Das Domino-Problem ist NP-vollständig.

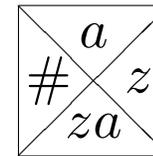
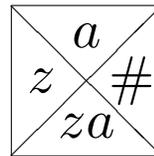
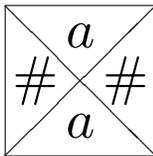
## Beweisidee:

- Das Domino-Problem ist offensichtlich in NP.
- Zu einer beliebigen NTM  $M$  konstruiere Dominospiel  $\Pi$ .
- Jede Eingabe  $w$  wird in polynomialer Zeit auf einen Rahmen  $R_w$  abgebildet.
- Eine korrekte Ausfüllung von  $R_w$  entspricht einem akzeptierenden Lauf von  $M$  bei Eingabe  $w$ .  
Also:  $w \in T(M)$  genau dann, wenn  $(\Pi, R_w)$  eine Lösung hat.

# Domino-Problem: Konstruktion des Dominospiels

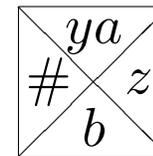
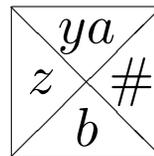
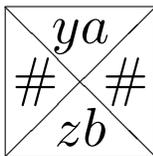
NTM sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ .

$\Pi$  enthält folgende Steine, wobei  $a, b \in \Gamma$ ,  $y, z \in Z$ ,  $\# \notin \Gamma \cup Z$ :



“Kopf von links”

“Kopf von rechts”



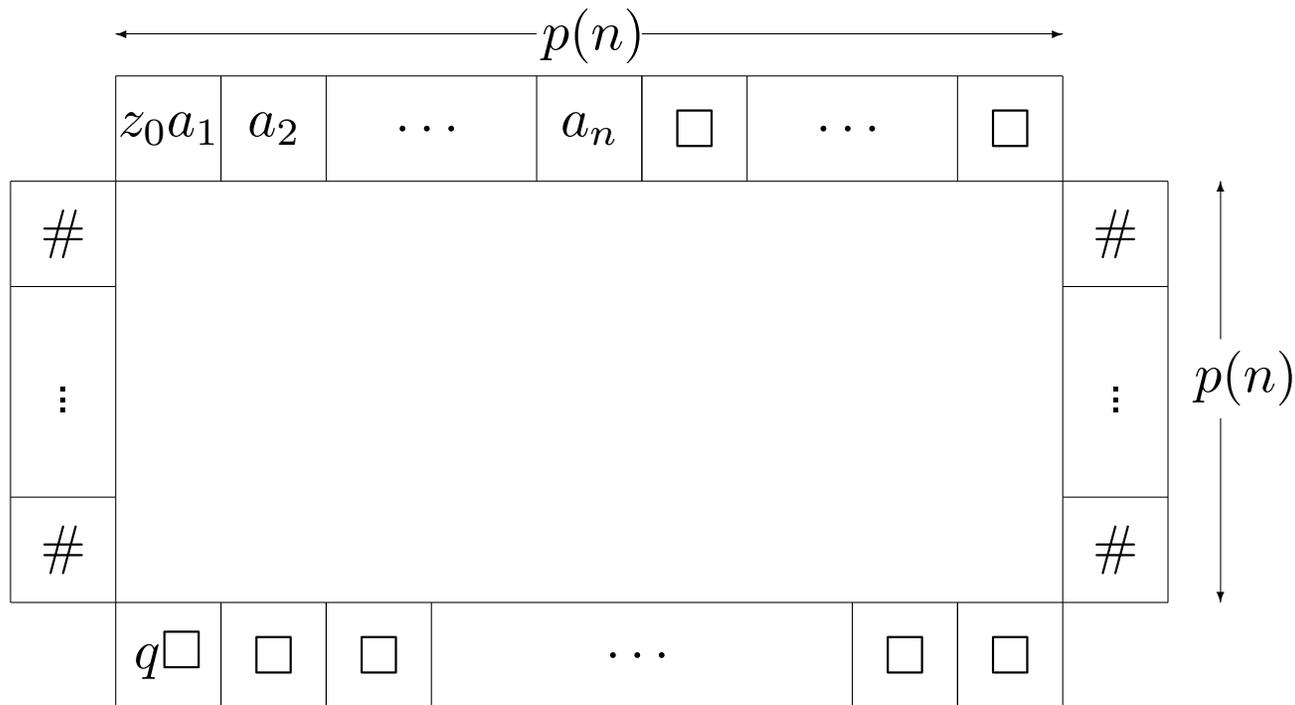
$(z, b, N) \in \delta(y, a)$

$(z, b, L) \in \delta(y, a)$

$(z, b, R) \in \delta(y, a)$

# Domino-Problem: Konstruktion des Rahmens

- Es gelte  $time_M(n) \leq p(n)$  für ein Polynom  $p$ .
- Eingabe sei  $a_1 a_2 \cdots a_n$ .



