Vorlesung 19
Zusammenfassung

Wdh.: Was tun mit NP-schweren Problemen?

Viele praxisrelevante Optimierungsprobleme sind NP-schwer, z.B. das Bin Packing Problem (BPP), das Rucksack Problem (KP) und das Traveling Salesperson Problem (TSP).

In der Praxis müssen die Probleme dennoch gelöst werden. Verschiedene Strategien können hier zum Erfolg führen; die wichtigsten sind:

- Approximationsalgorithmen
- Ausnutzen der Eingabestruktur durch spezielle Algorithmen
- Parametrisierte Algorithmen
- Randomisierte Algorithmen
- Heuristiken (ohne irgendwelche Performanzgarantien)

Wdh.: Approximationsalgorithmen

Sei Π ein Optimierungsproblem. Für eine Instanz I von Π bezeichnen wir den optimalen Zielfunktionswert mit $opt(I)$.

- Ein α-Approximationsalgorithmus, $\alpha > 1$, für ein *Minimierungsproblem* Π berechnet für jede Instanz I von Π eine zulässige Lösung mit Zielfunktionswert höchstens $\alpha \cdot opt(I)$.
- Ein α-Approximationsalgorithmus, $\alpha < 1$, für ein *Maximierungsproblem* Π berechnet für jede Instanz I von Π eine zulässige Lösung mit Zielfunktionswert mindestens $\alpha \cdot opt(I)$.

Die Zahl α wird auch als Approximationsfaktor oder Approximationsgüte bezeichnet.

Approximierbarkeit von einigen Problemen

	obere Schranke	untere Schranke	Bemerkung
BPP	2	$\frac{3}{2}$	$(1+\varepsilon)\text{OPT}+1$ möglich
TSP	–	∞	
TSP metrisch	$\frac{3}{2} - \epsilon$ für ein $\epsilon > 0$	$\frac{123}{122}$	
TSP euklidisch	$1+\varepsilon$ für alle $\epsilon > 0$	–	
KP	$1+\varepsilon$ für alle $\epsilon > 0$	–	

Die unteren Schranken gelten für effiziente Approximierbarkeit unter der Annahme $\text{P} \neq \text{NP}$.

Rückblick

In BuK haben wir uns mit den

Grenzen der (effizienten) Berechenbarkeit

beschäftigt.

Teil I
Grundlagen

Vorlesung 1
Einführung

Vorlesung 2
Berechnungsprobleme und Turingmaschinen

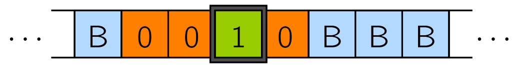
Wdh.: Kodierung von Berechnungsprobleme

3 mögliche formale Definitionen.

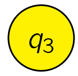
Als Relation:	Als Funktion	Als Sprache
▶ Primfaktor:		
$(110, 11) \in R$ $(101, 11) \notin R$ $(00110, 11) \notin R$		
▶ Multiplikation		
$(11\#10, 110) \in R$ $(11\#10, 11) \notin R$ $(1\#1\#0, 110) \notin R$	$f(11\#10) = 110$ $(f(1\#1\#0) = \perp)$	
▶ Wörter die auf 1 enden.		
$(11, 1) \in R$ $(110, 1) \notin R$ $(10, 0) \in R$	$f(11) = 1$ $f(110) = 0$	$11 \in L$ $110 \notin L$

Wdh.: Turingmaschinen

Anschauliche Definition:



δ	0	1	B
q_1	$(q_1, 1, L)$	$(q_2, 1, R)$	(q_1, B, N)
q_2	(q_3, B, R)	$(q_1, 0, L)$	(q_3, B, R)
q_3	$(q_2, 0, N)$	$(q_2, 0, R)$	(q_3, B, R)



Formale Definition:

Eine Turingmaschine ist ein 7-Tupel $(Q, \Sigma, \Gamma, B, q_0, \bar{q}, \delta)$, wobei

- Q, Σ, Γ endliche Mengen sind,
- $\Sigma \subseteq \Gamma$,
- $B \in \Gamma \setminus \Sigma$,
- $q_0, \bar{q} \in Q$ und
- $\delta \colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$.

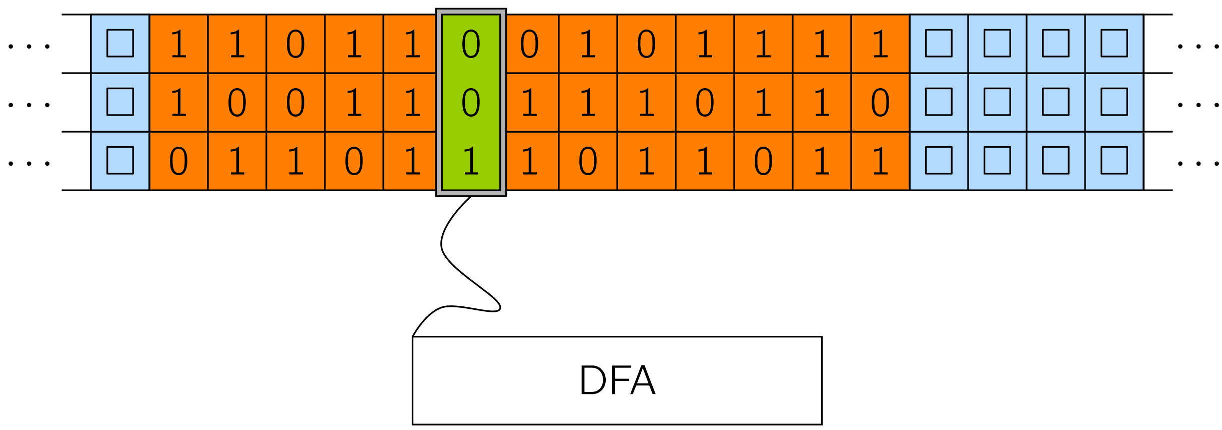
Wdh.: TM-Techniken

- Speicher im Zustandsraum:

$$Q_{\mathrm{neu}} := Q \times \Gamma^k$$

- Mehrspurmaschinen:

$$\Gamma_{\mathrm{neu}} := \Gamma \cup \Gamma^k$$



- Schleifen, Variablen, Felder (Arrays), Unterprogramme

Vorlesung 3
Mehrband-Turingmaschinen und die universelle Turingmaschine

Wdh.: k-Band- vs 1-Band-TM

Satz

Eine k-Band-TM M, die mit Rechenzeit $t(n)$ und Platz $s(n)$ auskommt, kann von einer (1-Band-)TM M' mit Zeitbedarf $O(t^2(n))$ und Platzbedarf $O(s(n))$ simuliert werden.

