Grundlagen der Komplexitätstheorie

Einführung für das Seminar
Komplexität und Kryptologie

Sebastian Kuhnert

23. und 30. April 2008

Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Übersicht

1. Zeit- und Platzklassen
2. Schaltkreise
3. Reduktionen
   Many-One-Reduktionen
   Turing-Reduktionen und Orakel

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Turingmaschinen als Berechnungsmodell

Erweiterte Church'sche These

Alle (hinreichend mächtigen) Berechnungsmodelle können sich gegenseitig simulieren. Dabei treten nur polynomielle Laufzeitunterschiede auf.

- Sprachprobleme:
  - Gegeben: $A \subseteq \Sigma^*$
  - Wir betrachten Verfahren f, die für $w \in \Sigma^*$ entscheiden, ob $w \in A$ gilt
  - $f : \Sigma^* \to \{0, 1\}$ mit $f(w) = 1 \Leftrightarrow w \in A$
- Turingmaschinen:
  - Bänder, Lese-/Schreibköpfe, Zustände
  - Deterministisch (DTM): Eindeutige Folgekonfiguration
  - Nichtdeterministisch (NTM): Menge von Folgekonfigurationen
  - Endzustände, akzeptierte Sprache $L(M)$

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Ressourcenverbrauch von Turingmaschinen

- Zeitverbrauch: Anzahl der Berechnungsschritte
  $\text{time}_M : \Sigma^* \to \mathbb{N}$ (einzelnes Wort)
  $\text{time}_M : \mathbb{N} \to \mathbb{N}$, $\text{time}_M(n) := \max_{x \in \Sigma^n} \text{time}_M(x)$
- Platzverbrauch: Anzahl der verwendeten Bandfelder
  $\text{space}_M : \Sigma^* \to \mathbb{N}$, $\text{space}_M : \mathbb{N} \to \mathbb{N}$

Sei $f : \mathbb{N} \to \mathbb{N}$.

$$\text{DTIME}(f) := \left\{ A \subseteq \Sigma^* \;\middle|\; \exists M \text{ DTM} : L(M) = A \wedge \text{time}_M \in \mathcal{O}(f) \right\}$$

$$\text{DSPACE}(f) := \left\{ A \subseteq \Sigma^* \;\middle|\; \exists M \text{ DTM} : L(M) = A \wedge \text{space}_M \in \mathcal{O}(f) \right\}$$

$$\text{NTIME}(f) := \left\{ A \subseteq \Sigma^* \;\middle|\; \exists M \text{ NTM} : L(M) = A \wedge \text{time}_M \in \mathcal{O}(f) \right\}$$

$$\text{NSPACE}(f) := \left\{ A \subseteq \Sigma^* \;\middle|\; \exists M \text{ NTM} : L(M) = A \wedge \text{space}_M \in \mathcal{O}(f) \right\}$$

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Wichtige Zeit- und Platzklassen

Zeitklassen:

$$\mathrm{P} = \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(n^k)$$
$$\mathrm{NP} = \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k)$$
$$\mathrm{coNP} = \left\{A \subseteq \Sigma^* \,\middle|\, \overline{A} \in \mathrm{NP}\right\}$$
$$\mathrm{E} = \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(2^{kn})$$
$$\mathrm{NE} = \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(2^{kn})$$
$$\mathrm{EXP} = \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(2^{n^k})$$
$$\mathrm{NEXP} = \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(2^{n^k})$$

Platzklassen:

$$\mathrm{L} = \mathrm{DSPACE}(\log(n))$$
$$\mathrm{NL} = \mathrm{NSPACE}(\log(n))$$
$$\mathrm{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathrm{DSPACE}(n^k)$$

Fakten über Zeit- und Platzklassen

- $\mathrm{DTIME}(f) \subseteq \mathrm{NTIME}(f)$
  $\mathrm{DSPACE}(f) \subseteq \mathrm{NSPACE}(f)$ (Spezialfall)
- $\mathrm{DTIME}(f) \subseteq \mathrm{DSPACE}(f)$
  $\mathrm{NTIME}(f) \subseteq \mathrm{NSPACE}(f)$ (begrenzte Laufweite)
- $\mathrm{NTIME}(f) \subseteq \mathrm{DSPACE}(f)$ (iterieren über Berechnungspfade)
- $\mathrm{NSPACE}(f) \subseteq \mathrm{DSPACE}(f^2)$ (vgl. Papadimitriou, *Computational Complexity*, S. 150)
  $\Rightarrow \mathrm{NPSPACE} = \mathrm{PSPACE}$
- $\mathrm{NSPACE}(f) \subseteq \mathrm{DTIME}(k^{\log(n)+f(n)})$ (vgl. Papadimitriou, S. 149)

