

# Diagonalisierung und polynomielle Hierarchie

Corvin Paul, Matthias Schimek

Institut für Theoretische Informatik - Algorithmik I

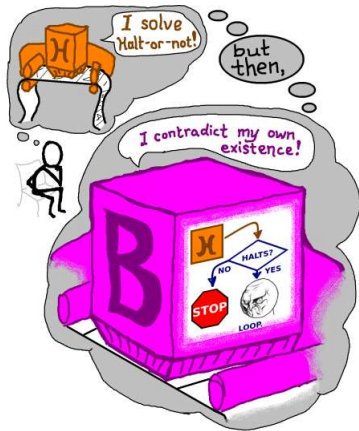


P=NP?

A photograph of the text "P=NP?" written in glowing neon lights on a dark, textured wooden surface. The neon is bright white/yellow, and the background is dark and grainy.

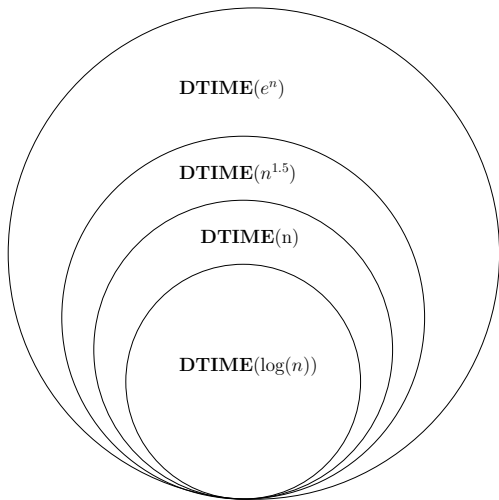
# Diagonalisierung als Beweistechnik

Diagonalisierung : Was ist das eigentlich?

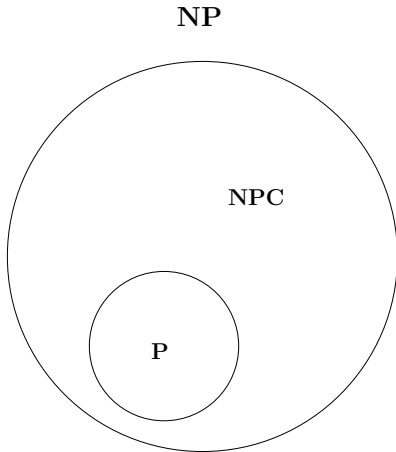


# Diagonalisierung als Beweistechnik

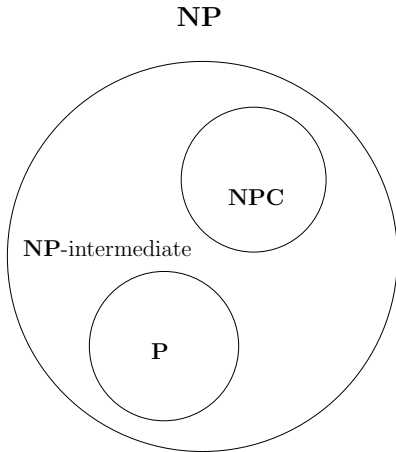
## Eine Hierarchie von Komplexitätsklassen



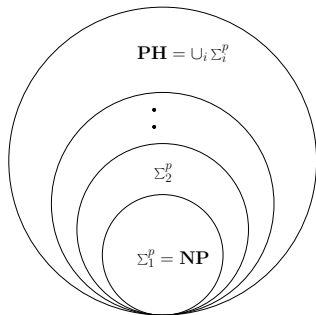
## P oder NPC : gibt es noch mehr in NP?



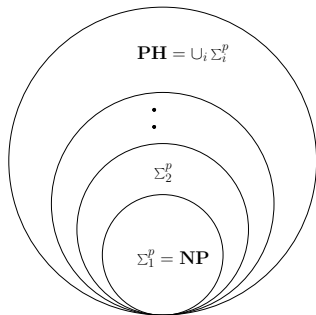
## P oder NPC : gibt es noch mehr in NP?



- Orakelmaschinen und die P, NP Frage



- Verallgemeinerung von **P – NP**
- Kollabiert die **PH**?