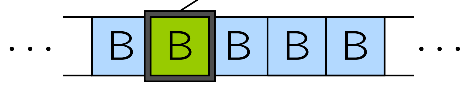
…	B	*	*	*	*	A	**r**	b	e	i	t	s	b	a	n	d	1	B	B	B	…

…	B	**A**	r	b	e	i	t	s	b	a	n	d	2	*	*	*	*	B	B	B	…

Simuliert durch

…	B	*****	*	*	*	A	r	b	e	i	t	s	b	a	n	d	1	B	B	B	…
…	B	**B**	B	B	B	B	#	B	B	B	B	B	B	B	B	B	B	B	B	B	…
…	B	**A**	r	b	e	i	t	s	b	a	n	d	2	*	*	*	*	B	B	B	…
…	B	**#**	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	…

Wdh.: Gödelnummer $\langle M \rangle$



$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, B\}$
$Q = \{q_1, q_2, q_3\}$

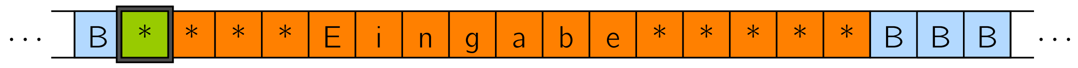
Start	Ende	Blank
q_1	q_2	B

δ	0	1	B
q_1	(q_1, B, R)	(q_3, B, R)	(q_2, B, N)
q_3	$(q_1, 1, R)$	$(q_2, 0, R)$	(q_1, B, L)

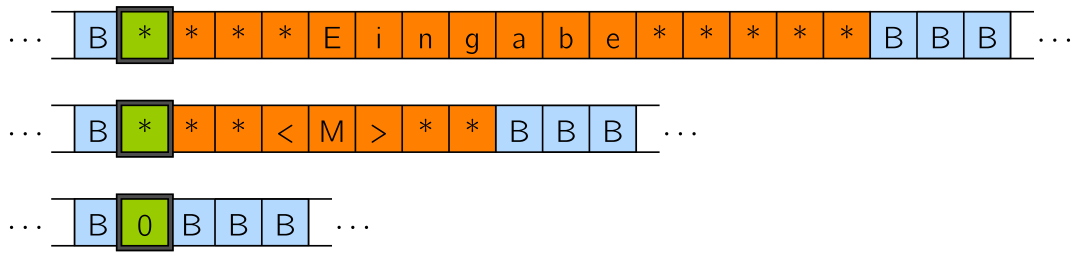
111 0101010001000 11 0100100010001000 11 010001001000100 11
00010101001000 11 000100100101000 11 000100010100010 111

Wdh.: Universelle TM

simulierte Turingmaschine M

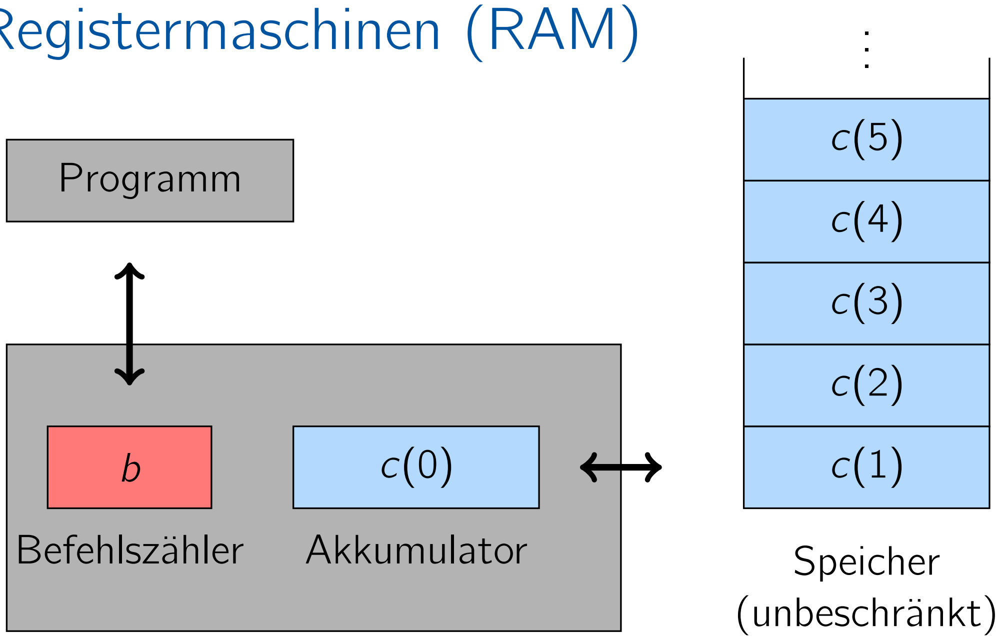


Initialisierung der universellen Maschine U



Vorlesung 4
Registermaschine (RAM), Church-Turing-These

Wdh.: Registermaschinen (RAM)



Befehlssatz:

LOAD, STORE, ADD, SUB, MULT, DIV
INDLOAD, INDSTORE, INDADD, INDSUB, INDMULT, INDDIV
CLOAD, CADD, CSUB, CMULT, CDIV
GOTO, IF c(0)?x THEN GOTO j (wobei ? aus {=,<,<=,>,>=} ist),
END

Wdh.: RAM vs. TM

Satz (RAM $\to$ TM)

Für jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte RAM R gibt es ein Polynom q und zu diesem eine $O\big(q(n + t(n))\big)$-TM M, die R simuliert.

Satz (TM $\to$ RAM)

Jede $t(n)$-zeitbeschränkte TM kann durch eine RAM simuliert werden, die zeitbeschränkt ist durch

- $O(t(n) + n)$ *im uniformen Kostenmaß und*
- $O((t(n) + n) \cdot \log(t(n) + n))$ *im logarithmischen Kostenmaß.*

Wdh.: Die Church-Turing-These

Kein jemals bisher vorgeschlagenes „vernünftiges“ Rechnermodell hat eine größere Mächtigkeit als die TM.

