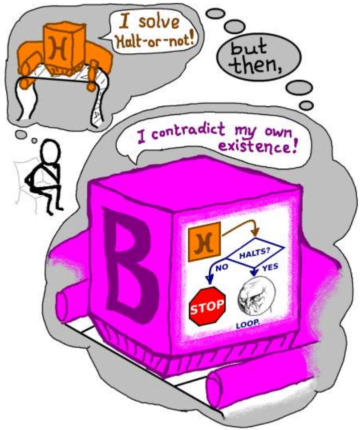
Diagonalisierung und polynomielle Hierarchie

Corvin Paul, Matthias Schimek

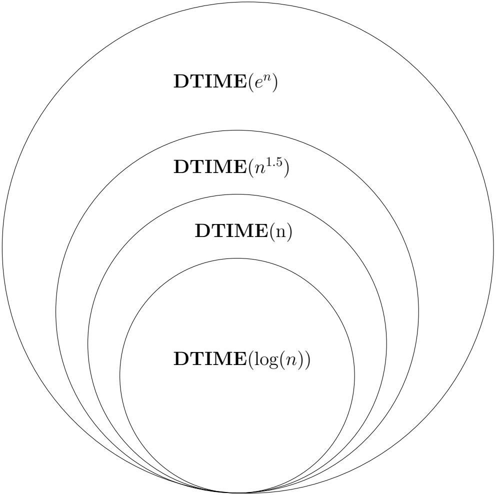
Institut für Theoretische Informatik - Algorithmik I

Einleitung

Diagonalisierung : Was ist das eigentlich?

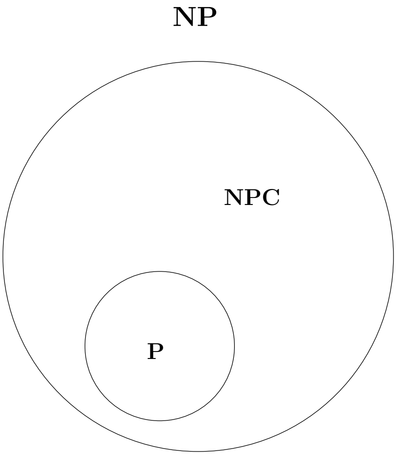


Einleitung
Eine Hierarchie von Komplexitätsklassen



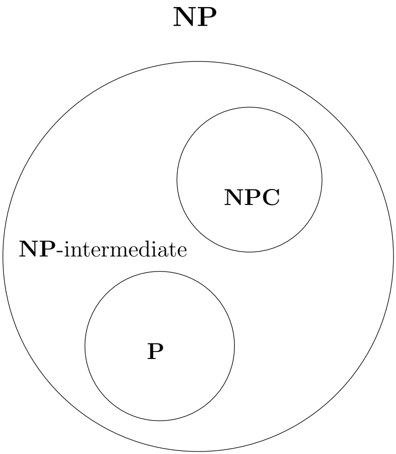
Einleitung

P oder NPC : gibt es noch mehr in NP?



Einleitung

P oder NPC : gibt es noch mehr in NP?



Einleitung

Grenzen der Diagonalisierung

- Orakelmaschinen und die P, NP Frage
- Polynomial Hierarchy : Eine Verallgemeinerung von **P**, **NP**

Einleitung

Grenzen der Diagonalisierung

- Orakelmaschinen und die P, NP Frage
- Polynomial Hierarchy : Eine Verallgemeinerung von **P**, **NP**

Gliederung

Diagonalisierung
- Einleitung
- Was verstehen wir unter Diagonalisierung?
- Time Hierarchy
- Satz von Ladner
- Orakelmaschinen - Grenzen der Diagonalisierung

Die polynomielle Hierarchie
- Einleitung
- Die Klasse **PH**

Diagonalisierung

Was verstehen wir darunter?

- Cantors Diagonalselement zur Überabzählbarkeit von $\mathbb{R}$
- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen

$$
\begin{array}{ll}
s_1 & = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & \quad \vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\ddots
\end{array}
$$

$$
s = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots
$$

Fakultät für Informatik
Institut für Theoretische Informatik

Diagonalisierung

Was verstehen wir darunter?

- Cantors Diagonalselement zur Überabzählbarkeit von $\mathbb{R}$
- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen

$$
\begin{array}{ll}
s_1 & = \textcolor{red}{0}\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,\textcolor{red}{1}\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,\textcolor{red}{0}\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,\textcolor{red}{0}\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,\textcolor{red}{0}\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,\textcolor{red}{1}\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,\textcolor{red}{0}\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,\textcolor{red}{1}\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,\textcolor{red}{1}\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,\textcolor{red}{0}\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,\textcolor{red}{0}\ldots \\
\vdots & \quad \vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\ddots
\end{array}
$$

$$
s = \textcolor{blue}{1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1}\ldots
$$

Diagonalisierung

Was verstehen wir darunter?

- Cantors Diagonalselement zur Überabzählbarkeit von $\mathbb{R}$
- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen

$$\boxed{\begin{array}{lcl}
s_1 & = & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = & 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = & 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = & 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = & 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = & 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = & 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = & 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = & 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = & 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = & 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & & \vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\;\ddots
\end{array}}$$

$$\boxed{s = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots}$$

Diagonalisierung

Was verstehen wir darunter?

- Cantors Diagonalselement zur Überabzählbarkeit von $\mathbb{R}$
- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen

$$
\boxed{\begin{array}{ll}
s_1 & = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots \\
s_2 & = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\ldots \\
s_3 & = 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\ldots \\
s_4 & = 1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\ldots \\
s_5 & = 1\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1\ldots \\
s_6 & = 0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\ldots \\
s_7 & = 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\ldots \\
s_8 & = 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\ldots \\
s_9 & = 1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\ldots \\
s_{10} & = 1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,1\ldots \\
s_{11} & = 1\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\ldots \\
\vdots & \quad \vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\ddots
\end{array}}
$$

$$
\boxed{s = 1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\ldots}
$$

Diagonalisierung

Was verstehen wir darunter?

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Diagonalisierung

Was verstehen wir darunter?

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Diagonalisierung

Was verstehen wir darunter?

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Time Hierarchy

Vorraussetzungen

Wiederholung :

- Für $i \in \mathbb{N}$ beschreibt i die TM M_i
- Jede TM wird von unendlich vielen $i \in \mathbb{N}$ beschrieben
- Es existiert eine universelle TM U, die jede TM mit logarithmischem Overhead simulieren kann

Time Hierarchy

Vorraussetzungen

Wiederholung :

- Für $i \in \mathbb{N}$ beschreibt i die TM M_i
- Jede TM wird von unendlich vielen $i \in \mathbb{N}$ beschrieben
- Es existiert eine universelle TM U, die jede TM mit logarithmischem Overhead simulieren kann

Time Hierarchy

Vorraussetzungen

Wiederholung :

- Für $i \in \mathbb{N}$ beschreibt i die TM M_i
- Jede TM wird von unendlich vielen $i \in \mathbb{N}$ beschrieben
- Es existiert eine universelle TM U, die jede TM mit logarithmischem Overhead simulieren kann

Time Hierarchy

Vorraussetzungen

Universelle TM

TM M_i läuft bei Eingabe x in $\mathcal{O}(f(n)) \Rightarrow$ TM U läuft bei Eingabe i, x in $\mathcal{O}(f(n)log(fn))$

Fakultät für Informatik
Institut für Theoretische Informatik

Time Hierarchy

Vorraussetzungen

Definition Time-constructible functions

Wir nennen eine Funktion f time-constructible, falls gilt :
f(n) ist in $\mathcal{O}(f(n))$ berechenbar.

Definition DTIME

DTIME(f(n)) = { L |∃ deterministische Turingmaschine , die L in $\mathcal{O}(f(n))$ entscheidet }

Time Hierarchy

Vorraussetzungen

Definition Time-constructible functions

Wir nennen eine Funktion f time-constructible, falls gilt :
f(n) ist in $\mathcal{O}(f(n))$ berechenbar.