- Verallgemeinerung von **P** – **NP**
- Kollabiert die **PH**?



## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

# Diagonalisierung

## Was verstehen wir darunter?

- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen
- In späteren Beweisen gewisse Abstraktion vom "Diagonalprinzip"

$s_1$	=	0	0	0	0	0	0	0	0	0	0	0	0	...
$s_2$	=	1	1	1	1	1	1	1	1	1	1	1	1	...
$s_3$	=	0	1	0	1	0	1	0	1	0	1	0	1	...
$s_4$	=	1	0	1	0	1	0	1	0	1	0	1	0	...
$s_5$	=	1	1	0	1	1	0	1	0	1	1	0	1	...
$s_6$	=	0	0	1	1	0	1	1	0	1	1	0	1	...
$s_7$	=	1	0	0	0	1	0	0	1	0	0	1	0	...
$s_8$	=	0	0	1	1	0	0	1	1	0	0	1	1	...
$s_9$	=	1	1	0	0	1	1	0	0	1	1	0	0	...
$s_{10}$	=	1	1	0	1	1	0	0	1	1	0	1	1	...
$s_{11}$	=	1	1	0	1	0	0	1	0	1	0	1	0	...
$\vdots$		:	:	:	:	:	:	:	:	:	:	:	:	...
$s$	=	1	0	1	1	0	1	0	0	1	1	0	1	...

# Diagonalisierung

## Was verstehen wir darunter?

- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen
- In späteren Beweisen gewisse Abstraktion vom "Diagonalprinzip"

$s_1$	=	0	0	0	0	0	0	0	0	0	0	0	0	...
$s_2$	=	1	1	1	1	1	1	1	1	1	1	1	1	...
$s_3$	=	0	1	0	1	0	1	0	1	0	1	0	1	...
$s_4$	=	1	0	1	0	1	0	1	0	1	0	1	0	...
$s_5$	=	1	1	0	1	1	0	1	0	1	1	0	1	...
$s_6$	=	0	0	1	1	0	1	1	0	1	1	0	1	...
$s_7$	=	1	0	0	1	0	0	1	0	0	1	0	0	...
$s_8$	=	0	0	1	1	0	0	1	1	0	0	1	1	...
$s_9$	=	1	1	0	0	1	1	0	0	1	1	0	0	...
$s_{10}$	=	1	1	0	1	1	0	0	1	0	1	1	0	...
$s_{11}$	=	1	1	0	1	0	0	1	0	1	0	0	1	...
$\vdots$		:	:	:	:	:	:	:	:	:	:	:	:	...
$s$	=	1	0	1	1	0	1	0	0	1	1	0	1	...

# Diagonalisierung

## Was verstehen wir darunter?

- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen
- In späteren Beweisen gewisse Abstraktion vom "Diagonalprinzip"

$s_1$	=	0	0	0	0	0	0	0	0	0	0	0	0	...
$s_2$	=	1	1	1	1	1	1	1	1	1	1	1	1	...
$s_3$	=	0	1	0	1	0	1	0	1	0	1	0	1	...
$s_4$	=	1	0	1	0	1	0	1	0	1	0	1	0	...
$s_5$	=	1	1	0	1	1	0	1	1	0	1	1	0	...
$s_6$	=	0	0	1	1	0	1	0	1	1	0	1	1	...
$s_7$	=	1	0	0	0	1	0	0	1	0	0	1	0	...
$s_8$	=	0	0	1	1	0	0	1	1	0	0	1	1	...
$s_9$	=	1	1	0	0	1	1	0	0	1	1	0	0	...
$s_{10}$	=	1	1	0	1	1	0	0	1	0	1	1	0	...
$s_{11}$	=	1	1	0	1	0	0	1	0	1	0	0	1	...
$\vdots$		:	:	:	:	:	:	:	:	:	:	:	:	...
$s$	=	1	0	1	1	0	1	0	0	1	1	0	1	...

# Diagonalisierung

Was verstehen wir darunter?