Church-Turing-These

Die Klassen der TM-berechenbaren (partiellen) Funktionen und TM-entscheidbaren Sprachen stimmen mit den Klassen der “intuitiv berechenbaren” (partiellen) Funktionen bzw. “intuitiv entscheidbaren” Sprachen überein.

Wir sprechen deshalb nicht mehr von TM-berechenbaren (partiellen) Funktionen oder TM-entscheidbaren Sprachen, sondern allgemein von berechenbaren (partiellen) Funktionen bzw. entscheidbaren Sprachen.

Teil II
Berechenbarkeit

Vorlesung 5
Unentscheidbare Probleme: Diagonalisierung

Wdh.: Abzählbarkeit

Definition (Abzählbare Menge)

Eine Menge M heißt abzählbar, wenn sie leer ist oder wenn es eine surjektive Funktion $c\colon \mathbb{N} \to M$ gibt.

Abzählbare Mengen: endlichen Mengen, $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$, $\{0,1\}^*$, Menge der Gödelnummern, die Menge der endlichen Teilmengen von $\mathbb{N}$.

Satz

Die Menge $\mathcal{P}(\mathbb{N})$ ist überabzählbar.

Überabzählbare Mengen: $\mathbb{R}$, $\mathcal{P}(\mathbb{N})$, $\mathcal{P}(\{0,1\}^*)$, Menge der Berechnungsprobleme.

Schlussfolgerung: Es gibt nicht-berechenbare Probleme.

Wdh.: Unentscheidbarkeit der Diagonalsprache

Die Diagonalsprache:

$$\begin{aligned} D &= \{\, w \in \{0,1\}^* \mid w = w_i \text{ und } M_i \text{ akzeptiert } w \text{ nicht}\} \\ &= \{\langle M\rangle \mid M \text{ akteptiert } \langle M\rangle \text{ nicht}\}. \end{aligned}$$

Satz

Die Diagonalsprache D ist unentscheidbar (= nicht entscheidbar).

Beweisansatz: Diagonalisierung

Wdh.: Unentscheidbarkeit der Diagonalsprache

$$A_{i,j} = \begin{cases} 1 & \text{falls } M_i \text{ das Wort } w_j \text{ akzeptiert} \\ 0 & \text{sonst} \end{cases}$$

Beispiel:

	w_0	w_1	w_2	w_3	w_4	
M_0	**0**	1	1	0	1	$\cdots$
M_1	1	**0**	1	0	1	$\cdots$
M_2	0	0	**1**	0	1	$\cdots$
M_3	0	1	1	**1**	0	$\cdots$
M_4	0	1	0	0	**0**	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		

Die Diagonalsprache lässt sich auf der Diagonale der Matrix ablesen. Es ist

$$D = \{w_i \mid A_{i,i} = 0\} .$$

Vorlesung 6

Unentscheidbarkeit des Halteproblems: Unterprogrammtechnik

Wdh.: Unentscheidbare Probleme

Die Diagonalsprache:

$$D \;=\; \{\langle M\rangle \mid M \text{ akzeptiert } \langle M\rangle \text{ nicht}\}$$

Das Diagonalsprachenkomplement:

$$\overline{D} \;=\; \{\langle M\rangle \mid M \text{ akzeptiert } \langle M\rangle\}$$

Das Halteproblem:

$$H \;=\; \{\langle M\rangle w \mid M \text{ hält auf } w\}$$

Das spezielle Halteproblem:

$$H_\epsilon \;=\; \{\langle M\rangle \mid M \text{ hält auf Eingabe } \epsilon\}$$

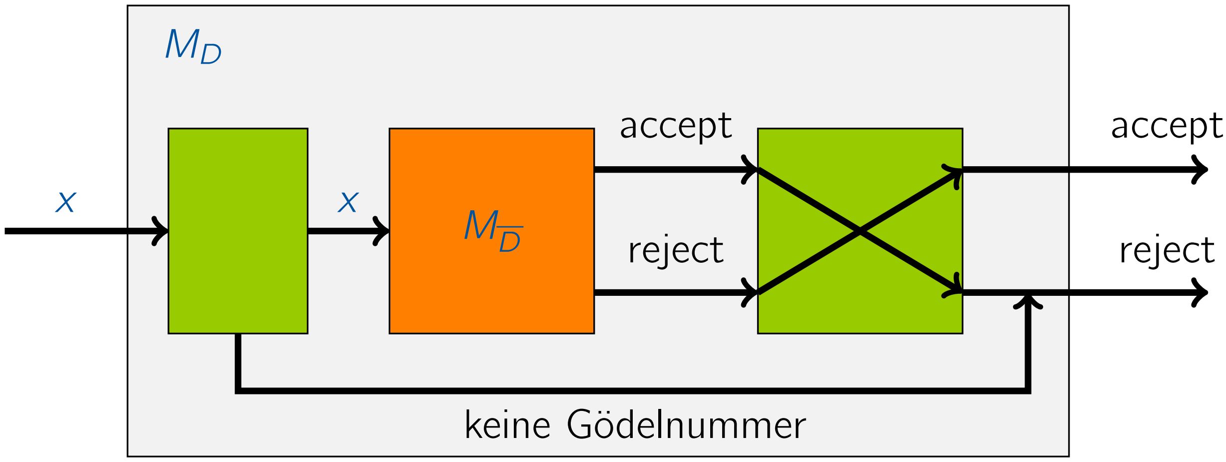
Wdh.: Beweise durch Unterprogrammtechnik

D ist unentscheidbar auf Grund eines Diagonalisierungs-Argumentes.