Definition DTIME

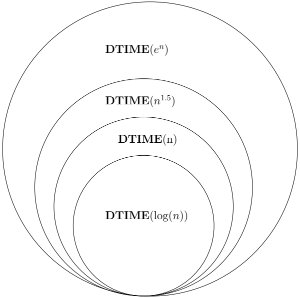
DTIME(f(n)) = { L | $\exists$ deterministische Turingmaschine , die L in $\mathcal{O}(f(n))$ entscheidet }

Time Hierarchy

Deterministische Time Hierarchy

Satz: Time Hierarchy Theorem, 65

Seien f, g time-constructible mit $f(n)\log(f(n)) \in o(g(n))$, dann gilt $\mathbf{DTIME}(f(n)) \subsetneq \mathbf{DTIME}(g(n))$



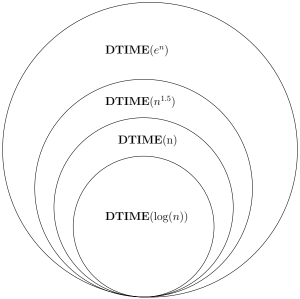
Frage : Warum brauchen wir den Faktor $\log(f(n))$?

Time Hierarchy

Deterministische Time Hierarchy

Satz: Time Hierarchy Theorem, 65

Seien f, g time-constructible mit $f(n)\log(f(n)) \in o(g(n))$, dann gilt $\mathbf{DTIME}(f(n)) \subsetneq \mathbf{DTIME}(g(n))$



Frage : Warum brauchen wir den Faktor $\log(f(n))$?

Time Hierarchy

Beweis det. Time Hierarchy

Wir zeigen $\mathbf{DTIME}(n) \subsetneq \mathbf{DTIME}(n^{1.5})$

Definition Turing Maschine D

Bei Eingabe x : Simuliere die TM M_x mit Eingabe x genau für $|x|^{1.4}$ Schritte. Danach gebe folgendes aus :

$$D(x) = \begin{cases} \overline{M_x(x)} & \text{falls die Simulation eine Ausgabe hatte} \\ 0 & \text{sonst} \end{cases}$$

Sei $L = \{x | D(x) = 1\}$ die von D erzeugte Sprache

Time Hierarchy

Beweis det. Time Hierarchy

Wir zeigen $\mathbf{DTIME}(n) \subsetneq \mathbf{DTIME}(n^{1.5})$

Definition Turing Maschine D

Bei Eingabe x : Simuliere die TM M_x mit Eingabe x genau für $|x|^{1.4}$ Schritte. Danach gebe folgendes aus :

$$D(x) = \begin{cases} \overline{M_x(x)} & \text{falls die Simulation eine Ausgabe hatte} \\ 0 & \text{sonst} \end{cases}$$

Sei $L = \{x | D(x) = 1\}$ die von D erzeugte Sprache

Time Hierarchy

Beweis det. Time Hierarchy

Behauptung

$L \in \textbf{DTIME}(n^{1.5})$ und $L \notin \textbf{DTIME}(n)$

- Wir nehmen an , dass $L \in \textbf{DTIME}(n)$
- $\Rightarrow \exists$ Turing Maschine M , die L entscheidet ($\Leftrightarrow \forall x \in \{0,1\}^*$ D(x) = M(x)) und für Eingabe x höchstens c|x| Schritte benötigt. (c ist konstant)
- Wir konstruieren Wiederspruch , indem wir D eine Gödelnummer i mit $M_i = M$ als Eingabe geben.

Time Hierarchy

Beweis det. Time Hierarchy

Behauptung

$L \in \mathbf{DTIME}(n^{1.5})$ und $L \notin \mathbf{DTIME}(n)$

- Wir nehmen an , dass $L \in \mathbf{DTIME}(n)$
- $\Rightarrow \exists$ Turing Maschine M , die L entscheidet ($\Leftrightarrow \forall x \in \{0,1\}^*$ D(x) = M(x)) und für Eingabe x höchstens c|x| Schritte benötigt. (c ist konstant)
- Wir konstruieren Wiederspruch , indem wir D eine Gödelnummer i mit $M_i = M$ als Eingabe geben.

Time Hierarchy

Beweis det. Time Hierarchy

Behauptung

$L \in \textbf{DTIME}(n^{1.5})$ und $L \notin \textbf{DTIME}(n)$

- Wir nehmen an , dass $L \in \textbf{DTIME}(n)$
- $\Rightarrow \exists$ Turing Maschine M , die L entscheidet ($\Leftrightarrow \forall x \in \{0,1\}^*$ D(x) = M(x)) und für Eingabe x höchstens c|x| Schritte benötigt. (c ist konstant)
- Wir konstruieren Wiederspruch , indem wir D eine Gödelnummer i mit $M_i = M$ als Eingabe geben.

Fakultät für Informatik
Institut für Theoretische Informatik

Time Hierarchy

Beweis det. Time Hierarchy

Behauptung

$L \in \mathbf{DTIME}(n^{1.5})$ und $L \notin \mathbf{DTIME}(n)$

- Wir nehmen an , dass $L \in \mathbf{DTIME}(n)$
- $\Rightarrow \exists$ Turing Maschine M , die L entscheidet ($\Leftrightarrow \forall x \in \{0,1\}^*$ D(x) = M(x)) und für Eingabe x höchstens c|x| Schritte benötigt. (c ist konstant)
- Wir konstruieren Wiederspruch , indem wir D eine Gödelnummer i mit $M_i = M$ als Eingabe geben.

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ □
- Beweis ähnlich auf allgemeinen Fall übertragbar

Fakultät für Informatik
Institut für Theoretische Informatik

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\, n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Time Hierarchy

Beweis det. Time Hierarchy

- Wollen dieses i groß genug, dass D für M_i eine Ausgabe erhält !
- M simuliert auf U läuft in $c|x|\log(|x|)$
- Wir wählen dazu n_0 so groß, dass $\forall\ n \geq n_0$ gilt : $n^{1.4} > cn\log(n)$
- Nun wählen wir eine Gödelnummer x , so dass $|x| > n_0$ und M_x = M
- Damit läuft M_i in der Simulation in D komplett durch und D invertiert das Ergebniss
- Nun gilt $D(x) \neq M(x)$ $\square$
- Beweis ähnlich auf allgemeinen Fall übertragbar

Satz von Ladner

Frage : Gibt es **NP** Probleme , die nicht **NP**-vollständig sind , aber auch nicht in **P** liegen?

Satz von Ladner

NP-intermediate Probleme

Mögliche Kandidaten :

- Graphisomorphie (kommt in Vortrag 7)
- Faktorisierungsproblem
- Kein "natürliches" Problem bekannt

aber,

Satz von Ladner

Behauptung

Existenz einer NP-intermediate Sprache, Ladner, 75

Wenn **P** $\neq$ **NP** dann gilt :
Es existiert eine Sprache $L \in$ **NP** $\setminus$ **P** die nicht **NP**-vollständig ist

Satz von Ladner

Beweisidee

Konstruieren Sprache mit diesen Eigenschaften und zeigen, dass sie in **NP** - intermediate ist, falls **P** $\neq$ **NP** :

Die Sprache $\mathbf{SAT}_H$

Für eine Funktion $H : \mathbb{N} \rightarrow \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

Beispiel für $\mathbf{SAT}_H$

Für $H(n) = n - 1$ und $\psi = a \wedge b$ gilt :
$(a \wedge b)01^{3^2} = (a \wedge b)0111111111 \in \mathbf{SAT}_H$

Satz von Ladner

Beweisidee

Konstruieren Sprache mit diesen Eigenschaften und zeigen, dass sie in **NP** - intermediate ist, falls **P** $\neq$ **NP** :

Die Sprache SAT$_H$

Für eine Funktion $H : \mathbb{N} \rightarrow \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

Beispiel für SAT$_H$

Für $H(n) = n - 1$ und $\psi = a \wedge b$ gilt :
$(a \wedge b)01^{3^2} = (a \wedge b)0111111111 \in \textbf{SAT}_H$

Satz von Ladner

Beweisidee

Konstruieren Sprache mit diesen Eigenschaften und zeigen, dass sie in **NP** - intermediate ist, falls **P** $\neq$ **NP** :

Die Sprache SAT$_H$

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

Beispiel für SAT$_H$

Für $H(n) = n - 1$ und $\psi = a \wedge b$ gilt :
$(a \wedge b)01^{3^2} = (a \wedge b)0111111111 \in \textbf{SAT}_H$

Satz von Ladner

Beweis : Wahl von H

Wir müssen nun also H geschickt konstruieren !

