# Formale Sprachen und Komplexitätstheorie

WS 2019/20

**Robert Elsässer**

# 3. Komplexität

## Von Berechenbarkeit zu Komplexität

- Berechenbarkeit in Bezug auf Entscheidbarkeit und Aufzählbarkeit betrachtet nur prinzipielle Lösbarkeit von Problemen mit Hilfe von Computern
- Prinzipiell lösbare Probleme können praktisch nicht lösbar sein, weil jeder Algorithmus zur Lösung des Problems zu viel Zeit (und/oder Platz) benötigt.
- Komplexitätstheorie versucht Probleme gemäß des Zeit- und Platzbedarfs des besten Algorithmus zu ihrer Lösung zu klassifizieren.
- Konzentrieren uns auf Zeitbedarf und die Klassen $P$ und $NP$

# Laufzeit einer DTM

**Definition**

DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{n-1}, q_n)$ halte bei jeder Eingabe.

- Für $w$ aus $\Sigma^*$ ist $T_M(w)$ die Anzahl der Rechenschritte von $M$ bei Eingabe $w$.
- Für eine natürliche Zahl $n$ ist $T_M(n) \coloneqq \max\{T_M(w) \mid w \text{ aus } \Sigma^{\leq n}\}$.
- Die Funktion $T_M$ heißt Zeitkomplexität oder Laufzeit der DTM $M$.
- DTM $M$ hat Laufzeit $O\big(f(n)\big)$, wenn $T_M(n) = O\big(f(n)\big)$.

**Satz**

Sei $t$ eine monoton wachsende Funktion mit $t(n) \geq n$.
Jede Mehrband-DTM mit Laufzeit $t(n)$ kann durch eine 1-Band-DTM mit Laufzeit $O(t(n)^2)$ simuliert werden.

## Rucksackproblem und Verifizierbarkeit

- **Gegeben:**
  - $n$ Gegenstände mit
  - Gewichten $G = \{g_1, g_2, \dots, g_n\}$ und
  - Werten $W = \{w_1, w_2, \dots, w_n\}$,
  - sowie zulässiges Gesamtgewicht $g$.
- **Gesucht:**
  Teilmenge $S$ aus $\{1,2,\dots,n\}$ mit $\Sigma_S w_i$ maximal
  unter der Bedingung $\Sigma_S g_i \leq g$.

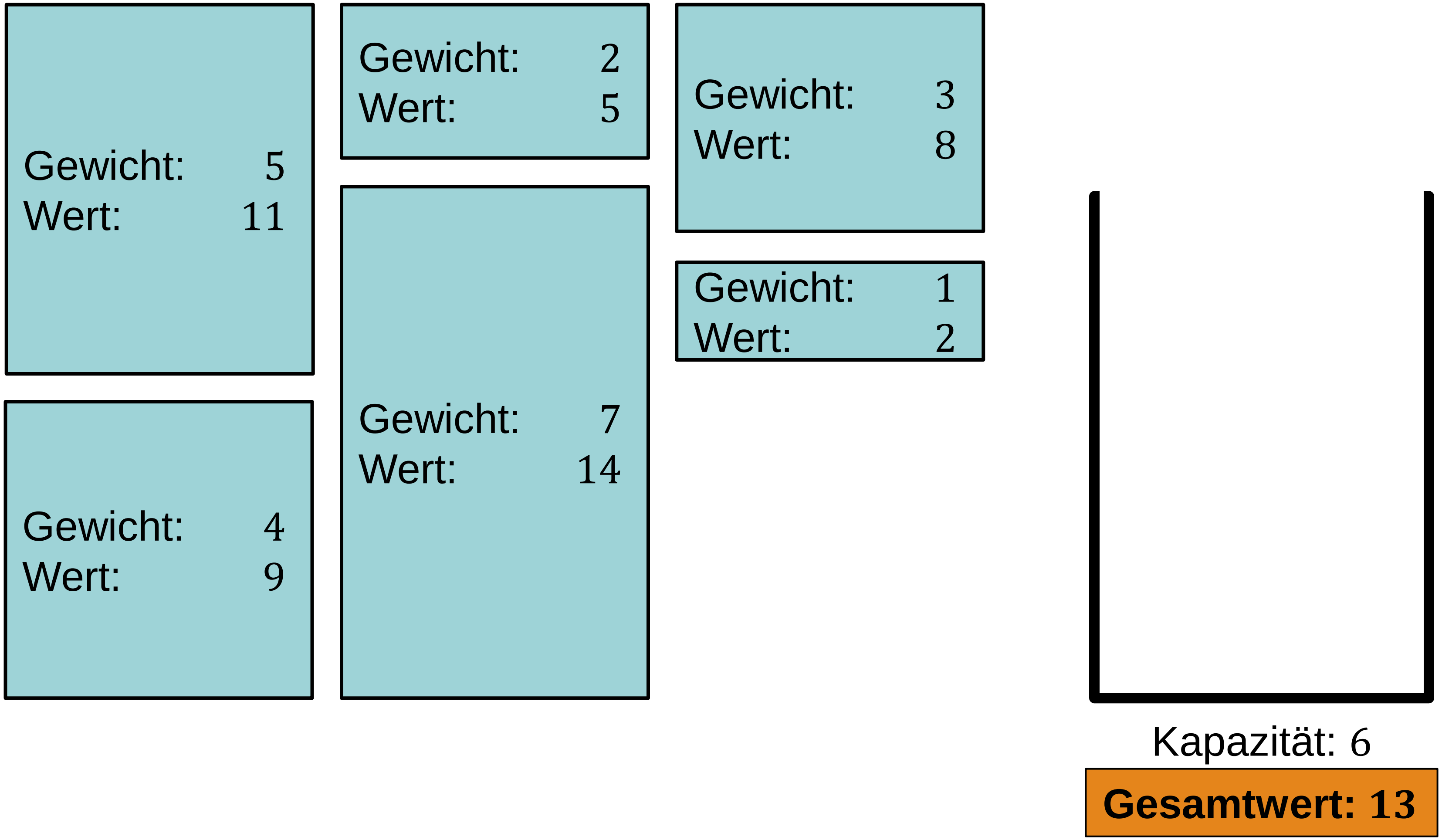
## Rucksackproblem und Verifizierbarkeit