Die Argumentationskette war:

$$\begin{array}{ccc}
D \text{ ist unentscheidbar} & & M_D \\
\Downarrow & & \uparrow \\
\overline{D} \text{ ist unentscheidbar} & & M_{\overline{D}} \\
\Downarrow & & \uparrow \\
H \text{ ist unentscheidbar} & & M_H \\
\Downarrow & & \uparrow \\
H_\epsilon \text{ ist unentscheidbar} & & M_{H_\epsilon}
\end{array}$$

Wdh.: Unentscheidbarkeit des Komplements der Diagonalsprache



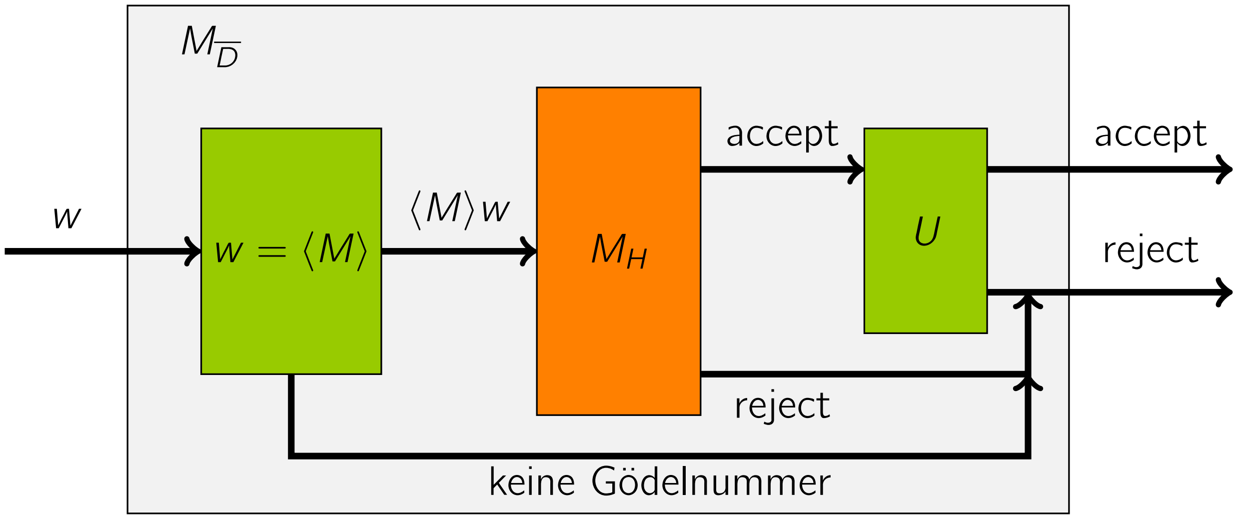
Wdh.: Beweise durch Unterprogrammtechnik

D ist unentscheidbar auf Grund eines Diagonalisierungs-Argumentes.

Die Argumentationskette war:

$$\begin{array}{cc}
D \text{ ist unentscheidbar} & M_D \\
\Downarrow & \rotatebox{90}{$\rightsquigarrow$} \\
\overline{D} \text{ ist unentscheidbar} & M_{\overline{D}} \\
\Downarrow & \rotatebox{90}{$\rightsquigarrow$} \\
H \text{ ist unentscheidbar} & M_H \\
\Downarrow & \rotatebox{90}{$\rightsquigarrow$} \\
H_\epsilon \text{ ist unentscheidbar} & M_{H_\epsilon}
\end{array}$$

Wdh.: Unentscheidbarkeit des Halteproblems



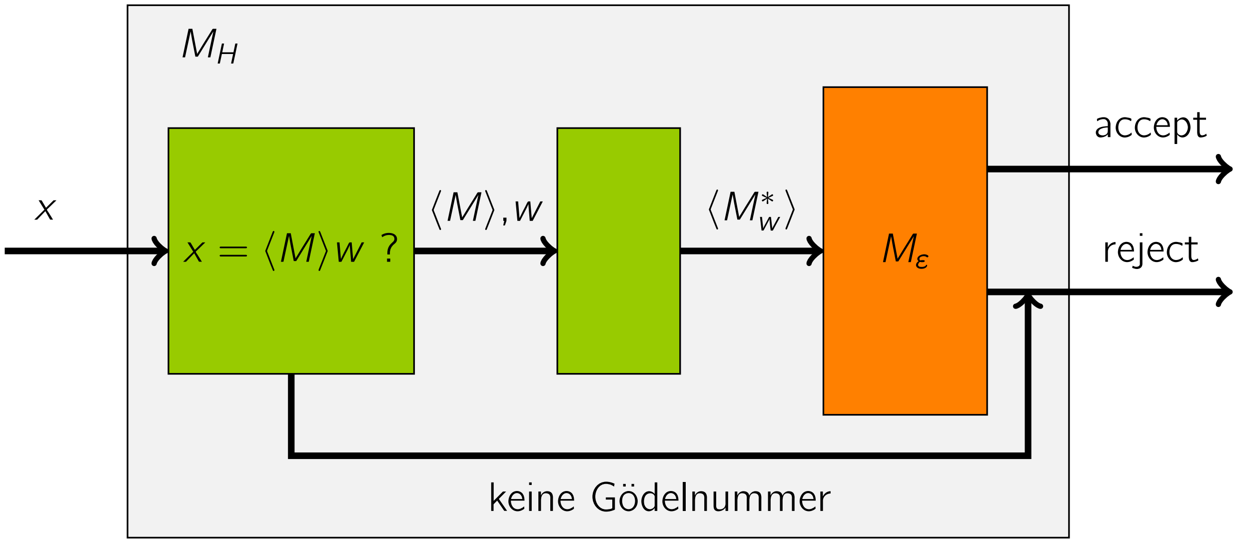
Wdh.: Beweise durch Unterprogrammtechnik

D ist unentscheidbar auf Grund eines Diagonalisierungs-Argumentes.

Die Argumentationskette war:

$$\begin{array}{cc}
D \text{ ist unentscheidbar} & M_D \\
\Downarrow & \uparrow \\
\overline{D} \text{ ist unentscheidbar} & M_{\overline{D}} \\
\Downarrow & \uparrow \\
H \text{ ist unentscheidbar} & M_H \\
\Downarrow & \uparrow \\
H_\epsilon \text{ ist unentscheidbar} & M_{H_\epsilon}
\end{array}$$

Wdh.: Unentscheidbarkeit des speziellen Halteproblems



Vorlesung 7
Der Satz von Rice

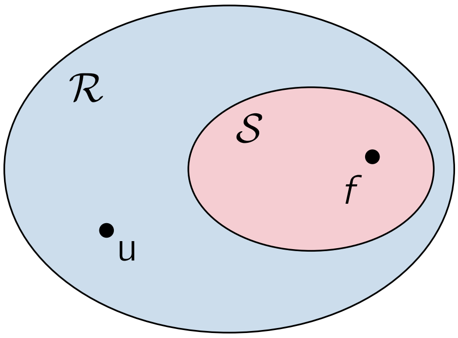
Wdh.: Der Satz von Rice

Satz

Sei $\mathcal{R}$ die Menge der von TMen berechenbaren partiellen Funktionen und $\mathcal{S}$ eine Teilmenge von $\mathcal{R}$ mit $\emptyset \subsetneq \mathcal{S} \subsetneq \mathcal{R}$. Dann ist die Sprache

$$L(\mathcal{S}) = \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } \mathcal{S}\}$$

unentscheidbar.