Eigenschaft, die wir von H wollen

SAT$_H \in$ **P** $\Leftrightarrow H(n) \in O(1)$ (also $H(n) \leq C$ für alle n)
und damit insbesondere $\lim_{n\to\infty} H(n) = \infty$ für **SAT**$_H \notin$ **P**

- Es existiert Funktion H , die dies erfüllt
- und **SAT**$_H \in$ **NP**
- ohne Beweis

Satz von Ladner

Beweis : Wahl von H

Wir müssen nun also H geschickt konstruieren !

Eigenschaft, die wir von H wollen

$\mathbf{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1)$ (also $H(n) \leq C$ für alle n)
und damit insbesondere $\lim_{n\to\infty} H(n) = \infty$ für $\mathbf{SAT}_H \notin \mathbf{P}$

- Es existiert Funktion H , die dies erfüllt
- und $\mathbf{SAT}_H \in \mathbf{NP}$
- ohne Beweis

Satz von Ladner

Beweis : Wahl von H

Wir müssen nun also H geschickt konstruieren !

Eigenschaft, die wir von H wollen

$\mathbf{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1)$ (also $H(n) \leq C$ für alle n)
und damit insbesondere $\lim_{n\to\infty} H(n) = \infty$ für $\mathbf{SAT}_H \notin \mathbf{P}$

- Es existiert Funktion H , die dies erfüllt
- und $\mathbf{SAT}_H \in \mathbf{NP}$
- ohne Beweis

Beweis von Ladner

SAT$_H$ weder in P noch NP-complete

Definiton von SAT_H

Für eine Funktion $H : \mathbb{N} \rightarrow \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

SAT$_H$ ist nicht in P

- Angenommen $\textbf{SAT}_H \in \textbf{P} \Rightarrow H(n) \leq C$, C Konstante wegen des gerade bewiesenen Lemmas
- $\textbf{SAT}_H$ ist also **SAT** mit höchsten n^C angehänten 1en
- **SAT** kann somit durch dieselbe TM gelöst werden $\Rightarrow \textbf{P} = \textbf{NP}$

Beweis von Ladner

SAT_H weder in P noch NP-complete

Definiton von SAT_H

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

SAT_H ist nicht in P

- Angenommen $\textbf{SAT}_H \in \textbf{P} \Rightarrow H(n) \leq C$, C Konstante wegen des gerade bewiesenen Lemmas
- $\textbf{SAT}_H$ ist also **SAT** mit höchsten n^C angehänten 1en
- **SAT** kann somit durch dieselbe TM gelöst werden $\Rightarrow \textbf{P} = \textbf{NP}$

Beweis von Ladner

$\mathbf{SAT}_H$ weder in P noch NP-complete

Definiton von SAT_H

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

$\mathbf{SAT}_H$ ist nicht in P

- Angenommen $\mathbf{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$, C Konstante wegen des gerade bewiesenen Lemmas
- $\mathbf{SAT}_H$ ist also **SAT** mit höchsten n^C angehänten 1en
- **SAT** kann somit durch dieselbe TM gelöst werden $\Rightarrow \mathbf{P} = \mathbf{NP}$

Fakultät für Informatik
Institut für Theoretische Informatik

Beweis von Ladner

$\textbf{SAT}_H$ weder in P noch NP-complete

Definiton von SAT_H

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

$\textbf{SAT}_H$ ist nicht in P

- Angenommen $\textbf{SAT}_H \in \textbf{P} \Rightarrow H(n) \leq C$, C Konstante wegen des gerade bewiesenen Lemmas
- $\textbf{SAT}_H$ ist also **SAT** mit höchsten n^C angehänten 1en
- **SAT** kann somit durch dieselbe TM gelöst werden $\Rightarrow \textbf{P} = \textbf{NP}$

Beweis von Ladner

$\mathbf{SAT}_H$ weder in P noch NP-complete

Definiton von $\mathbf{SAT}_H$

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

- Angenommen $\mathbf{SAT}_H \in \mathbf{NP} - \mathit{complete} \Rightarrow$ es existiert poly. Reduktion f von **SAT** auf $\mathbf{SAT}_H$ in $\mathcal{O}(n^i)$
- Da $\mathbf{SAT}_H \notin \mathbf{P}$ geht H gegen ∞
- **SAT**-Instanz φ wird mit f auf $\mathbf{SAT}_H$-Instanz der Form $\psi 0 1^{n^{H(n)}}$ abgebildet und mit $f \in \mathcal{O}(n^i)$ folgt
  $|\psi| + |\psi|^{H(|\psi|)}$ und damit $|\psi| \in o(n)$
- Wegen $\psi| \in o(n)$ existiert dann ein Polynomialzeitalgorithmus für **SAT** und damit $\mathbf{P} = \mathbf{NP} \Rightarrow$ Widerspruch!

Beweis von Ladner

$\mathbf{SAT}_H$ weder in P noch NP-complete

Definiton von $\mathbf{SAT}_H$

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

$\mathbf{SAT}_H$ ist nicht NP-complete

- Angenommen $\mathbf{SAT}_H \in \mathbf{NP} - \textit{complete} \Rightarrow$ es existiert poly. Reduktion f von **SAT** auf $\mathbf{SAT}_H$ in $\mathcal{O}(n^i)$
- Da $\mathbf{SAT}_H \notin \mathbf{P}$ geht H gegen ∞
- **SAT**-Instanz φ wird mit f auf $\mathbf{SAT}_H$-Instanz der Form $\psi 0 1^{n^{H(n)}}$ abgebildet und mit $f \in \mathcal{O}(n^i)$ folgt $|\psi| + |\psi|^{H(|\psi|)}$ und damit $|\psi| \in o(n)$
- Wegen $\psi| \in o(n)$ existiert dann ein Polynomialzeitalgorithmus für **SAT** und damit $\mathbf{P} = \mathbf{NP} \Rightarrow$ Widerspruch!

Beweis von Ladner

$\mathbf{SAT}_H$ weder in P noch NP-complete

Definiton von $\mathbf{SAT}_H$

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 0 1^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

$\mathbf{SAT}_H$ ist nicht NP-complete

- Angenommen $\mathbf{SAT}_H \in \mathbf{NP} - \mathit{complete} \Rightarrow$ es existiert poly. Reduktion f von **SAT** auf $\mathbf{SAT}_H$ in $\mathcal{O}(n^i)$
- Da $\mathbf{SAT}_H \notin \mathbf{P}$ geht H gegen ∞
- **SAT**-Instanz φ wird mit f auf $\mathbf{SAT}_H$-Instanz der Form $\psi 0 1^{n^{H(n)}}$ abgebildet und mit $f \in \mathcal{O}(n^i)$ folgt $|\psi| + |\psi|^{H(|\psi|)}$ und damit $|\psi| \in o(n)$
- Wegen $\psi| \in o(n)$ existiert dann ein Polynomialzeitalgorithmus für **SAT** und damit $\mathbf{P} = \mathbf{NP} \Rightarrow$ Widerspruch!