- **Gegeben:**
  - $n$ Gegenstände mit
  - Gewichten $G = \{g_1, g_2, \dots, g_n\}$ und
  - Werten $W = \{w_1, w_2, \dots, w_n\}$,
  - sowie zulässiges Gesamtgewicht $g$.
- **Gesucht:**
  Teilmenge $S$ aus $\{1,2,\dots,n\}$ mit $\Sigma_S w_i$ maximal
  unter der Bedingung $\Sigma_S g_i \leq g$.

$$RS_{ent} \coloneqq \left\{ \begin{array}{c} \langle G, W, g, w \rangle \mid \text{es existiert eine Teilmenge } S \text{ aus } \{1,2,\dots,n\} \\ \text{mit } \Sigma_S g_i \leq g \text{ und } \Sigma_S w_i \geq w \end{array} \right\}$$

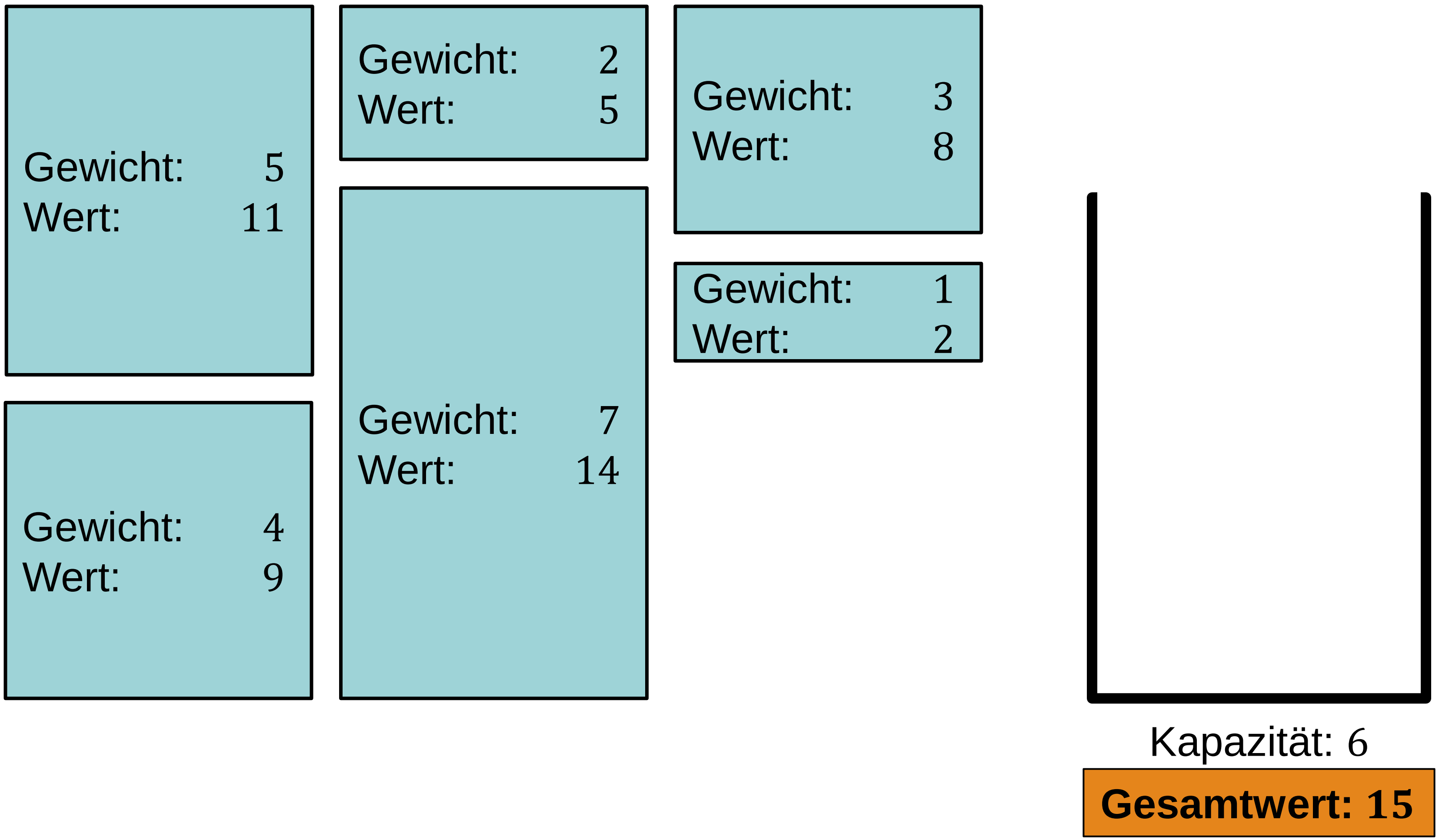
# Rucksackproblem

**Beispiel:**



# Rucksackproblem

**Beispiel:**



# 3. Komplexität

## Rucksackproblem und Verifizierbarkeit

- Liegt $RS_{ent}$ in $P$?
- Algorithmus mit Laufzeit $O(n\,W)$ aus Algodat ist kein polynomieller Algorithmus!
- Kennen weder Polynomialzeit DTM für $RS_{ent}$, noch können wir beweisen, dass eine solche existiert.
- $RS_{ent}$ teilt diese Eigenschaften mit vielen anderen Problemen.

# Verifizierer

**Definition**

Sei $L$ eine Sprache. DTM $V$ heißt Verifizierer für $L$, falls

$$L = \{w \mid \text{es gibt ein } c, \text{so dass } V \; \langle w, c\rangle \text{ akzeptiert}\}$$

$c$: Zertifikat oder Zeuge

$V$ heißt polynomieller Verifizierer, falls eine natürliche Zahl $k$ existiert mit

$L = \{w \mid \text{es gibt ein } c \text{ mit } |c| \leq |w|^k, \text{sodass } V \; \langle w, c\rangle \text{ akzeptiert}\}$

und die Laufzeit von $V$ bei Eingabe $\langle w, c\rangle$ polynomiell in $|w|$ ist.

$L$ heißt dann polynomiell verifizierbar.

# Klasse $NP$

**Definition**

$NP$ ist die Klasse der Sprachen, die polynomiell verifizierbar sind.

$RS_{ent}, TSP_{ent}$ sind in $NP$.

**Satz**

$P$ ist eine Teilmenge von $NP$.