Wdh.: Satz von Rice – Beispiele

Beispiel

- Sei $\mathcal{S} = \{f_M \mid \forall w \in \{0,1\}^* \colon f_M(w) \neq \perp\}$.
- Dann ist

$$\begin{aligned} L(\mathcal{S}) &= \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } \mathcal{S}\} \\ &= \{\langle M \rangle \mid M \text{ hält auf jeder Eingabe}\} \end{aligned}$$

- Diese Sprache ist auch als das *allgemeine Halteproblem* H_{all} bekannt.
- Gemäß Satz von Rice ist H_{all} unentscheidbar.

Beispiel

- Sei $H_{42} = \{\langle M \rangle \mid$
  Auf jeder Eingabe hält M nach höchstens 42 Schritten$\}$.
- Über diese Sprache sagt der Satz von Rice nichts aus!
- H_{42} ist entscheidbar.

Vorlesung 8
Rekursive Aufzählbarkeit

Wdh.: Semi-Entscheidbarkeit

Eine Sprache L wird von einer TM M erkannt, wenn

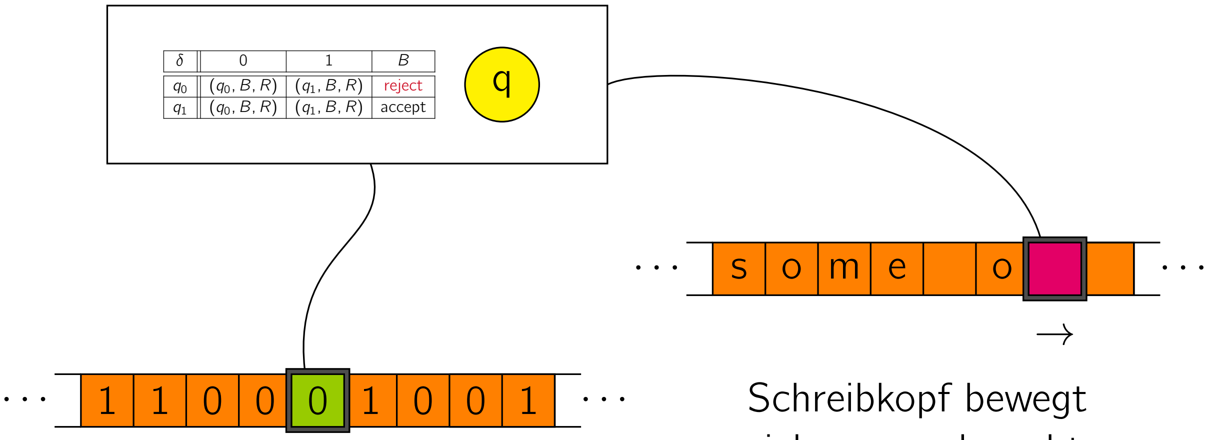
- M jedes Wort aus L akzeptiert, und
- M kein Wort akzeptiert, das nicht in L enthalten ist.

Es ist $L(M)$ die von M erkannte Sprache.

Definition
Eine Sprache L, für die eine TM existiert, die L erkennt, wird als semi-entscheidbar bezeichnet.

Beobachtung
Das Halteproblem ist semi-entscheidbar.

Wdh.: Aufzähler



Schreibkopf bewegt sich nur nach rechts

Wdh.: rekursiv aufzählbar = semi-entscheidbar

Satz

Eine Sprache L ist genau dann semi-entscheidbar, wenn sie rekursiv aufzählbar ist.

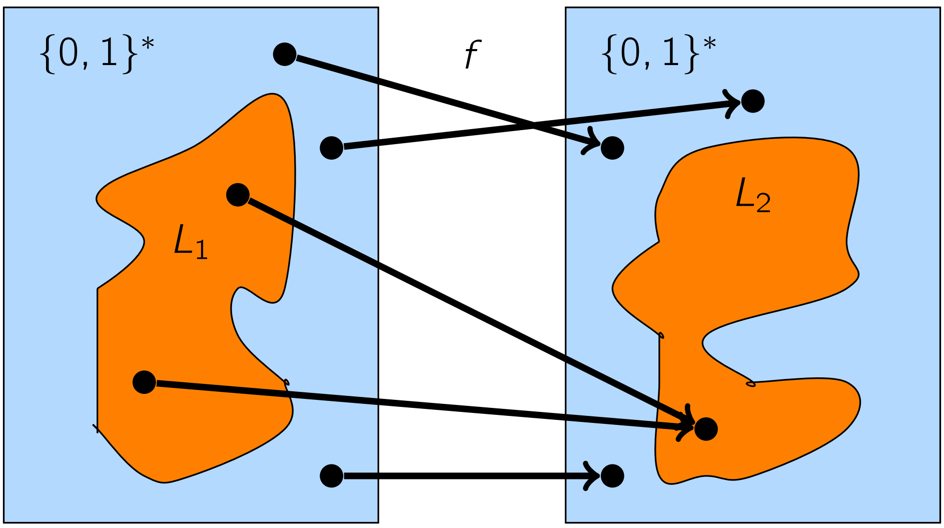
Vorlesung 9
Allgemeines Halteproblem und Hilberts 10. Problem

Wdh.: Reduktionen

Definition

Es seien L_1 und L_2 Sprachen über einem Alphabet Σ. Dann heißt L_1 auf L_2 reduzierbar, Notation $L_1 \leq L_2$, wenn es eine berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt

$$x \in L_1 \;\Leftrightarrow\; f(x) \in L_2 \; .$$



Wdh.: Komplexität des allgemeines Halteproblem

Satz

Weder $\overline{H}_{all}$ noch H_{all} sind semi-entscheidbar.