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Übersicht

1. Zeit- und Platzklassen
2. Schaltkreise
3. Reduktionen
   - Many-One-Reduktionen
   - Turing-Reduktionen und Orakel

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Syntax von Schaltkreisen

- Ein an Hardware angelehntes Berechnungsmodell

Definition (Syntax Boolescher Schaltkreise)

Ein *Boolescher Schaltkreis* für Eingaben $x_1 \dots x_n \in \{0,1\}^n$ ist ein gerichteter Graph $C_n = (V, E)$ mit folgenden Eigenschaften:

- Die Knoten $v \in V = \{1, \dots, m\}$ heißen *Gatter*, denen ein *Typ* $s(v) \in \{\text{true}, \text{false}, \wedge, \vee, \neg\} \cup \{x_1, \dots, x_n\}$ zugeordnet ist
- Gatter der Typen $\wedge$ und $\vee$ haben 2 eingehende Kanten, solche des Typs $\neg$ eine, die übrigen keine
- Nur vorhergehende Knoten können Eingang für ein Gatter sein: $\forall (i,j) \in E : i < j$

- Schaltkreise sind kompakter als Formeln

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Semantik von Schaltkreisen

- Auswertung bei Eingabe x: $\mathrm{eval}_x : V \to \{0,1\}$

$$\mathrm{eval}_x(v) = \begin{cases} 1 & s(v) = \text{true} \\ 0 & s(v) = \text{false} \\ x|_i & s(v) = x_i \\ \mathrm{eval}_x(i) \cdot \mathrm{eval}_x(j) & s(v) = \wedge, (i,v), (j,v) \in E \\ \vdots & \end{cases}$$

- Wert $C_n(x) = \mathrm{eval}_x(m)$ (letztes Gatter)

Schaltkreisfamilien

- Um beliebige Eingabelängen zulassen zu können, werden Familien von Schaltkreisen betrachtet, von denen je einer für eine Eingabelänge zuständig ist:

$$C = (C_n)_{n \in \mathbb{N}}$$

- Akzeptierte Sprache:
  $L(C) = \{x \in \{0,1\}^* \mid C_{|x|}(x) = 1\}$
- Weil die einzelnen Schaltkreise vollkommen unabhängig voneinander konstruiert werden können, spricht man von einem *nichtuniformen Berechnungsmodell*
- Komplexitätsmaße für Schaltkreise:

$$\mathrm{size}_C(n) = \|V(C_n)\| + \|E(C_n)\|$$
$$\mathrm{depth}_C(n) = \text{ längster Pfad in } C_n$$

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Klassen Boolescher Schaltkreise

Klassen von Sprachen, die durch bestimmte Schaltkreise entschieden werden können:

- $\mathrm{P/poly}$: Polynomielle Größe
- NC^k: Polynomielle Größe, $\mathrm{depth}_C(n) \in \mathcal{O}(\log^k n)$, logspace-uniform
- $\mathrm{NC} = \bigcup_{k \in \mathbb{N}} \mathrm{NC}^k$
- AC^k: wie NC^k, aber beliebig viele Eingänge pro Gatter
- $\mathrm{AC} = \bigcup_{k \in \mathbb{N}} \mathrm{AC}^k$
- TC^k: Wie AC^k, aber zusätzlich *Threshold-Gates*, die 1 ausgeben wenn mindestens die Hälfte ihrer Eingänge 1 ist

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Fakten über Schaltkreisklassen

- $\mathrm{NC}^k \subseteq \mathrm{AC}^k \subseteq \mathrm{TC}^k$ (per Definition)
- $\mathrm{AC}^k \subseteq \mathrm{NC}^{k+1}$ (Gatter → Baum)
- $\mathrm{NC} \subseteq \mathrm{P}$ (generieren + CVP)
- $\mathrm{P} \subseteq \mathrm{P/poly}$ (advice, Schichten pro Berechnungsschritt)

Insgesamt:

$$\mathrm{AC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1 \subseteq \mathrm{NC}^2 \subseteq \mathrm{NC} \subseteq \mathrm{P} \subseteq \mathrm{P/poly}$$

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Übersicht

1. Zeit- und Platzklassen
2. Schaltkreise
3. Reduktionen
   - Many-One-Reduktionen
   - Turing-Reduktionen und Orakel

Warum Reduktionen?