**Millenium-Problem**

Ist $P = NP$? (Clay Mathematics Institute)

# 3. Komplexität

## Nichtdeterministische Turingmaschinen

- Liefern alternative Beschreibung von $NP$.
- Erlaubt uns, die schwierigsten Probleme in $NP$ zu identifizieren.
  - $NP$-Vollständigkeit
- Geben der Turingmaschine die Möglichkeit, einen von mehreren möglichen Rechenschritten auszuwählen.
  - Nichtdeterminismus
- Realisierung: $\delta(q, a)$ ist Menge von Tripeln der Form $(q', b, D)$
- NTMs nicht realistisch, aber sehr nützlich für das Verständnis der Komplexität von Problemen

# Nichtdeterministische Turingmaschinen

**Definition**

Eine nichtdeterministische 1-Band-Turingmascihne (NTM) ist ein 7-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, wobei $Q, \Sigma, \Gamma$ endliche Menge sind.

Weiter gilt:

1. $Q$ ist die Zustandsmenge mit $q_0, q_{accept}, q_{reject}$ Elemente dieser Menge und $q_{accept} \neq q_{reject}$
2. $\Sigma$ ist das Eingabealphabet
3. $\Gamma$ ist das Bandalphabet
4. $\delta: Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ ist die Übergangsfunktion. $\mathcal{P}(M)$ bezeichnet hier die Potenzmenge von $M$.
5. Rechenschritt: einmalige Anwendung der Übergangsfunktion.

# Nichtdeterministische Turingmaschinen

$\mathrm{NTM} = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$. Berechnung bei Eingabe $w$.

1. Startet im Zustand $q_0$, mit Bandinhalt $w$ und Lesekopf auf dem ersten Zeichen von $w$.
2. wendet in jedem Rechenschritt Übergangsfunktion $\delta$ an,
3. bis Zustand $q_{accept}$ oder $q_{reject}$ erreicht wird,
4. falls einer dieser Zustände erreicht wird, sonst Endlosschleife.

# Nichtdeterministische Turingmaschinen

$\mathrm{NTM} = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ ist in Konfiguration $K = \alpha q \beta$, wenn gilt:

1. auf dem Band von $N$ steht $\alpha\beta$, gefolgt von Blanks,
2. $N$ befindet sich im Zustand $q$,
3. der Lesekopf von $N$ steht auf dem ersten Symbol von $\beta$.

# NTM – Rechenschritt

1. NTM $N$ in Konfiguration $K = \alpha q a \beta$.

2. $\delta(q, a) = \{(q_1, b_1, D_1), \ldots, (q_l, b_l, D_l)\}$.

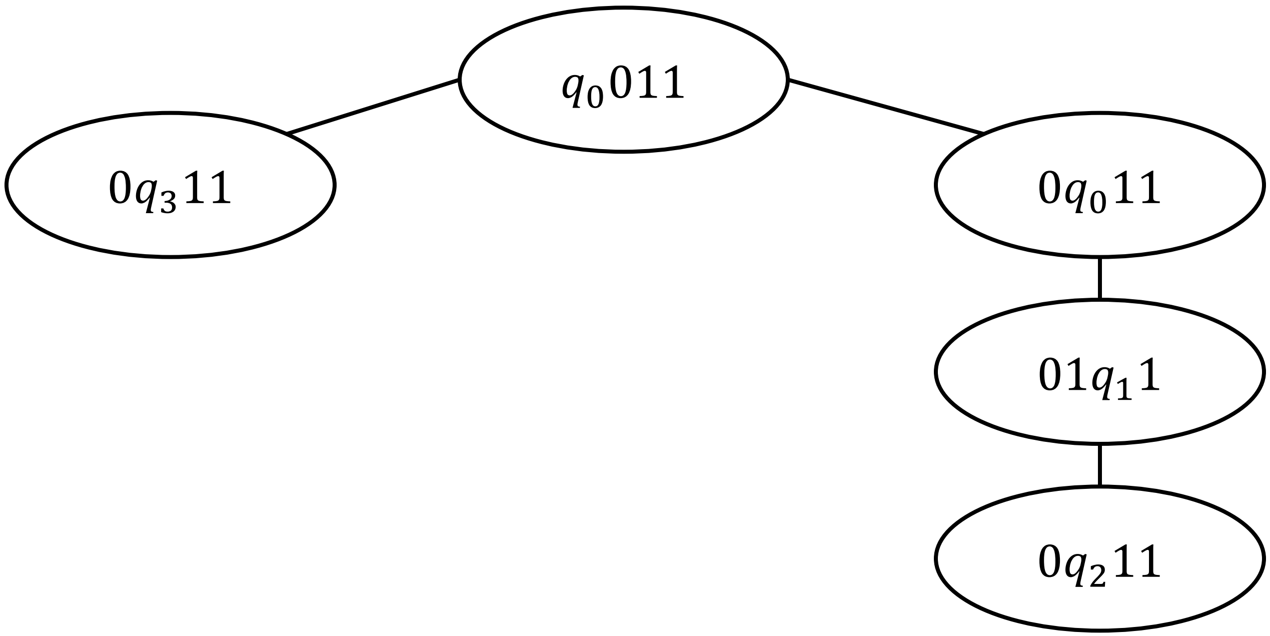
3. $N$ kann jeden durch ein Tripel $(q_i, b_i, D_i)$ aus $\delta(q, a)$ beschriebenen Rechenschritt ausführen.