# Domino-Problem: Simulation eines Laufes der NTM

- Ist die obere Hälfte einer Domino-Zeile eine Konfiguration  $k$ , so ist die untere Hälfte eine Nachfolgekonfiguration von  $k$ .
- Die untere Hälfte einer Domino-Zeile wird auf die obere Hälfte der nächsten Zeile kopiert.
- Oberer Rand ist Startkonfiguration, unterer Rand ist Endkonfiguration; d.h. Lösung des Domino-Problems entspricht akzeptierendem Lauf.
- Technische Voraussetzungen an NTM  $M = (Z, \Sigma, \Gamma, z_0, \delta, \square, E)$ :
  - $E = \{q\}$ ,  $\delta(q, \square) = \{(q, \square, N)\}$ .
  - Der Schreib/Lese-Kopf befindet sich niemals links von seiner Startposition.
  - Bei Akzeptanz ist das Band leer, und der Schreib/Lese-Kopf ist in seiner Startposition.

# Das Erfüllbarkeitsproblem (SAT)

**Problem:** Erfüllbarkeitsproblem (*Satisfiability*, **SAT**)

**Eingabe:** Boolesche Formel  $\varphi$  in konjunktiver Normalform

**Frage:** Ist die Formel  $\varphi$  erfüllbar?

**Satz:** **SAT** ist NP-vollständig.

## Beweisidee

- **SAT**  $\in$  NP: Rate Belegung und prüfe, ob sie die Formel erfüllt.
- NP-Vollständigkeit: Zeige **Domino**  $\leq_p$  **SAT**.

## Weitere NP-vollständige Probleme

**Problem:** *3-Satisfiability*, (**3SAT**)

**Eingabe:** Boolesche Formel  $\varphi$  in konjunktiver Normalform mit höchstens 3 Variablen je Klausel

**Frage:** Ist die Formel  $\varphi$  erfüllbar?

**Problem:** Cliques-Problem (**Clique**)

**Eingabe:** ungerichteter Graph  $G$ ,  $k \in \mathbb{N}$

**Frage:** Enthält  $G$  einen vollständigen Teilgraphen mit  $k$  Knoten?

## Weitere NP-vollständige Probleme

- Problem:** Hamiltonkreis-Problem (*Hamiltonian Circuit*, **HC**)
- Eingabe:** ungerichteter Graph  $G$
- Frage:** Gibt es in  $G$  einen Kreis, der jeden Knoten genau einmal passiert?
- 
- Problem:** Rundreiseproblem (*Traveling Salesperson Problem*, **TSP**)
- Eingabe:** Schranke  $M \in \mathbb{N}$ ,  $n$  Städte,  $n \times n$ -Kostenmatrix  $C$  mit  $C_{i,j}$  = Kosten einer Fahrt von Stadt  $i$  nach Stadt  $j$
- Frage:** Gibt es eine Rundreise durch alle Städte mit Kosten von höchstens  $M$ ?

## Weitere NP-vollständige Probleme

**Problem:** Partitionierungs-Problem (**PARTITION**)

**Eingabe:** Menge  $U$ , Gewichtsfunktion  $g : U \rightarrow \mathbb{N}$

**Frage:** Gibt es eine Zerlegung  $U = V \cup W$ ,  $V \cap W = \emptyset$ ,  
so dass  $\sum_{v \in V} g(v) = \sum_{w \in W} g(w)$ ?

**Problem:** Bin Packing

**Eingabe:**  $m$  Gegenstände mit Volumen  $a_1, a_2, \dots, a_m$ ,  
 $k$  Behälter mit Volumen jeweils  $\ell$ .

**Frage:** Kann man die Gegenstände auf die  $k$  Behälter verteilen?

# NP-vollständige Probleme in der Praxis

- zumeist **Optimierungsprobleme**, z.B.
  - Problem:** Rundreiseproblem
  - Eingabe:**  $n$  Städte,  $n \times n$ -Kostenmatrix  $C$
  - gesucht:** Rundreise mit minimalen Kosten
- NP-Vollständigkeit des zugehörigen Entscheidungsproblems heißt:  
Suche nach einem effizienten und **exakten** Algorithmus ist “aussichtslos”.
- Häufig existieren aber gute **Näherungsalgorithmen**.