Wdh.: Hilberts zehntes Problem

Hilberts zehntes Problem

Beschreibe einen Algorithmus, der entscheidet, ob ein gegebenes Polynom mit ganzzahligen Koeffizienten eine ganzzahlige Nullstelle hat.

Die diesem Entscheidungsproblem zugrundeliegende Sprache ist

$$N = \{ p \mid p \text{ ist ein Polynom mit einer ganzzahligen Nullstelle} \} .$$

Satz von Matijasevič (1970)

Das Problem, ob ein ganzzahliges Polynom eine ganzzahlige Nullstelle hat, ist unentscheidbar.

Vorlesung 10
Das Postsche Korrespondenzproblem

Wdh.: Das Postsche Korrespondenzproblem

Das Postsche Korrespondenzproblem (PKP) ist eine Art Puzzle aus Dominos.
Eine Instanz ist zum Beispiel

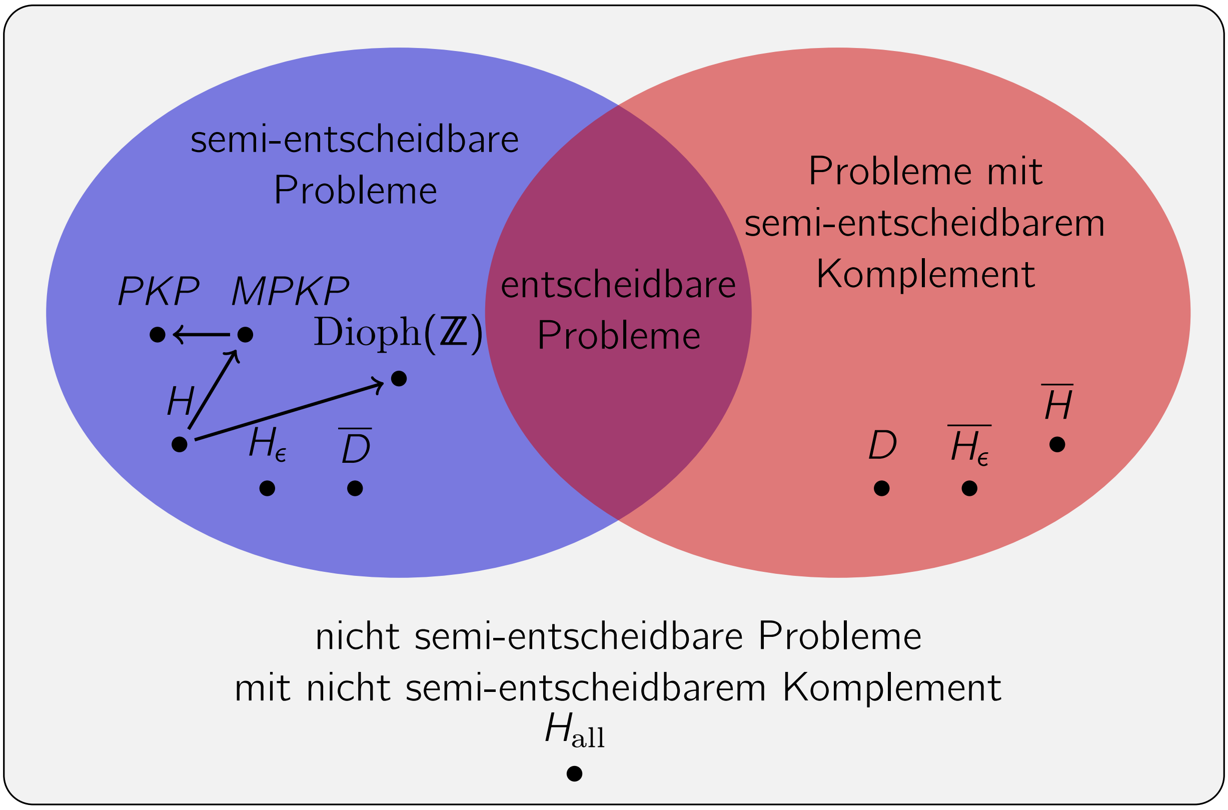
$$K = \left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\} .$$

Eine Lösung ist

$$\left[\frac{a}{ab}\right] \left[\frac{b}{ca}\right] \left[\frac{ca}{a}\right] \left[\frac{a}{ab}\right] \left[\frac{abc}{c}\right] .$$

Satz
Das PKP ist nicht entscheidbar.

Wdh.: Berechenbarkeitslandschaft



Vorlesung 11

WHILE-Programme

Wdh.: Turing-mächtige Programmiersprachen

Definition

Eine Programmiersprache wird als Turing-mächtig bezeichnet, wenn jede Funktion, die durch eine TM berechnet werden kann, auch durch ein Programm in dieser Programmiersprache berechnet werden kann.

Satz

Die Programmiersprache WHILE ist Turing-mächtig.

Vorlesung 12

LOOP-Programme

Whd.: Die Ackermann-Funktion

Definition

Die Ackermannfunktion $A\colon \mathbb{N}^2 \to \mathbb{N}$ ist folgendermaßen definert:

$$
\begin{array}{lcll}
A(0,n) & = & n+1 & \text{für } n \geq 0 \\
A(m+1,0) & = & A(m,1) & \text{für } m \geq 0 \\
A(m+1,n+1) & = & A(m,A(m+1,n)) & \text{für } m,n \geq 0
\end{array}
$$

Wdh.: LOOP vs WHILE

Lemma

Für jedes LOOP-Programm P gibt es eine natürliche Zahl m, so dass für alle $n \in \mathbb{N}$ gilt: $F_P(n) < A(m,n)$.

Satz

Die Ackermannfunktion ist nicht LOOP-berechenbar.