- Gegeben: Zwei Sprachen $A, B \subseteq \Sigma^*$
- Wir wollen Mitgliedschaft in A ist nicht schwerer als Mitgliedschaft in B zu entscheiden formal ausdrücken
- »A kann auf B reduziert werden«
- Notation: $A \leq_r B$
  (gleich: unterschiedliche Reduktionsarten)
- Es soll gelten: $A \leq_r B \wedge B \in \mathrm{C} \Rightarrow A \in \mathrm{C}$
- Außerdem interessant: Gibt es in C Probleme, sodass kein Problem in C schwerer sind?

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Many-One-Reduktionen

Definition

Seien $A, B \subseteq \Sigma^*$. Es gilt $A \leq_m^p B$ genau dann, wenn es eine in Polynomialzeit berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ gibt, sodass gilt:

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B$$

- Der Name kommt daher, dass f nicht injektiv sein muss
- Variante: Bei $\leq_m^{\log}$ ist f eine logarithmisch platzbeschränkte Funktion
- Reduktionen sind eine (partielle) Ordnung auf der Menge der Sprachen

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Vollständigkeit

Definition

Sei $A \subseteq \Sigma^*$ eine Sprache und $\mathrm{C} \subseteq \mathcal{P}(\Sigma^*)$ eine Sprachklasse.

- A ist C-hart $:\Leftrightarrow \forall L \in \mathrm{C} : L \leq_m^p A$
- A ist C-vollständig $:\Leftrightarrow A \in \mathrm{C}$ und A ist C-hart

- Für Sprachklassen C kleiner als NP wird $\leq_m^{\log}$ verwendet, denn:

$$\forall A \in \mathrm{P} : A \leq_m^p \{1\}$$

Beispiel (Vollständige Probleme)

NP:	SAT, 3SAT, CIRCUIT-SAT, TSP, 3COL, ...
P:	CVP, ...
coNP:	TAUT, UNSAT, ...
NL:	s-t-CON, ...

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Inklusion durch Reduktion

- Zu zeigen: $\mathrm{C} \subseteq \mathrm{D}$ (wobei D-Maschinen P-Algorithmen ausführen können müssen)
- Es gilt:
  - $L_1 \in \mathrm{C}$ ist $\leq_m^p$-vollständig für C (d. h. $\forall A \in \mathrm{C} : A \leq_m^p L_1$)
  - $L_1 \in \mathrm{D}$
- Dann gibt es für jedes $A \in \mathrm{C}$ einen D-Entscheider:
  - Es gilt $A \leq_m^p L_1$, sei f die zugehörige Reduktionsfunktion
  - Es gilt $L_1 \in \mathrm{D}$, sei M_1 die zugehörige Turingmaschine

  D-Algorithmus M_A zum Entscheiden ob $x \in A$:
  1. Eingabe: $x \in \Sigma^*$
  2. Berechne $y := f(x)$
  3. Simuliere $M_1(y)$, gib das Ergebnis aus
- $x \in A \Leftrightarrow y \in L_1 \Leftrightarrow y \in L(M_1) \Leftrightarrow x \in L(M_A)$
- Wegen $\mathrm{P} \subseteq \mathrm{D}$ ist M_A eine D-Maschine

Beispiele für Many-One-Reduktionen

Beispiel (Auswertung zu Erfüllbarkeit)

$\mathrm{EVAL} \leq_m^{\log} \mathrm{SAT}$ via id_{Σ^*}:
Variablenfreie Formeln werden genau dann zu 1 ausgewertet, wenn sie erfüllbar sind.

Beispiel (Schaltkreise zu Formeln)

$form(C) = \varphi$ ist polynomialzeit-berechenbar (kurz $form \in \mathrm{FP}$):
Teilmenge der Kanten merken.
Damit: $\mathrm{CVP} \leq_m^p \mathrm{EVAL}$ und $\mathrm{CIRCUIT\text{-}SAT} \leq_m^p \mathrm{SAT}$

Korollar (unnötig umständlich!)

Wegen Transitivität gilt $\mathrm{CVP} \leq_m^p \mathrm{SAT}$.
Da CVP *P-vollständig ist und* $\mathrm{SAT} \in \mathrm{NP}$, *folgt* $\mathrm{P} \subseteq \mathrm{NP}$.

Orakel

Definition (Orakel-Turingmaschine, OTM)

Eine Turingmaschine mit Orakel $A \subseteq \Sigma^*$ ist eine TM, der eine zusätzliche Operation zur Verfügung steht: Sie kann ein $w \in \Sigma^*$ auf ein spezielles Band schreiben und dann in einem Schritt erfahren, ob $w \in A$ (»Orakel-Frage«).