Corvin Paul, Matthias Schimek:

Fakultät für Informatik
Institut für Theoretische Informatik

Beweis von Ladner

$\mathbf{SAT}_H$ weder in P noch NP-complete

Definiton von $\mathbf{SAT}_H$

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\mathbf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$

$\mathbf{SAT}_H$ ist nicht NP-complete

- Angenommen $\mathbf{SAT}_H \in \mathbf{NP} - \mathit{complete} \Rightarrow$ es existiert poly. Reduktion f von **SAT** auf $\mathbf{SAT}_H$ in $\mathcal{O}(n^i)$
- Da $\mathbf{SAT}_H \notin \mathbf{P}$ geht H gegen ∞
- **SAT**-Instanz φ wird mit f auf $\mathbf{SAT}_H$-Instanz der Form $\psi 01^{n^{H(n)}}$ abgebildet und mit $f \in \mathcal{O}(n^i)$ folgt
  $|\psi| + |\psi|^{H(|\psi|)}$ und damit $|\psi| \in o(n)$
- Wegen $\psi| \in o(n)$ existiert dann ein Polynomialzeitalgorithmus für **SAT** und damit $\mathbf{P} = \mathbf{NP} \Rightarrow$ Widerspruch!

Beweis von Ladner

SAT_H weder in P noch NP-complete

Definiton von SAT_H

Für eine Funktion $H : \mathbb{N} \to \mathbb{N}$ definieren wir :
$\textbf{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \textbf{SAT} \text{ und } n = |\psi|\}$

SAT_H ist nicht NP-complete

- Angenommen $\textbf{SAT}_H \in \textbf{NP} - \textit{complete} \Rightarrow$ es existiert poly. Reduktion f von **SAT** auf $\textbf{SAT}_H$ in $\mathcal{O}(n^i)$
- Da $\textbf{SAT}_H \notin \textbf{P}$ geht H gegen ∞
- **SAT**-Instanz φ wird mit f auf $\textbf{SAT}_H$-Instanz der Form $\psi 01^{n^{H(n)}}$ abgebildet und mit $f \in \mathcal{O}(n^i)$ folgt
  $|\psi| + |\psi|^{H(|\psi|)}$ und damit $|\psi| \in o(n)$
- Wegen $\psi| \in o(n)$ existiert dann ein Polynomialzeitalgorithmus für **SAT** und damit $\textbf{P} = \textbf{NP} \Rightarrow$ Widerspruch!

Grenzen der Diagonalisierung

Wiederholung Diagonalisierung

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Grenzen der Diagonalisierung

Wiederholung Diagonalisierung

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Fakultät für Informatik
Institut für Theoretische Informatik

Grenzen der Diagonalisierung

Wiederholung Diagonalisierung

Was ist Diagonalisierung

Als Diagonalisierung wird (hier) ein Beweis bezeichnet, der nur auf den beiden folgenden Eigenschaften von TM aufbaut.

1. Die Existenz einer Repräsentation von TM durch Zeichenketten (Gödelnummer)
2. Die Fähigkeit eine andere TM mit geringem zusätzlichen Zeit- oder Platzbedarf zu simulieren (Universelle TM)

Grenzen der Diagonalisierung

Definition von Orakelmschinen

Definition Orakel-Turingmaschine

Eine Orakel-Turingmaschine M ist eine TM, die folgende zusätzliche Eigenschaften hat:

- ein spezielles zusätzliches Band (Orakelband) und 3 spezielle zusätzliche Zustände $q_{query}, q_{yes}, q_{no}$.
- ein Orakel $O \subset \{0, 1\}^*$
- Wenn M den Zustand q_{query} betritt, ist der Folgezustand
  - q_{yes}, wenn für Inhalt s des Orakelbands gilt $s \in O$ und
  - q_{no}, wenn $s \notin O$
- Das Orakel liefert die Antwort in einem Berechnungsschritt

Grenzen der Diagonalisierung

Definition von Orakelmschinen

Definition Orakel-Turingmaschine

Eine Orakel-Turingmaschine M ist eine TM, die folgende zusätzliche Eigenschaften hat:

- ein spezielles zusätzliches Band (Orakelband) und 3 spezielle zusätzliche Zustände $q_{query}, q_{yes}, q_{no}$.
- ein Orakel $O \subset \{0,1\}^*$
- Wenn M den Zustand q_{query} betritt, ist der Folgezustand
  - q_{yes}, wenn für Inhalt s des Orakelbands gilt $s \in O$ und
  - q_{no}, wenn $s \notin O$
- Das Orakel liefert die Antwort in einem Berechnungsschritt

Grenzen der Diagonalisierung

Definition von Orakelmschinen

Definition Orakel-Turingmaschine

Eine Orakel-Turingmaschine M ist eine TM, die folgende zusätzliche Eigenschaften hat:

- ein spezielles zusätzliches Band (Orakelband) und 3 spezielle zusätzliche Zustände $q_{query}, q_{yes}, q_{no}$.
- ein Orakel $O \subset \{0,1\}^*$
- Wenn M den Zustand q_{query} betritt, ist der Folgezustand
  - q_{yes}, wenn für Inhalt s des Orakelbands gilt $s \in O$ und
  - q_{no}, wenn $s \notin O$
- Das Orakel liefert die Antwort in einem Berechnungsschritt

Grenzen der Diagonalisierung

Definition von Orakelmschinen

Definition Orakel-Turingmaschine

Eine Orakel-Turingmaschine M ist eine TM, die folgende zusätzliche Eigenschaften hat:

- ein spezielles zusätzliches Band (Orakelband) und 3 spezielle zusätzliche Zustände $q_{query}, q_{yes}, q_{no}$.
- ein Orakel $O \subset \{0,1\}^*$
- Wenn M den Zustand q_{query} betritt, ist der Folgezustand
  - q_{yes}, wenn für Inhalt s des Orakelbands gilt $s \in O$ und
  - q_{no}, wenn $s \notin O$
- Das Orakel liefert die Antwort in einem Berechnungsschritt

Grenzen der Diagonalisierung

Definition von Orakelmschinen

Definition Orakel-Turingmaschine

Eine Orakel-Turingmaschine M ist eine TM, die folgende zusätzliche Eigenschaften hat:

- ein spezielles zusätzliches Band (Orakelband) und 3 spezielle zusätzliche Zustände $q_{query}, q_{yes}, q_{no}$.
- ein Orakel $O \subset \{0, 1\}^*$
- Wenn M den Zustand q_{query} betritt, ist der Folgezustand
  - q_{yes}, wenn für Inhalt s des Orakelbands gilt $s \in O$ und
  - q_{no}, wenn $s \notin O$
- Das Orakel liefert die Antwort in einem Berechnungsschritt

Grenzen der Diagonalisierung

Defintion von Orakelmaschinen

Komplexitätsklassen von Orakelmaschinen

Für jedes $0 \in \{0,1\}^*$ ist $\mathbf{P}^O$ die Menge aller Sprachen, die eine det. Orakel-TM mit Orakel O entscheiden kann. $\mathbf{NP}^O$ analog für nichtdet. Orakel-TM.

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel SAT kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: M eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von M det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{\sigma(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Fakultät für Informatik
Institut für Theoretische Informatik

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \text{ berechnet } 1 \text{ bei Eingabe } x \text{ innerhalb von } 2^n \text{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: M eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von M det. in Exponentialzeit simulieren
  - Orakelaufrufe in Exponentialzeit simulieren (da $EXPCOM \in \mathbf{EXP}$)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \text{ berechnet } 1 \text{ bei Eingabe } x \text{ innerhalb von } 2^n \text{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: *M* eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von *M* det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus **EXP** $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: *M* eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von *M* det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: *M* eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von *M* det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: *M* eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von *M* det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel SAT kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: M eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von M det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $$\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: M eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von M det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Beispiele für Orakelmaschinen

$\overline{SAT}$

- Für $\overline{SAT}$, Sprache der nicht erfüllbaren Formeln, gilt $\overline{SAT} \in \mathbf{P}^{SAT}$.
- Mit Orakel *SAT* kann TM in $\mathcal{O}(1)$ entscheiden, ob $\varphi \in SAT$ und gegenteilige Antwort ausgeben.