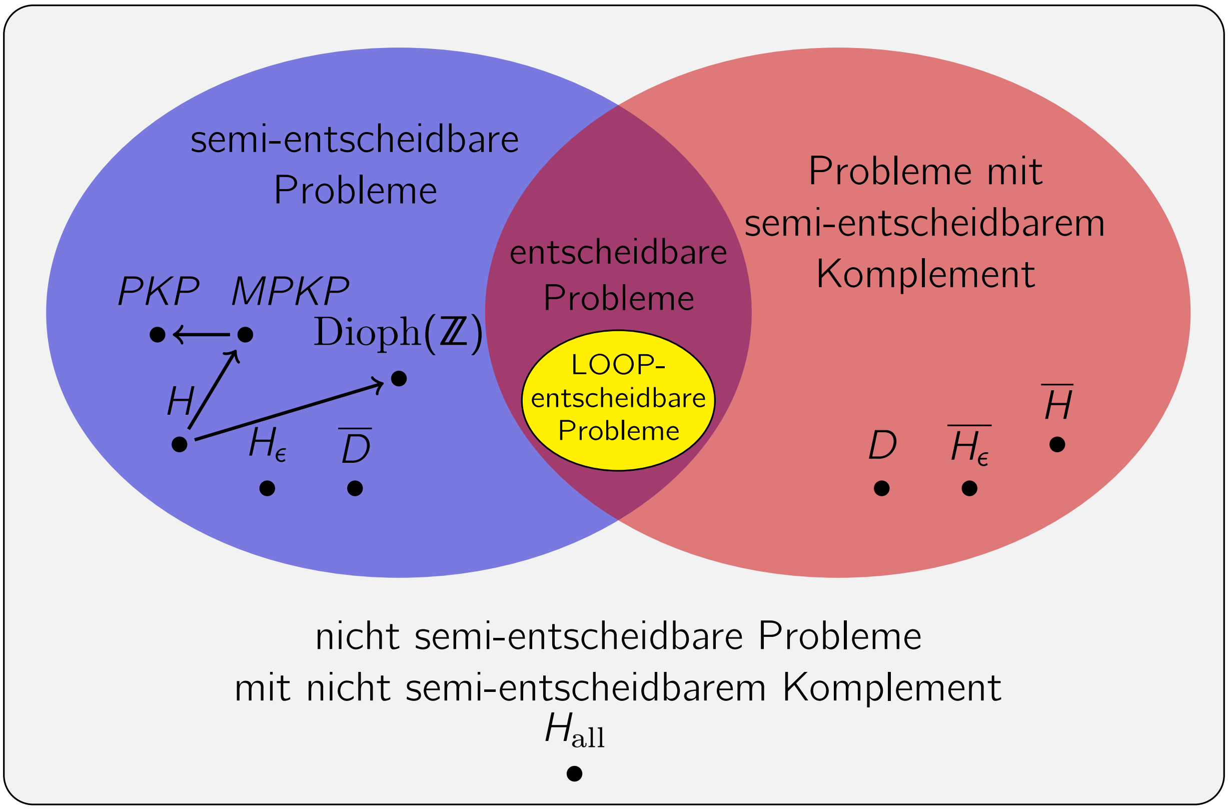
Korollar

Die Klasse der LOOP-berechenbaren Funktionen ist eine echte Teilmenge der berechenbaren (totalen) Funktionen.

Bemerkung

Mit Hilfe eines Diagonalisierungsarguments lässt sich auch beweisen, dass es entscheidbare Sprachen gibt, die nicht LOOP-entscheidbar sind.

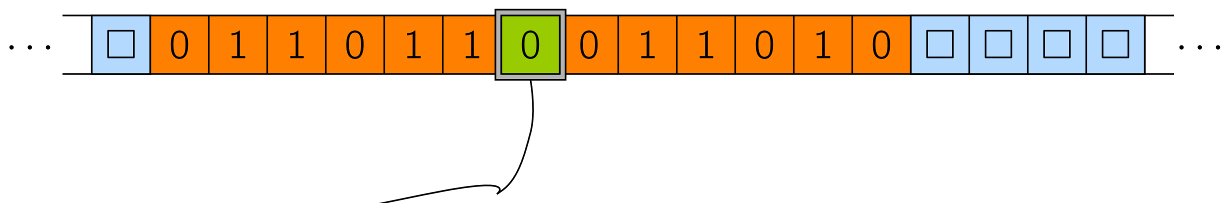
Wdh.: Berechenbarkeitslandschaft



Teil III
Komplexität

Vorlesung 13
Die Komplexitätsklassen P und NP

Wdh.: Nichtdeterministische Turingmaschine (NTM)



δ	0	1	B
q_0	$\{(q_0, B, R), (q_1, B, R)\}$	{reject}	{reject}
q_1	{reject}	$\{(q_1, B, R), (q_2, B, R)\}$	{reject}
q_2	{reject}	{reject}	{accept}

Wdh.: Komplexitätsklassen

Definition (Komplexitätsklassen)

- P ist die Klasse der Entscheidungsprobleme, für die es einen Polynomialzeitalgorithmus gibt.
- NP ist die Klasse der Entscheidungsprobleme, die durch eine NTM M erkannt werden, deren worst case Laufzeit $t_M(n)$ polynomiell beschränkt ist.
- EXPTIME ist die Klasse der Entscheidungsprobleme L, für die es ein Polynom q gibt, so dass sich L auf einer DTM mit Laufzeitschranke $2^{q(n)}$ berechnen lässt.

Satz

P $\subseteq$ NP $\subseteq$ EXPTIME

Vorlesung 14

Die Klasse NP und polynomielle Reduktionen

Wdh.: Optimierungs- versus Entscheidungsproblem

Mit Hilfe eines Algorithmus, der ein Optimierungsproblem löst, kann man die Entscheidungsvariante lösen.

Umgekehrt gilt:

Satz

Wenn die Entscheidungsvariante von KP in polynomieller Zeit lösbar ist, dann auch die Optimierungsvariante.

Dieser Satz gilt auch für TSP und BPP.