- Unentscheidbarkeit des Halteproblems
- informell: Konstruktion eines Elements, das sich von jedem anderen Element unterscheidet
- Diagonalisierung nicht immer "schön" zu sehen
- In späteren Beweisen gewisse Abstraktion vom "Diagonalprinzip"

$s_1$	=	0	0	0	0	0	0	0	0	0	0	0	0	...
$s_2$	=	1	1	1	1	1	1	1	1	1	1	1	1	...
$s_3$	=	0	1	0	1	0	1	0	1	0	1	0	1	...
$s_4$	=	1	0	1	0	1	0	1	0	1	0	1	0	...
$s_5$	=	1	1	0	1	1	0	1	0	1	1	0	1	...
$s_6$	=	0	0	1	1	0	1	0	1	1	0	1	0	...
$s_7$	=	1	0	0	0	1	0	0	1	0	0	1	0	...
$s_8$	=	0	0	1	1	0	0	1	1	0	0	1	1	...
$s_9$	=	1	1	0	0	1	1	0	0	1	1	0	0	...
$s_{10}$	=	1	1	0	1	1	0	0	1	0	1	1	0	...
$s_{11}$	=	1	1	0	1	0	1	0	0	1	0	1	0	...
$\vdots$		:	:	:	:	:	:	:	:	:	:	:	:	...
$s$	=	1	0	1	1	0	1	0	0	1	1	0	1	...

# Diagonalisierung

## Was verstehen wir darunter?

### Was ist Diagonalisierung

Als Diagonalisierung wird (in der Informatik) 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 (in der Informatik) 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 (in der Informatik) 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 unter Diagonalisierung?

**Time Hierarchy**

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

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

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

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

### Universelle TM

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

### Definition Time-constructible functions

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

### Definition DTIME

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

### Definition Time-constructible functions

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

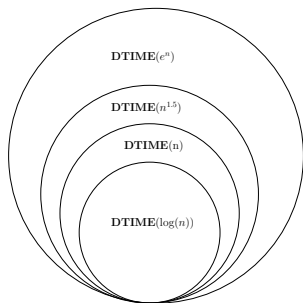
### Definition DTIME

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



### Satz: Time Hierarchy Theorem, 65

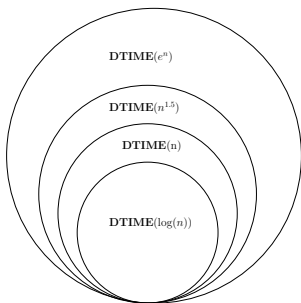
Seien  $f, g$  time-constructible mit  $f(n) \log(f(n)) \in o(g(n))$ , dann gilt  
 **$\text{DTIME}(f(n)) \subsetneq \text{DTIME}(g(n))$**



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

### Satz: Time Hierarchy Theorem, 65

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



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

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

# 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 Widerspruch , indem wir  $D$  eine Gödelnummer  $x$  mit  $M_x = M$  als Eingabe geben.

### 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 Widerspruch , indem wir  $D$  eine Gödelnummer  $x$  mit  $M_x = M$  als Eingabe geben.

### 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 Widerspruch , indem wir  $D$  eine Gödelnummer  $x$  mit  $M_x = M$  als Eingabe geben.

### 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 Widerspruch , indem wir  $D$  eine Gödelnummer  $x$  mit  $M_x = M$  als Eingabe geben.



### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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

### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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

### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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

### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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



### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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

### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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

### Definition Turing Maschine D

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

- Wollen  $|x|$  groß genug, dass D für  $M_x$  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  $M_x$  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



## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

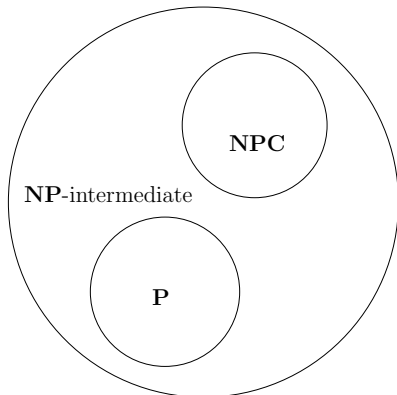
## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**



NP



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  $\mathbf{P} \neq \mathbf{NP}$  dann gilt :