EXPCOM

- Sei *EXPCOM* folgende Sprache:

  $\{(M, x, 1^n) : M \textit{ berechnet } 1 \textit{ bei Eingabe } x \textit{ innerhalb von } 2^n \textit{ Schritten}\}$

  Dann gilt $\mathbf{P}^{EXPCOM} = \mathbf{NP}^{EXPCOM} = \mathbf{EXP}$.
- Wegen Orakel aus $\mathbf{EXP} \Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM}$
- Außerdem: *M* eine nichtdet. TM mit Orakel *EXPCOM*:
  - Ausführung von *M* det. in Exponentialzeit simulieren
  - Orakelaufruf in Exponentialzeit simulieren (max $2^{|x|} \cdot 2^{q(|x|)}$ Aufrufe)
- $\Rightarrow \mathbf{EXP} \subseteq \mathbf{P}^{EXPCOM} \subseteq \mathbf{NP}^{EXPCOM} \subseteq \mathbf{EXP}$

Grenzen der Diagonalisierung

Satz von Baker-Gill-Solovay

Satz (Baker,Gill,Solovay, 75)

Es existieren Orakel A, B so dass $\mathbf{P}^A = \mathbf{NP}^A$ und $\mathbf{P}^B \neq \mathbf{NP}^B$

Grenzen der Diagonalisierung

Satz von Baker-Gill-Solovay

Satz (Baker,Gill,Solovay, 75)

Es existieren Orakel A, B so dass $\mathbf{P}^A = \mathbf{NP}^A$ und $\mathbf{P}^B \neq \mathbf{NP}^B$

relativierende Beweise

Wir nennen einen Beweis, der auch für TM mit Orakel gilt, einen *relativierenden Beweis*

- Diagonalisierung ist relativierend und kann damit nicht für die $\mathbf{P} - \mathbf{NP}$ Frage genutzt werden.
- $\Rightarrow$ ein Beweis für die $\mathbf{P} - \mathbf{NP}$ Frage muss ein nicht relativierendes Verfahren nutzen !

Grenzen der Diagonalisierung

Satz von Baker-Gill-Solovay

Satz (Baker,Gill,Solovay, 75)

Es existieren Orakel A, B so dass $\mathbf{P}^A = \mathbf{NP}^A$ und $\mathbf{P}^B \neq \mathbf{NP}^B$

relativierende Beweise

Wir nennen einen Beweis, der auch für TM mit Orakel gilt, einen *relativierenden Beweis*

- Diagonalisierung ist relativierend und kann damit nicht für die $\mathbf{P} - \mathbf{NP}$ Frage genutzt werden.
- $\Rightarrow$ ein Beweis für die $\mathbf{P} - \mathbf{NP}$ Frage muss ein nicht relativierendes Verfahren nutzen !

Grenzen der Diagonalisierung

Beweis : $\mathbf{P}^A = \mathbf{NP}^A$

- $\mathbf{P}^A = \mathbf{NP}^A$ haben wir gerade schon gesehen: Nutze einfach das Orakel A = **EXPCOM**
- B zu konstruieren ist schwieriger (und interessanter!)

Grenzen der Diagonalisierung

Beweis : $\mathbf{P}^A = \mathbf{NP}^A$

- $\mathbf{P}^A = \mathbf{NP}^A$ haben wir gerade schon gesehen: Nutze einfach das Orakel A = **EXPCOM**
- B zu konstruieren ist schwieriger (und interessanter!)

Grenzen der Diagonalisierung

Beweis : $\mathbf{P}^B \neq \mathbf{NP}^B$

Definition unäre Sprache U_B

Für eine Sprache B sei $U_B = \{1^n$: Es gibt einen String der Länge n in B $\}$

- Wir sehen sofort ein : $U_B \in \mathbf{NP}^B$, da eine nicht det. TM ein Zertifikat raten kann.
- Müssen also nur noch B so konstruieren, dass $U_B \notin \mathbf{P}^B$

Grenzen der Diagonalisierung

Beweis : $\mathbf{P}^B \neq \mathbf{NP}^B$

Definition unäre Sprache U_B

Für eine Sprache B sei $U_B = \{1^n$: Es gibt einen String der Länge n in B $\}$

- Wir sehen sofort ein : $U_B \in \mathbf{NP}^B$, da eine nicht det. TM ein Zertifikat raten kann.
- Müssen also nur noch B so konstruieren, dass $U_B \notin \mathbf{P}^B$

Grenzen der Diagonalisierung

Beweis : $\mathbf{P}^B \neq \mathbf{NP}^B$

Definition unäre Sprache U_B

Für eine Sprache B sei $U_B = \{1^n$: Es gibt einen String der Länge n in B $\}$

- Wir sehen sofort ein : $U_B \in \mathbf{NP}^B$, da eine nicht det. TM ein Zertifikat raten kann.
- Müssen also nur noch B so konstruieren, dass $U_B \notin \mathbf{P}^B$

Grenzen der Diagonalisierung

Konstruktion von B

Wir konstruieren eine Folge von Sprachen $(B_i)_{i \in \mathbb{N}}$ so , dass $B = \lim_{n \to \infty} B_i$

- Wie stellen wir sicher, dass alle Turing Maschinen U_B nicht in polynomieller Zeit entscheiden können?
- Tipp: Die Menge aller Turing Maschinen ist abzählbar

Grenzen der Diagonalisierung

Konstruktion von B

Wir konstruieren eine Folge von Sprachen $(B_i)_{i \in \mathbb{N}}$ so , dass $B = \lim_{n \to \infty} B_i$

- Wie stellen wir sicher, dass alle Turing Maschinen U_B nicht in polynomieller Zeit entscheiden können?
- Tipp: Die Menge aller Turing Maschinen ist abzählbar

Grenzen der Diagonalisierung

Konstruktion von B

- Genau : Wir iterieren über alle Turing Maschinen M_i und stellen sicher, dass M_i nicht in polynomieller Zeit U_B entscheiden kann
- Nutze dabei, dass die Anzahl der Wörter exponentiell in der Eingabelänge wächst

Grenzen der Diagonalisierung

Konstruktion von B

- Genau : Wir iterieren über alle Turing Maschinen M_i und stellen sicher, dass M_i nicht in polynomieller Zeit U_B entscheiden kann
- Nutze dabei, dass die Anzahl der Wörter exponentiell in der Eingabelänge wächst

Grenzen der Diagonalisierung

Konstruktion von B

Wir fangen an mit $B_0 = \emptyset$. Konstruktion fr̈ B_i :

- Wähle n so , dass n größer als alle Strings in B_{i-1}
- Lasse M_i auf Eingabe 1^n genau $2^n/10$ Schritte laufen
  (Beachte, dass M_i das Orakel B hat!)