Wdh.: Alternative Charakterisierung der Klasse NP

Satz

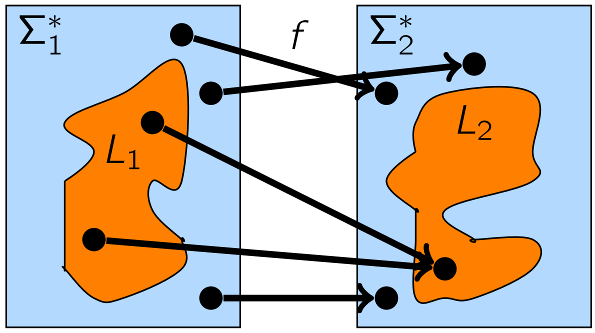
Eine Sprache $L \subseteq \Sigma^*$ ist genau dann in NP, wenn es einen Polynomialzeitalgorithmus V (einen sogenannten Verifizierer) und ein Polynom p mit der folgenden Eigenschaft gibt:

$$x \in L \quad \Leftrightarrow \quad \exists y \in \{0,1\}^*, |y| \leq p(|x|) : V \text{ akzeptiert } y\#x.$$

Wdh.: Polynomielle Reduktionen

Definition (Polynomielle Reduktion)

L_1 und L_2 seien zwei Sprachen über Σ_1 bzw. Σ_2. Dann heißt L_1 polynomiell reduzierbar auf L_2, wenn es eine Reduktion von L_1 nach L_2 gibt, die in polynomieller Zeit berechenbar ist. Wir schreiben $L_1 \leq_p L_2$.



Lemma

Angenommen $L_1 \leq_p L_2$, dann gilt: $L_2 \in P \Rightarrow L_1 \in P$.

Satz

COLORING $\leq_p$ SAT.

Wdh.: Das Erfüllbarkeitsproblem – SAT

Problem (Erfüllbarkeitsproblem / Satisfiability – SAT)

Eingabe: Aussagenlogische Formel φ in KNF
Frage: Gibt es eine erfüllende Belegung für φ?

SAT-Beispiel 1:

$$\varphi = (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee x_4)$$

φ ist erfüllbar, denn $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ ist eine erfüllende Belegung.

Vorlesung 15
NP-Vollständigkeit

Wdh.: NP-Vollständigkeit

Definition (NP-vollständig)

Ein Problem L heißt NP-vollständig (engl. NP-complete), falls gilt

1. $L \in \mathrm{NP}$, und
2. L ist NP-schwer.

Die Klasse der NP-vollständigen Probleme wird mit NPC bezeichnet.

Satz (Cook und Levin)

SAT ist NP-vollständig.

Lemma

3-SAT $\in$ NP und SAT $\leq_p$ 3-SAT.

Korollar

3-SAT ist NP-vollständig.

Vorlesung 16

NP-Vollständigkeit ausgewählter Zahlprobleme

Wdh.: NP-Vollständigkeit von Zahlproblemen

Satz
SUBSET-SUM ist NP-vollständig.

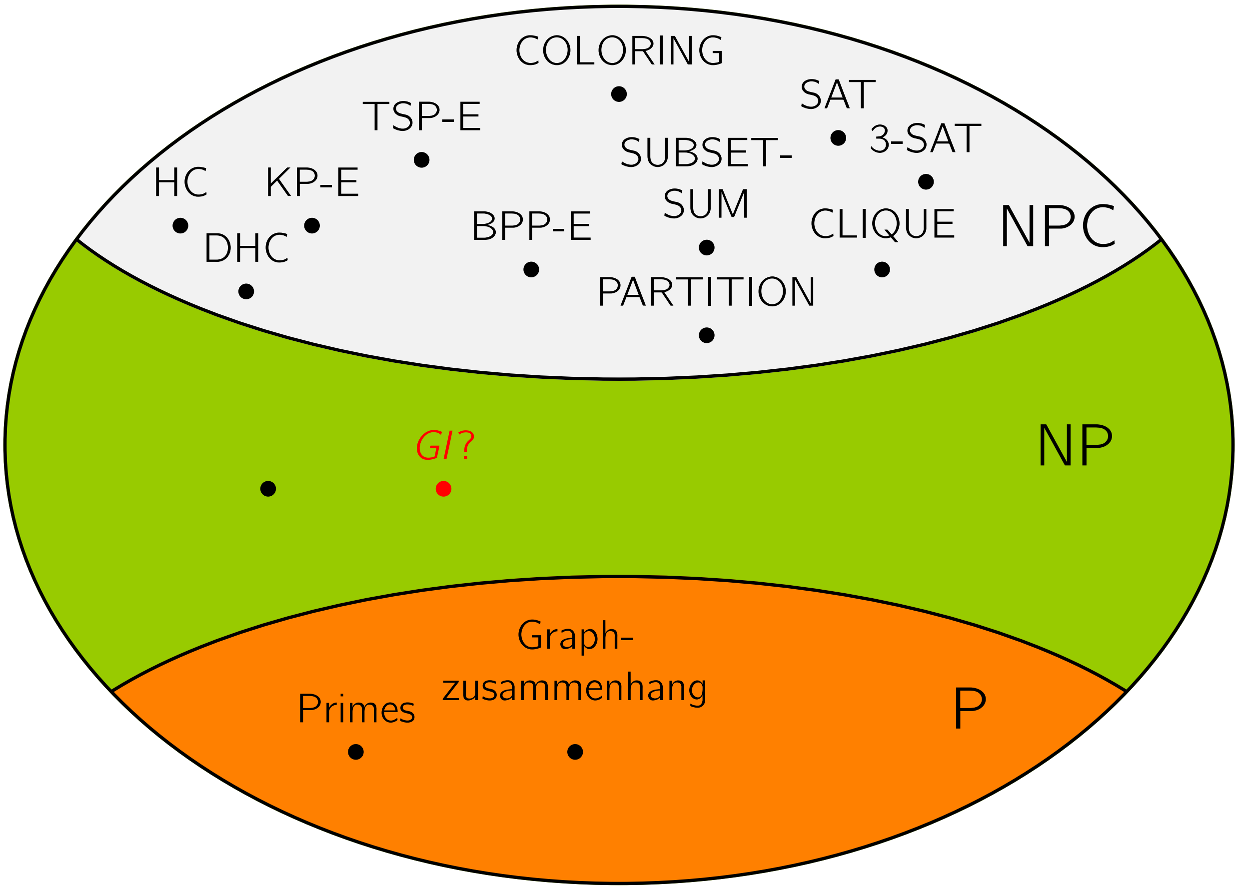
Satz
PARTITION ist NP-vollständig.

Satz
BPP-E ist NP-vollständig.

Satz
KP-E ist NP-vollständig.

Vorlesung 17
NP-Vollständigkeit ausgewählter Graphprobleme

Wdh.: Die Komplexitätslandschaft



Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

Vorlesung 18
Ausblick: Algorithmen und Komplexität

Vorlesung 19
Zusammenfassung

Berechenbarkeits- und Komplexitätslandschaft