Es existiert eine Sprache  $L \in \mathbf{NP} \setminus \mathbf{P}$  die nicht  $\mathbf{NP}$ -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 $\text{SAT}_H$

Für eine Funktion  $H : \mathbb{N} \rightarrow \mathbb{N}$  definieren wir :

$$\text{SAT}_H = \{ \psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi| \}$$

### Beispiel für $\text{SAT}_H$

Für  $H(n) = n - 1$  und  $\psi = a \wedge b$  gilt :

$$(a \wedge b) 01^{3^2} = (a \wedge b) 0111111111 \in \text{SAT}_H$$

# Satz von Ladner

## Beweisidee

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

### Die Sprache $\text{SAT}_H$

Für eine Funktion  $H : \mathbb{N} \rightarrow \mathbb{N}$  definieren wir :

$$\text{SAT}_H = \{ \psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi| \}$$

### Beispiel für $\text{SAT}_H$

Für  $H(n) = n - 1$  und  $\psi = a \wedge b$  gilt :

$$(a \wedge b) 01^{3^2} = (a \wedge b) 0111111111 \in \text{SAT}_H$$

# Satz von Ladner

## Beweisidee

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

### Die Sprache $\text{SAT}_H$

Für eine Funktion  $H : \mathbb{N} \rightarrow \mathbb{N}$  definieren wir :

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

### Beispiel für $\text{SAT}_H$

Für  $H(n) = n - 1$  und  $\psi = a \wedge b$  gilt :

$$(a \wedge b) 01^{3^2} = (a \wedge b) 0111111111 \in \text{SAT}_H$$

### Definition von $H$

- Betrachte die TM  $M_1, M_2, \dots, M_{\lfloor \log(\log(n)) \rfloor}$ .
- Wähle unter diesen die TM  $M_i$  mit kleinster Gödelnummer  $i$ , welche für alle  $|x| \leq \log(n)$  **SAT** $_H(x)$  in  $i|x|^i$  Schritten berechnet
- Setze  $H(n) = i$ .
- Falls eine solche TM nicht existiert, setze  $H(n) = \log(\log(n))$

### Gewähltes $H$ erfüllt die folgenden Eigenschaften

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

- $H$  erfüllt diese und ist polynomiell berechenbar.
- (ohne Beweis)

# Satz von Ladner

Beweis : Wahl von  $H$

## Definition von $H$

- Betrachte die TM  $M_1, M_2, \dots, M_{\lfloor \log(\log(n)) \rfloor}$ .
- Wähle unter diesen die TM  $M_i$  mit kleinster Gödelnummer  $i$ , welche für alle  $|x| \leq \log(n)$  **SAT** $_H(x)$  in  $i|x|^i$  Schritten berechnet
- Setze  $H(n) = i$ .
- Falls eine solche TM nicht existiert, setze  $H(n) = \log(\log(n))$

## Gewähltes $H$ erfüllt die folgenden Eigenschaften

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

- $H$  erfüllt diese und ist polynomiell berechenbar.
- (ohne Beweis)



### Definition von $H$

- Betrachte die TM  $M_1, M_2, \dots, M_{\lfloor \log(\log(n)) \rfloor}$ .
- Wähle unter diesen die TM  $M_i$  mit kleinster Gödelnummer  $i$ , welche für alle  $|x| \leq \log(n)$  **SAT** $_H(x)$  in  $i|x|^i$  Schritten berechnet
- Setze  $H(n) = i$ .
- Falls eine solche TM nicht existiert, setze  $H(n) = \log(\log(n))$

### Gewähltes $H$ erfüllt die folgenden Eigenschaften

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

- $H$  erfüllt diese und ist polynomiell berechenbar.
- (ohne Beweis)

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

Gewähltes  $H$  erfüllt die folgenden Eigenschaften

$$\text{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1) \text{ (also } H(n) \leq C \text{ für alle } n)$$

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

- Angenommen  $\text{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$ ,  $C$  Konstante
- $\text{SAT}_H$  ist also SAT mit höchstens polynomiell vielen angehängten 1en
- SAT kann somit durch dieselbe TM wie  $\text{SAT}_H$  gelöst werden  
 $\Rightarrow \mathbf{P} = \mathbf{NP}$