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$

Orakel

B

Turing Maschine

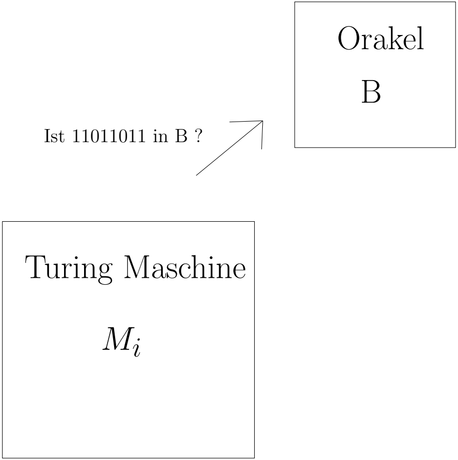
M_i

- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$

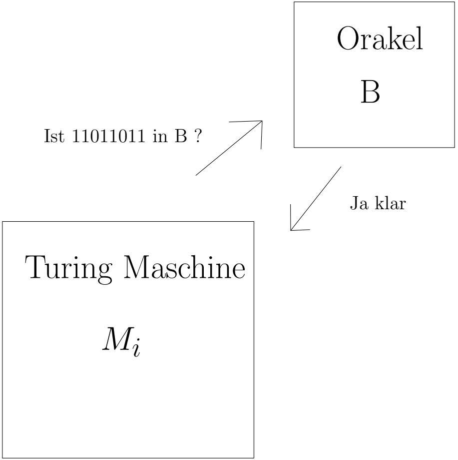


- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$



- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$

Orakel

B

Turing Maschine

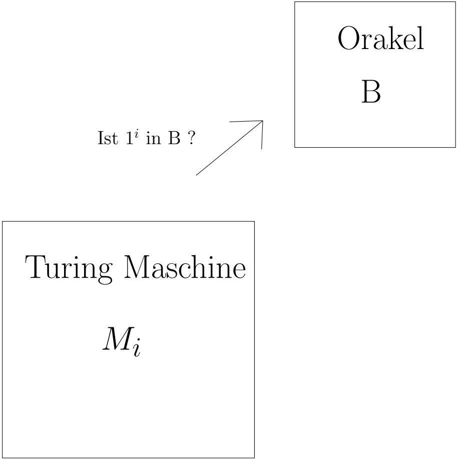
M_i

- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$

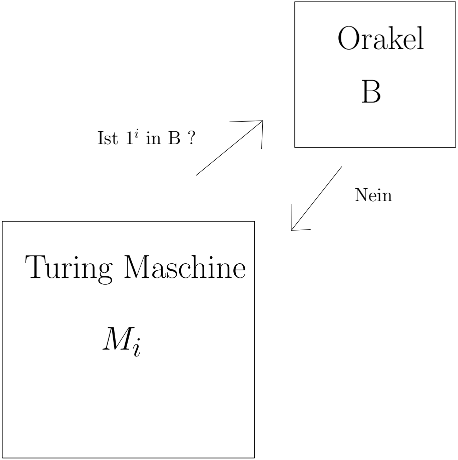


- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

$B_i = \{11011011, 10, 101, 111, 000111\}$



- Das Orakel antwortet konsistent auf dem bisherigen B_i
- Wir merken uns alle Strings der Länge n, die M_i an fragt!

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Fakultät für Informatik
Institut für Theoretische Informatik

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Grenzen der Diagonalisierung

Konstruktion von B

- Wir definieren nun B_{i+1} wie folgt :
- Wenn M_i nicht gehalten hat : $B_{i+1} = B_i$
- ansonsten :
  - M_i akzeptiert 1^n : Wir definieren, dass kein String der Länge n in B ist.
  - M_i lehnt ab : Wähle $x \in \{0,1\}^n$, welches nicht von M_i an gefragt wurde und setze $B_{i+1} = B_i \cup \{x\}$
  - warum existiert dieses x?

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i/10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ $\square$

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i/10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ $\square$

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i/10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ $\square$

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i / 10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ □

Fakultät für Informatik
Institut für Theoretische Informatik

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i/10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ $\square$

Grenzen der Diagonalisierung

Beweis Schluss

- Haben oben ein gesehen, dass $U_B \in \mathbf{NP}^B$
- Und für jede polynomiell beschränkte TM M existiert ein i,so dass
  - M = M_i
  - M auf der Eingabe 1^i weniger als $2^i/10$ Schritte benötigt
  - und damit M_i nach Konstruktion die Frage $1^i \in U_B$ falsch beantwortet
- $\Rightarrow U_B \notin \mathbf{P}^B$ und damit $P^B \neq \mathbf{NP}^B$ $\square$

Einleitun

Verallgemeinerung

- bisher die Komplexitätsklassen **P**, **NPcoNP**
- es gibt Probleme, die sich nicht mit diesen klassifizieren lassen
- durch Verallgemeinerung dieser Klassen kann eine Reihe weiterer Probleme "eingefangen" werden
- Verallgemeinerung ist die "polynomielle Hierarchie" **PH**

Einleitun

Verallgemeinerung

- bisher die Komplexitätsklassen **P**, **NPcoNP**
- es gibt Probleme, die sich nicht mit diesen klassifizieren lassen
- durch Verallgemeinerung dieser Klassen kann eine Reihe weiterer Probleme "eingefangen" werden
- Verallgemeinerung ist die "polynomielle Hierarchie" **PH**

Einleitun

Verallgemeinerung

- bisher die Komplexitätsklassen **P**, **NPcoNP**
- es gibt Probleme, die sich nicht mit diesen klassifizieren lassen
- durch Verallgemeinerung dieser Klassen kann eine Reihe weiterer Probleme "eingefangen" werden
- Verallgemeinerung ist die "polynomielle Hierarchie" **PH**

Einleitun

Verallgemeinerung

- bisher die Komplexitätsklassen **P**, **NPcoNP**
- es gibt Probleme, die sich nicht mit diesen klassifizieren lassen
- durch Verallgemeinerung dieser Klassen kann eine Reihe weiterer Probleme "eingefangen" werden
- Verallgemeinerung ist die "polynomielle Hierarchie" **PH**

Einleitung

Beispiele

g

Definition INDSET

Sei **INDSET** $= \{\langle G, k\rangle$: Graph G hat ein independent set , welches Größe k hat $\}$

Bekannt : **INDSET** $\in$ **NPC**

Definition EXACTINDSET

Sei **EXACTINDSET** $= \{\langle G, k\rangle$: das größte independent set in G hat Größe genau k$\}$
$=\{\langle G, k\rangle$: $\exists$ independent set der Größe k in G und $\forall$ independent sets in G haben Größe $\leq k\}$

Einleitung

Beispiele

g

Definition INDSET

Sei **INDSET** $= \{\langle G, k\rangle$: Graph G hat ein independent set , welches Größe k hat $\}$

Bekannt : **INDSET** $\in$ **NPC**

Definition EXACTINDSET

Sei **EXACTINDSET** $= \{\langle G, k\rangle$: das größte independent set in G hat Größe genau k$\}$

$=\{\langle G, k\rangle$: $\exists$ independent set der Größe k in G und $\forall$ independent sets in G haben Größe $\leq k\}$

Einleitung

Beispiele

g

Definition INDSET

Sei **INDSET** $= \{\langle G, k\rangle$: Graph G hat ein independent set , welches Größe k hat $\}$

Bekannt : **INDSET** $\in$ **NPC**

Definition EXACTINDSET

Sei **EXACTINDSET** $= \{\langle G, k\rangle$: das größte independent set in G hat Größe genau k$\}$
=$\{\langle G, k\rangle$: $\exists$ independent set der Größe k in G und $\forall$ independent sets in G haben Größe $\leq k\}$

Einleitung

Die Klasse $\sum_2^p$

INDSET

Sei **INDSET** $= \{\langle G, k\rangle$: $\exists$ independent set in G, welches Größe k hat $\}$

Wiederholung NP

NP ist die Menge aller Sprachen L für die gilt :
Es gibt deterministische polynomielle TM M und ein Polynom q so dass :

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)} \;\; M(x,u) = 1$$

Einleitung

Die Klasse $\sum_2^p$

EXACTINDSET

Sei **INDSET** $= \{\langle G, k\rangle$: $\exists$ independent set in G, welches Größe k hat und $\forall$ independent sets in G haben Größe $\leq k\}$

Definition $\sum_2^p$

$\sum_2^p$ ist die Menge aller Sprachen L für die gilt :
Es gibt deterministische polynomielle TM M und ein Polynom q so dass :

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall v \in \{0,1\}^{q(|x|)}\ M(x,u,v) = 1$$

Einleitung

Noch mehr Quantoren?