Definition (Schaltkreis mit Orakel)

Ein Schaltkreis mit Orakel $A \subseteq \Sigma^*$ ist ein Schaltkreis, dem für jede Wortlänge $l \in \mathbb{N}$ ein zusätzlicher Gattertyp zur Verfügung steht: Bei Eingabe $w = w_1 \dots w_l$ geben diese Gatter genau dann 1 aus, wenn $w \in A$ (»Orakel-Gatter«).

Turing-Reduktionen

Notation:

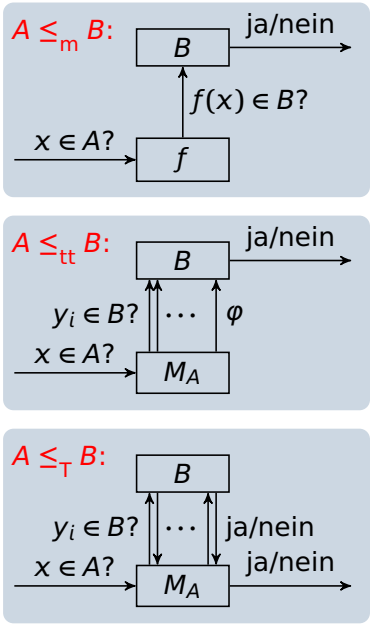
- $\mathrm{C}^A := \left\{ L \subseteq \Sigma^* \;\middle|\; \begin{array}{l} L \text{ kann von einer C-TM bzw. einem} \\ \text{C-Schaltkreis mit Orakel } A \\ \text{entschieden werden} \end{array} \right\}$
- $\mathrm{C}^{\mathrm{D}} := \bigcup_{A \in \mathrm{D}} \mathrm{C}^A$ (nur eine Orakel-Sprache!)

Definition (Turing-Reduktion)

Seien $A, B \subseteq \Sigma^*$ und $\mathrm{C} \subseteq \mathcal{P}(\Sigma^*)$.
Es gilt $A \leq_T^p B$ genau dann, wenn $A \in \mathrm{P}^B$.
Es gilt $A \leq_T^{\log} B$ genau dann, wenn $A \in \mathrm{L}^B$.
Es gilt $A \leq_T^{\mathrm{C}} B$ genau dann, wenn $A \in \mathrm{C}^B$.

Vergleich von Reduktionsarten

- Turing-Reduktionen sind mächtiger als Many-One-Reduktionen
- Weiterer Reduktionstyp: Truth-Table-Reduktionen ($\leq^{\mathrm{p}}_{\mathrm{tt}}$)
  - Keine adaptiven Fragen
  - Formel φ bestimmt, wie die Antworten auf die Orakelfragen kombiniert werden



Turing-Reduktionen: Einfache Beispiele

Beispiel ($\textsc{unsat} \leq^{\mathrm{p}}_{\mathrm{T}} \textsc{sat}$)

UNSAT: Die Menge aller nicht-erfüllbaren Formeln (coNP-vollständig)

- Reduktion: Syntax prüfen, Orakel fragen und Antwort negieren

Beispiel ($\textsc{exact-col} \leq^{\mathrm{p}}_{\mathrm{T}} \textsc{col}$)

Frage: Werden für einen Graphen G genau k_0 Farben zum Färben benötigt?

- Reduktion auf COL (k-Färbbarkeit): Es muss $(G, k_0) \in \textsc{col}$ und $(G, k_0 - 1) \notin \textsc{col}$ gelten

- Beide Reduktionen sind nicht adaptiv, es gilt auch $\textsc{unsat} \leq^{\mathrm{p}}_{\mathrm{tt}} \textsc{sat}$ und $\textsc{exact-col} \leq^{\mathrm{p}}_{\mathrm{tt}} \textsc{col}$
- $\textsc{unsat} \leq^{\mathrm{p}}_{\mathrm{m}} \textsc{sat}$ würde $\mathrm{coNP} \subseteq \mathrm{NP}$ implizieren (unwahrscheinlich)

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Turing-Reduktionen: Weiteres Beispiel

Problem (FIND-SAT)

Eingabe: *Eine Formel φ*

Gesucht: $B\colon \mathrm{var}(\varphi) \to \{0,1\}$ *mit* $\mathrm{eval}_B(\varphi) = 1$