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

Gewähltes  $H$  erfüllt die folgenden Eigenschaften

$$\text{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1) \text{ (also } H(n) \leq C \text{ für alle } n)$$

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

- Angenommen  $\text{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$ ,  $C$  Konstante
- $\text{SAT}_H$  ist also SAT mit höchstens polynomiell vielen angehängten 1en
- SAT kann somit durch dieselbe TM wie  $\text{SAT}_H$  gelöst werden  
 $\Rightarrow \mathbf{P} = \mathbf{NP}$

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

Gewähltes  $H$  erfüllt die folgenden Eigenschaften

$$\text{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1) \text{ (also } H(n) \leq C \text{ für alle } n)$$

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

- Angenommen  $\text{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$ ,  $C$  Konstante
- $\text{SAT}_H$  ist also  $\text{SAT}$  mit höchstens polynomiell vielen angehängten 1en
- $\text{SAT}$  kann somit durch dieselbe TM wie  $\text{SAT}_H$  gelöst werden  
 $\Rightarrow \mathbf{P} = \mathbf{NP}$

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$$

Gewähltes  $H$  erfüllt die folgenden Eigenschaften

$$\text{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1) \text{ (also } H(n) \leq C \text{ für alle } n)$$

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

- Angenommen  $\text{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$ ,  $C$  Konstante
- $\text{SAT}_H$  ist also  $\mathbf{SAT}$  mit höchstens polynomiell vielen angehängten 1en
- $\mathbf{SAT}$  kann somit durch dieselbe TM wie  $\text{SAT}_H$  gelöst werden  
 $\Rightarrow \mathbf{P} = \mathbf{NP}$

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \mathbf{SAT} \text{ und } n = |\psi|\}$$

Gewähltes  $H$  erfüllt die folgenden Eigenschaften

$$\text{SAT}_H \in \mathbf{P} \Leftrightarrow H(n) \in O(1) \text{ (also } H(n) \leq C \text{ für alle } n)$$

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

- Angenommen  $\text{SAT}_H \in \mathbf{P} \Rightarrow H(n) \leq C$ ,  $C$  Konstante
- $\text{SAT}_H$  ist also  $\mathbf{SAT}$  mit höchstens polynomiell vielen angehängten 1en
- $\mathbf{SAT}$  kann somit durch dieselbe TM wie  $\text{SAT}_H$  gelöst werden  
 $\Rightarrow \mathbf{P} = \mathbf{NP}$

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{H(n)} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

- Angenommen  $\text{SAT}_H \in \text{NPC} \Rightarrow$  es existiert poly. Reduktion  $f$  von  $\text{SAT}$  auf  $\text{SAT}_H$ .
- Da  $\text{SAT}_H \notin \text{P}$  geht  $H$  gegen  $\infty$
- $\text{SAT}$ -Instanz  $\varphi$  wird mit  $f$  auf  $\text{SAT}_H$ -Instanz der Form  $\psi 01^{H(|\psi|)}$  abgebildet und da  $f$  polynomiell beschränkt, folgt wegen  $|f(\varphi)| = |\psi| + |\psi|^{H(|\psi|)}$ , dass  $|\psi| \in o(n)$ , da sonst  $|\psi|^{H(|\psi|)}$  nicht polynomiell beschränkt sein kann.
- Wegen  $|\psi| \in o(n)$  existiert dann ein Polynomialzeitalgorithmus für  $\text{SAT}$  und damit  $\text{P} = \text{NP} \Rightarrow$  Widerspruch!