# Berechnungen und Konfiguration

1. Ist die Eingabe für $N$ $w$, so heißt $q_0w$ Startkonfiguration.
2. $K = \alpha q \beta$ heißt akzeptierende Konfiguration, falls $q = q_{accept}$.
3. $K = \alpha q \beta$ heißt ablehnende Konfiguration, falls $q = q_{reject}$.
4. Berechnung von $N$ bei Eingabe $w$ führt zu Folge $K_1, K_2, \ldots$ von Konfigurationen.
5. Es gibt mehrere Berechnungen von $N$ bei Eingabe $w$, abhängig von den ausgewählten Rechenschritten.
6. Darstellung möglicher Berechnungen im sogenannten Berechnungsbaum.

# Nichtdeterministische Turingmaschinen

| $\boldsymbol{\delta}$ | **0** | **1** | $\sqcup$ |
|---|---|---|---|
| $q_0$ | $\{(q_0, 0, R), (q_3, 0, R)\}$ | $\{(q_1, 1, R)\}$ | $\{(q_3, \sqcup, R)\}$ |
| $q_1$ | $\{(q_0, 0, R), (q_3, 0, R)\}$ | $\{(q_2, 1, L)\}$ | $\{(q_3, \sqcup, R)\}$ |



# Akzeptieren und Entscheiden

**Definition**

Sei $N$ eine NTM. $N$ akzeptiert $w$, wenn es mindestens eine akzeptierende Berechnung von $N$ bei Eingabe $w$ gibt.

NTM $N$ hält bei Eingabe $w$, wenn alle Berechnungspfade von $N$ bei Eingabe $w$ endlich sind.

**Definition**

Die von einer NTM $N$ akzeptierte Sprache $L(N)$ ist definiert als

$$L(N) \coloneqq \{w \mid N \text{ akzeptiert } w\}$$

NTM $N$ akzeptiert die Sprache $L$, falls $L = L(N)$. $N$ entscheidet die von ihr akzeptierte Sprache $L(N)$, wenn $N$ immer hält.

# Laufzeit einer NTM

**Definition**

Sei $N$ eine NTM, die immer hält.

- Für $w$ ist $T_N(w)$ die maximale Anzahl von Rechenschritten in einer Berechnung von $N$ bei Eingabe $w$.
- Für eine natürliche Zahl $n$ ist $T_N(n) \coloneqq \max\{T_N(w) \mid w \text{ aus } \Sigma^{\leq n}\}$.
- Die Funktion $T_N$ heißt Zeitkomplexität oder Laufzeit der NTM $N$.
- $N$ haut Laufzeit $O\big(f(n)\big)$, wenn $T_N(n) = O\big(f(n)\big)$.

# Akzeptieren und Entscheiden

**Definition**

Sei $t$ eine monoton wachsende Funktion. Die Klasse $NTIME(t(n))$ ist dann definiert als

$$NTIME(t(n)) \coloneqq \left\{ \begin{array}{l} L \mid L \text{ ist eine Sprache, die von einer NTM} \\ \qquad \text{mit Laufzeit } O(t(n)) \text{ entschieden wird.} \end{array} \right\}$$

**Satz**

$NP$ ist die Klasse der Sprachen, die von einer nichtdeterministischen Turingmaschine mit polynomieller Laufzeit entschieden werden, d.h.,

$$NP = \bigcup_k NTIME(n^k)$$

# Simulation einer NTM durch eine DTM

**Satz**

Sei $t$ eine monoton wachsende Funktion mit $t(n) \geq n$ für alle natürlichen Zahlen $n$. Für jede NTM mit Laufzeit $t(n)$ gibt es eine DTM mit Laufzeit $2^{O(t(n))}$, die dieselbe Sprache entscheidet.