Die Klasse PH

Definition von PH

Definition $\sum_i^p$

$\sum_i^p$ ist die Menge aller Sprachen L für die gilt :
Es gibt deterministische polynomielle TM M und ein Polynom q so dass :

$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \; \forall u_2 \in \{0,1\}^{q(|x|)} \ldots \mathcal{Q}_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, \ldots, u_i) = 1$

wobei $\mathcal{Q}_i$ entweder $\forall$ oder $\exists$ beschreibt, abhängig davon ob i gerade oder ungerade ist

Die Klasse PH

Definition von PH

Definition $\sum_i^p$

$\sum_i^p$ ist die Menge aller Sprachen L für die gilt :
Es gibt deterministische polynomielle TM M und ein Polynom q so dass :

$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \ldots \mathcal{Q}_i u_i \in \{0,1\}^{q(|x|)} M(x, u_1, \ldots, u_i) = 1$

wobei $\mathcal{Q}_i$ entweder $\forall$ oder $\exists$ beschreibt, abhängig davon ob i gerade oder ungerade ist

Definition PH

Die polynomielle Hierarchie ist $\mathbf{PH} = \cup_{i \in \mathbb{N}} \sum_i^p$

Fakultät für Informatik
Institut für Theoretische Informatik

Die Klasse PH

Definition von PH

Definition $\sum_i^p$

$\sum_i^p$ ist die Menge aller Sprachen L für die gilt :
Es gibt deterministische polynomielle TM M und ein Polynom q so dass :

$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \; \forall u_2 \in \{0,1\}^{q(|x|)} \ldots \mathcal{Q}_i u_i \in \{0,1\}^{q(|x|)}$
$M(x, u_1, \ldots, u_i) = 1$

wobei $\mathcal{Q}_i$ entweder $\forall$ oder $\exists$ beschreibt, abhängig davon ob i gerade oder ungerade ist

- Man sieht : $\sum_1^p =$ **NP**
- $\Pi_i^p := co\sum_i^p$
- $\sum_i^p \subseteq \Pi_{i+1}^p \subseteq \sum_{i+2}^p$

Fakultät für Informatik
Institut für Theoretische Informatik

Die Klasse PH
Eigenschaften von PH

- Vermutung: $\mathbf{P} \neq \mathbf{NP}$ und $\mathbf{NP} \neq \mathbf{coNP}$
- Verallgemeinerung: $\sum_i^p \subsetneq \sum_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf $\mathbf{P} - \mathbf{NP}$

- Für alle $i \geq 0$ gilt: $\quad \sum_i^p = \prod_i^p \quad \Rightarrow \quad \mathbf{PH} = \sum_i^p$
- Wenn $\mathbf{P} = \mathbf{NP}$, dann folgt $\mathbf{PH} = \mathbf{P}$

Fakultät für Informatik
Institut für Theoretische Informatik

Die Klasse PH

Eigenschaften von PH

- Vermutung: **P** $\neq$ **NP** und **NP** $\neq$ **coNP**
- Verallgemeinerung: $\sum_i^p \subsetneq \sum_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf P – NP

- Für alle $i \geq 0$ gilt: $\sum_i^p = \prod_i^p \Rightarrow$ **PH** $= \sum_i^p$
- Wenn **P** = **NP**, dann folgt **PH** = **P**

Die Klasse PH

Eigenschaften von PH

- Vermutung: $\mathbf{P} \neq \mathbf{NP}$ und $\mathbf{NP} \neq \mathbf{coNP}$
- Verallgemeinerung: $\sum_i^p \subsetneq \sum_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf P = NP

- Für alle $i \geq 0$ gilt: $\sum_i^p = \prod_i^p \Rightarrow \mathbf{PH} = \sum_i^p$
- Wenn $\mathbf{P} = \mathbf{NP}$, dann folgt $\mathbf{PH} = \mathbf{P}$

Die Klasse PH
Eigenschaften von PH

- Vermutung: **P** $\neq$ **NP** und **NP** $\neq$ **coNP**
- Verallgemeinerung: $\sum_i^p \subsetneq \sum_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf P – NP

- Für alle $i \geq 0$ gilt: $\sum_i^p = \prod_i^p \Rightarrow \mathbf{PH} = \sum_i^p$
- Wenn $\mathbf{P} = \mathbf{NP}$, dann folgt $\mathbf{PH} = \mathbf{P}$

Die Klasse PH

Eigenschaften von PH

- Vermutung: **P** $\neq$ **NP** und **NP** $\neq$ **coNP**
- Verallgemeinerung: $\Sigma_i^p \subsetneq \Sigma_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf P − NP

1. Für alle $i \geq 0$ gilt: $\Sigma_i^p = \Pi_i^p \quad \Rightarrow \quad \mathbf{PH} = \Sigma_i^p$
2. Wenn **P** $=$ **NP**, dann folgt **PH** $=$ **P**

Die Klasse PH

Eigenschaften von PH

- Vermutung: $\mathbf{P} \neq \mathbf{NP}$ und $\mathbf{NP} \neq \mathbf{coNP}$
- Verallgemeinerung: $\sum_i^p \subsetneq \sum_{i+1}^p$ für alle i
- "The polynomial hierarchy does not collapse"

Satz Kollaps von PH und Auswirkungen auf $\mathbf{P} - \mathbf{NP}$

1. Für alle $i \geq 0$ gilt: $\quad \sum_i^p = \prod_i^p \quad \Rightarrow \quad \mathbf{PH} = \sum_i^p$
2. Wenn $\mathbf{P} = \mathbf{NP}$, dann folgt $\mathbf{PH} = \mathbf{P}$

Die Klasse PH

Beweis

Beweis von $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{PH} = \mathbf{P}$

- Sei $\mathbf{P} = \mathbf{NP}$, beweisen über Induktion $\sum_i^p, \prod_i^p \subseteq \mathbf{P}$ für alle i
- IA: $i = 1$, nach Voraussetzung: $\sum_1^p = \mathbf{NP}$, $\prod_1^p = \mathbf{coNP}$ und $\mathbf{P} = \mathbf{coP} = \mathbf{NP} = \mathbf{coNP}$ gilt
- IV: Es gelte $\sum_{i-1}^p \subseteq \mathbf{P}$ für $i - 1 \in \mathbb{N}$
- Anm: $\prod_{i-1}^p$ besteht aus Komplementsprachen der Sprachen in $\sum_{i-1}^p$ $\mathbf{P}$ ist abgeschlossen unter Komplementbildung $\Rightarrow \prod_{i-1}^p \subseteq \mathbf{P}$ unter IV.

Die Klasse PH
Beweis

Beweis von $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{PH} = \mathbf{P}$

- Sei $\mathbf{P} = \mathbf{NP}$, beweisen über Induktion $\sum_i^p, \prod_i^p \subseteq \mathbf{P}$ für alle i
- IA: $i = 1$, nach Voraussetzung: $\sum_1^p = \mathbf{NP}$, $\prod_1^p = \mathbf{coNP}$ und $\mathbf{P} = \mathbf{coP} = \mathbf{NP} = \mathbf{coNP}$ gilt
- IV: Es gelte $\sum_{i-1}^p \subseteq \mathbf{P}$ für $i - 1 \in \mathbb{N}$
- Anm: $\prod_{i-1}^p$ besteht aus Komplementsprachen der Sprachen in $\sum_{i-1}^p$ $\mathbf{P}$ ist abgeschlossen unter Komplementbildung $\Rightarrow \prod_{i-1}^p \subseteq \mathbf{P}$ unter IV.

Die Klasse PH

Beweis

Beweis von $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{PH} = \mathbf{P}$

- Sei $\mathbf{P} = \mathbf{NP}$, beweisen über Induktion $\sum_i^p, \prod_i^p \subseteq \mathbf{P}$ für alle i
- IA: $i = 1$, nach Voraussetzung: $\sum_1^p = \mathbf{NP}$, $\prod_1^p = \mathbf{coNP}$ und $\mathbf{P} = \mathbf{coP} = \mathbf{NP} = \mathbf{coNP}$ gilt
- IV: Es gelte $\sum_{i-1}^p \subseteq \mathbf{P}$ für $i - 1 \in \mathbb{N}$
- Anm: $\prod_{i-1}^p$ besteht aus Komplementsprachen der Sprachen in $\sum_{i-1}^p$ $\mathbf{P}$ ist abgeschlossen unter Komplementbildung $\Rightarrow \prod_{i-1}^p \subseteq \mathbf{P}$ unter IV.

Die Klasse PH

Beweis

Beweis von $\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{PH} = \mathbf{P}$

- Sei $\mathbf{P} = \mathbf{NP}$, beweisen über Induktion $\sum_i^p, \prod_i^p \subseteq \mathbf{P}$ für alle i
- IA: $i = 1$, nach Voraussetzung: $\sum_1^p = \mathbf{NP}$, $\quad \prod_1^p = \mathbf{coNP}$ und $\mathbf{P} = \mathbf{coP} = \mathbf{NP} = \mathbf{coNP}$ gilt
- IV: Es gelte $\sum_{i-1}^p \subseteq \mathbf{P}$ für $i - 1 \in \mathbb{N}$
- Anm: $\prod_{i-1}^p$ besteht aus Komplementsprachen der Sprachen in $\sum_{i-1}^p$ $\mathbf{P}$ ist abgeschlossen unter Komplementbildung $\Rightarrow \prod_{i-1}^p \subseteq \mathbf{P}$ unter IV.