Satz

$\textsc{find-sat} \leq^{\mathrm{FP}}_{\mathrm{T}} \textsc{sat}$, *also* $\textsc{find-sat} \in \mathrm{FP}^{\mathrm{SAT}}$

Ein $\mathrm{FP}^{\mathrm{SAT}}$-Algorithmus für FIND-SAT

```
1  B := ∅
2  for x_i in var(φ) do
3     if φ|_{B∪(x_i↦0)} ∈ SAT then
4        B := B ∪ (x_i ↦ 0)
5     else if φ|_{B∪(x_i↦1)} ∈ SAT then
6        B := B ∪ (x_i ↦ 1)
7     else
8        return fail
9  return B
```


Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Analyse des Algorithmus

Laufzeit: Polynomiell, da Orakelfragen als atomar behandelt werden

Korrektheit: Wenn der Algorithmus eine Belegung zurückgibt, hat er zuvor geprüft, dass sie erfüllend ist. Wenn umgekehrt eine erfüllende Belegung existiert, wird diese auch gefunden.

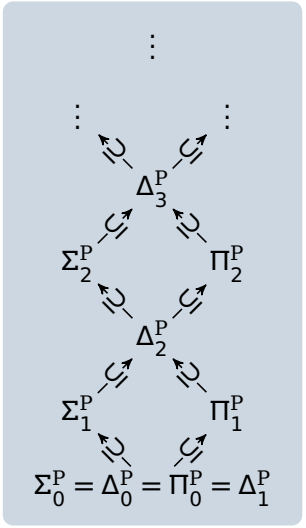
- Es ist keine Truth-Table-Reduktion, da adaptive Fragen verwendet werden
- Eine Truth-Table-Reduktion ist vermutlich nur mit exponentiell vielen Fragen möglich

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Polynomielle Hierarchie

$$\Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} := \mathrm{P}$$
$$\Delta_{i+1}^{\mathrm{P}} := \mathrm{P}^{\Sigma_i^{\mathrm{P}}}$$
$$\Sigma_{i+1}^{\mathrm{P}} := \mathrm{NP}^{\Sigma_i^{\mathrm{P}}}$$
$$\Pi_{i+1}^{\mathrm{P}} := \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$$
$$\mathrm{PH} := \bigcup_{k \in \mathbb{N}} \Sigma_k^{\mathrm{P}}$$

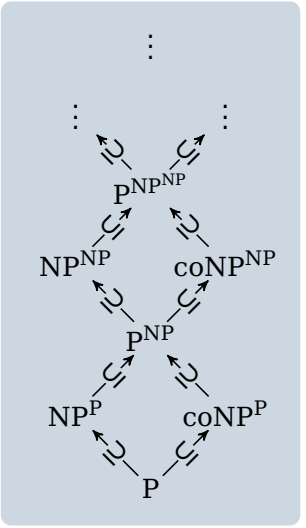
- Es gilt $\mathrm{PH} \subseteq \mathrm{PSPACE}$



Polynomielle Hierarchie

$$\Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} := \mathrm{P}$$
$$\Delta_{i+1}^{\mathrm{P}} := \mathrm{P}^{\Sigma_i^{\mathrm{P}}}$$
$$\Sigma_{i+1}^{\mathrm{P}} := \mathrm{NP}^{\Sigma_i^{\mathrm{P}}}$$
$$\Pi_{i+1}^{\mathrm{P}} := \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$$
$$\mathrm{PH} := \bigcup_{k \in \mathbb{N}} \Sigma_k^{\mathrm{P}}$$

- Es gilt $\mathrm{PH} \subseteq \mathrm{PSPACE}$



Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung

Zusammenfassung

Was wir kennengelernt haben:

- Komplexitätsklassen sind Mengen von Sprachen
- Übliche Ressourcenschranken:
  - Zeit, Platz bei Turingmaschinen
  - Größe und Tiefe bei Schaltkreisen
- Reduktionen zeigen, dass eine Sprache nicht schwerer als eine andere entscheidbar ist

Weiterführende Literatur:

Du, Ding-Zhu and Ker-I Ko.
Theory of Computational Complexity.
New York: Wiley, 2000. ISBN: 0-471-34506-7.

Papadimitriou, Christos H.
Computational Complexity.
Reading, Mass.: Addison-Wesley, 1995.
ISBN: 0-201-53082-1.

Grundlagen der Komplexitätstheorie
Sebastian Kuhnert
Zeit- und Platzklassen
Schaltkreise
Reduktionen
Many-One-Reduktionen
Turing-Reduktionen und Orakel
Zusammenfassung