Die Klasse PH

Beweis

- IS: Sei $L \in \sum_i^p$, dann ex. TM M und Polynom q so, dass

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \ldots Q_i u_i \in \{0,1\}^{q(|x|)}$$
$$M(x, u_1, u_2, \ldots, u_i) = 1 (Definition)$$

gilt

- Definiere Sprache L'

$$(x, u_1) \in L' \Leftrightarrow \forall u_2 \in \{0,1\}^{q(|x|)} \ldots Q_i u_i \in \{0,1\}^{q(|x|)}$$
$$M(x, u_1, u_2, \ldots u_i) = 1$$

Die Klasse PH

Beweis

- IS: Sei $L \in \sum_i^p$, dann ex. TM M und Polynom q so, dass

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \ldots Q_i u_i \in \{0,1\}^{q(|x|)}$$
$$M(x, u_1, u_2, \ldots, u_i) = 1 (\mathit{Definition})$$

gilt

- Definiere Sprache L'

$$(x, u_1) \in L' \Leftrightarrow \forall u_2 \in \{0,1\}^{q(|x|)} \ldots Q_i u_i \in \{0,1\}^{q(|x|)}$$
$$M(x, u_1, u_2, \ldots u_i) = 1$$

Die Klasse
Beweis

- L' ist in $\prod_{i-1}^{p}$ (für $\overline{L'}$ alle Quantoren und M negieren $\Rightarrow \overline{L'} \in \sum_{i-1}^{p}$)
- Nach IV gilt: $\prod_{i-1}^{p} \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- Damit ex. det. TM M', die L' in polynom. Zeit berechnet
-
$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1$$
- Damit $L \in \mathbf{NP}$ und da $\mathbf{P} = \mathbf{NP}$ vorausgesetzt, folgt $L \in \mathbf{P}$ $\square$

Die Klasse

Beweis

- L' ist in $\prod_{i-1}^p$ (für $\overline{L'}$ alle Quantoren und M negieren $\Rightarrow \overline{L'} \in \sum_{i-1}^p$)
- Nach IV gilt: $\prod_{i-1}^p \in \mathbf{P} \quad \Rightarrow \quad L' \in \mathbf{P}$
- Damit ex. det. TM M', die L' in polynom. Zeit berechnet
-
$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1$$
- Damit $L \in \mathbf{NP}$ und da $\mathbf{P} = \mathbf{NP}$ vorausgesetzt, folgt $L \in \mathbf{P}$ $\square$

Die Klasse

Beweis

- L' ist in $\prod_{i-1}^{p}$ (für $\overline{L'}$ alle Quantoren und M negieren $\Rightarrow \overline{L'} \in \sum_{i-1}^{p}$)
- Nach IV gilt: $\prod_{i-1}^{p} \in \mathbf{P} \quad \Rightarrow \quad L' \in \mathbf{P}$
- Damit ex. det. TM M', die L' in polynom. Zeit berechnet
- $$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1$$
- Damit $L \in \mathbf{NP}$ und da $\mathbf{P} = \mathbf{NP}$ vorausgesetzt, folgt $L \in \mathbf{P}$ $\square$

Die Klasse

Beweis

- L' ist in $\prod_{i-1}^{p}$ (für $\overline{L'}$ alle Quantoren und M negieren $\Rightarrow \overline{L'} \in \sum_{i-1}^{p}$)
- Nach IV gilt: $\prod_{i-1}^{p} \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- Damit ex. det. TM M', die L' in polynom. Zeit berechnet
- $$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1$$
- Damit $L \in \mathbf{NP}$ und da $\mathbf{P} = \mathbf{NP}$ vorausgesetzt, folgt $L \in \mathbf{P}$ □

Die Klasse
Beweis

- L' ist in $\prod_{i-1}^{p}$ (für $\overline{L'}$ alle Quantoren und M negieren $\Rightarrow \overline{L'} \in \sum_{i-1}^{p}$)
- Nach IV gilt: $\prod_{i-1}^{p} \in \mathbf{P} \quad \Rightarrow \quad L' \in \mathbf{P}$
- Damit ex. det. TM M', die L' in polynom. Zeit berechnet
- $$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} M'(x, u_1) = 1$$
- Damit $L \in \mathbf{NP}$ und da $\mathbf{P} = \mathbf{NP}$ vorausgesetzt, folgt $L \in \mathbf{P}$ $\square$

Die Klasse PH

PHVollständigkeit

Wir definieren **PH** Vollständigkeit analog zur **NP** Vollständigkeit und erhalten damit :

Überlegung zur PH Vollständigkeit

Wenn eine **PH**-vollständige Sprache L existiert dann existiert ein i so dass **PH** $= \sum_i^p$

Beweis :

- Da **PH** $= \bigcup_{k \in \mathbb{N}} \sum_k^p$ $\exists i$ so dass $L \in \sum_i^p$
- Können durch **PH** Vollständigkeit jedes $L' \in$ **PH** in pol. Zeit auf L reduzieren
- und damit also auch $L' \in \sum_i^p$ $\square$

Die Klasse PH

PHVollständigkeit

Wir definieren **PH** Vollständigkeit analog zur **NP** Vollständigkeit und erhalten damit :

Überlegung zur PH Vollständigkeit

Wenn eine **PH**-vollständige Sprache L existiert dann existiert ein i so dass **PH** $= \sum_i^p$

Beweis :

- Da **PH** $= \cup_{k \in \mathbb{N}} \sum_k^p$ $\exists i$ so dass $L \in \sum_i^p$
- Können durch **PH** Vollständigkeit jedes $L' \in$ **PH** in pol. Zeit auf L reduzieren
- und damit also auch $L' \in \sum_i^p$ $\square$

Die Klasse PH

PHVollständigkeit

Wir definieren **PH** Vollständigkeit analog zur **NP** Vollständigkeit und erhalten damit :

Überlegung zur PH Vollständigkeit

Wenn eine **PH**-vollständige Sprache L existiert dann existiert ein i so dass **PH** $= \sum_i^p$

Beweis :

- Da **PH** $= \cup_{k \in \mathbb{N}} \sum_k^p \; \exists i$ so dass $L \in \sum_i^p$
- Können durch **PH** Vollständigkeit jedes $L' \in$ **PH** in pol. Zeit auf L reduzieren
- und damit also auch $L' \in \sum_i^p$ $\square$

Fakultät für Informatik
Institut für Theoretische Informatik

Die Klasse PH

PHVollständigkeit

Wir definieren **PH** Vollständigkeit analog zur **NP** Vollständigkeit und erhalten damit :

Überlegung zur PH Vollständigkeit

Wenn eine **PH**-vollständige Sprache L existiert dann existiert ein *i* so dass **PH** $= \sum_i^p$

Beweis :

- Da **PH** $= \cup_{k \in \mathbb{N}} \sum_k^p \; \exists i$ so dass $L \in \sum_i^p$
- Können durch **PH** Vollständigkeit jedes $L' \in$ **PH** in pol. Zeit auf L reduzieren
- und damit also auch $L' \in \sum_i^p$

Fakultät für Informatik
Institut für Theoretische Informatik

Die Klasse PH

PHVollständigkeit

Wir definieren **PH** Vollständigkeit analog zur **NP** Vollständigkeit und erhalten damit :

Überlegung zur PH Vollständigkeit

Wenn eine **PH**-vollständige Sprache L existiert dann existiert ein *i* so dass **PH** $= \sum_i^p$

Beweis :

- Da **PH** $= \cup_{k \in \mathbb{N}} \sum_k^p \, \exists i$ so dass $L \in \sum_i^p$
- Können durch **PH** Vollständigkeit jedes $L' \in$ **PH** in pol. Zeit auf L reduzieren
- und damit also auch $L' \in \sum_i^p$ $\square$

Fakultät für Informatik
Institut für Theoretische Informatik