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

## $\text{SAT}_H$ ist nicht in $\mathbf{NPC}$

- Angenommen  $\text{SAT}_H \in \mathbf{NPC} \Rightarrow$  es existiert poly. Reduktion  $f$  von  $\text{SAT}$  auf  $\text{SAT}_H$ .
- Da  $\text{SAT}_H \notin \mathbf{P}$  geht  $H$  gegen  $\infty$
- $\text{SAT}$ -Instanz  $\varphi$  wird mit  $f$  auf  $\text{SAT}_H$ -Instanz der Form  $\psi 01^{\psi^{H(|\psi|)}}$  abgebildet und da  $f$  polynomiell beschränkt, folgt wegen  $|f(\varphi)| = |\psi| + |\psi|^{H(|\psi|)}$ , dass  $|\psi| \in o(n)$ , da sonst  $|\psi|^{H(|\psi|)}$  nicht polynomiell beschränkt sein kann.
- Wegen  $|\psi| \in o(n)$  existiert dann ein Polynomialzeitalgorithmus für  $\text{SAT}$  und damit  $\mathbf{P} = \mathbf{NP} \Rightarrow$  Widerspruch!



# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

## $\text{SAT}_H$ ist nicht in $\mathbf{NPC}$

- Angenommen  $\text{SAT}_H \in \mathbf{NPC} \Rightarrow$  es existiert poly. Reduktion  $f$  von  $\text{SAT}$  auf  $\text{SAT}_H$ .
- Da  $\text{SAT}_H \notin \mathbf{P}$  geht  $H$  gegen  $\infty$
- $\text{SAT}$ -Instanz  $\varphi$  wird mit  $f$  auf  $\text{SAT}_H$ -Instanz der Form  $\psi 01^{\psi^{H(|\psi|)}}$  abgebildet und da  $f$  polynomiell beschränkt, folgt wegen  $|f(\varphi)| = |\psi| + |\psi|^{H(|\psi|)}$ , dass  $|\psi| \in o(n)$ , da sonst  $|\psi|^{H(|\psi|)}$  nicht polynomiell beschränkt sein kann.
- Wegen  $|\psi| \in o(n)$  existiert dann ein Polynomialzeitalgorithmus für  $\text{SAT}$  und damit  $\mathbf{P} = \mathbf{NP} \Rightarrow$  Widerspruch!

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

## $\text{SAT}_H$ ist nicht in $\mathbf{NPC}$

- Angenommen  $\text{SAT}_H \in \mathbf{NPC} \Rightarrow$  es existiert poly. Reduktion  $f$  von  $\text{SAT}$  auf  $\text{SAT}_H$ .
- Da  $\text{SAT}_H \notin \mathbf{P}$  geht  $H$  gegen  $\infty$
- $\text{SAT}$ -Instanz  $\varphi$  wird mit  $f$  auf  $\text{SAT}_H$ -Instanz der Form  $\psi 01^{\psi^{H(|\psi|)}}$  abgebildet und da  $f$  polynomiell beschränkt, folgt wegen  $|f(\varphi)| = |\psi| + |\psi|^{H(|\psi|)}$ , dass  $|\psi| \in o(n)$ , da sonst  $|\psi|^{H(|\psi|)}$  nicht polynomiell beschränkt sein kann.
- Wegen  $|\psi| \in o(n)$  existiert dann ein Polynomialzeitalgorithmus für  $\text{SAT}$  und damit  $\mathbf{P} = \mathbf{NP} \Rightarrow$  Widerspruch!

# Beweis von Ladner

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

## Definition von $\text{SAT}_H$

$$\text{SAT}_H = \{\psi 01^{n^{H(n)}} : \psi \in \text{SAT} \text{ und } n = |\psi|\}$$

## $\text{SAT}_H$ ist nicht in $\mathbf{NPC}$

- Angenommen  $\text{SAT}_H \in \mathbf{NPC} \Rightarrow$  es existiert poly. Reduktion  $f$  von  $\text{SAT}$  auf  $\text{SAT}_H$ .
- Da  $\text{SAT}_H \notin \mathbf{P}$  geht  $H$  gegen  $\infty$
- $\text{SAT}$ -Instanz  $\varphi$  wird mit  $f$  auf  $\text{SAT}_H$ -Instanz der Form  $\psi 01^{\psi^{H(|\psi|)}}$  abgebildet und da  $f$  polynomiell beschränkt, folgt wegen  $|f(\varphi)| = |\psi| + |\psi|^{H(|\psi|)}$ , dass  $|\psi| \in o(n)$ , da sonst  $|\psi|^{H(|\psi|)}$  nicht polynomiell beschränkt sein kann.
- Wegen  $|\psi| \in o(n)$  existiert dann ein Polynomialzeitalgorithmus für  $\text{SAT}$  und damit  $\mathbf{P} = \mathbf{NP} \Rightarrow$  Widerspruch!

## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

### 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)

### 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)

### 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 Orakelschienen

- Werden zeigen, dass Diagonalisierung allein **P/NP** Frage nicht beantworten kann
- benötigen hierzu eine weitere Kategorie von Turingmaschinen.

### 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  $O \subseteq \{0, 1\}^*$
- 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 Orakelmaschinen

- Werden zeigen, dass Diagonalisierung allein **P/NP** Frage nicht beantworten kann
- benötigen hierzu eine weitere Kategorie von Turingmaschinen.

### 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  $O \subseteq \{0, 1\}^*$
- 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 Orakelmaschinen

- Werden zeigen, dass Diagonalisierung allein **P/NP** Frage nicht beantworten kann
- benötigen hierzu eine weitere Kategorie von Turingmaschinen.

### 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  $O \subseteq \{0, 1\}^*$
- 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 Orakelmaschinen

- Werden zeigen, dass Diagonalisierung allein **P/NP** Frage nicht beantworten kann
- benötigen hierzu eine weitere Kategorie von Turingmaschinen.

### 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  $O \subseteq \{0, 1\}^*$
- 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 Orakelmaschinen

- Werden zeigen, dass Diagonalisierung allein **P/NP** Frage nicht beantworten kann
- benötigen hierzu eine weitere Kategorie von Turingmaschinen.

### 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  $O \subseteq \{0, 1\}^*$
- 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

## Satz von Baker-Gill-Solovay

### Komplexitätsklassen von Orakelmaschinen

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

### Satz (Baker, Gill, Solovay, 75)

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

# Grenzen der Diagonalisierung

## Satz von Baker-Gill-Solovay

### Komplexitätsklassen von Orakelmaschinen

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

### Satz (Baker, Gill, Solovay, 75)

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

### 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 !

### 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

## Beispiele für Orakelmaschinen

### $\overline{\text{SAT}}$

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

# Grenzen der Diagonalisierung

## Beispiele für Orakelmaschinen

### $\overline{\text{SAT}}$

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

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus  $\text{EXP} \Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$
- 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)
- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus  $\text{EXP} \Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$

- 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)

- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus  $\text{EXP} \Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$
- 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)
- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus **EXP**  $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$
- 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)
- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus **EXP**  $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$
- 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)
- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$

# Grenzen der Diagonalisierung

Beweis :  $P^A = NP^A$

## Beweis des Satzes von Baker:

- 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  $P^{\text{EXPCOM}} = NP^{\text{EXPCOM}} = \text{EXP}$ .

- Wegen Orakel aus **EXP**  $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}}$
- 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)
- $\Rightarrow \text{EXP} \subseteq P^{\text{EXPCOM}} \subseteq NP^{\text{EXPCOM}} \subseteq \text{EXP}$



# Grenzen der Diagonalisierung

Beweis :  $P^B \neq NP^B$

## Definition unäre Sprache $U_B$

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

- Warum gilt  $U_B \in NP^B$ ?
- Müssen also nur noch  $B$  so konstruieren, dass  $U_B \notin P^B$

# Grenzen der Diagonalisierung

Beweis :  $P^B \neq NP^B$

## Definition unäre Sprache $U_B$

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

- Warum gilt  $U_B \in NP^B$ ?
- Müssen also nur noch  $B$  so konstruieren, dass  $U_B \notin P^B$

# Grenzen der Diagonalisierung

Beweis :  $P^B \neq NP^B$

## Definition unäre Sprache $U_B$

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

- Warum gilt  $U_B \in NP^B$ ?
- Müssen also nur noch  $B$  so konstruieren, dass  $U_B \notin P^B$

# Grenzen der Diagonalisierung

## Konstruktion von $B$

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

- Wie stellen wir sicher, dass keine Turing Maschine  $U_B$  in polynomieller Zeit entscheiden kann?
- 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  
 $B = \lim_{i \rightarrow \infty} B_i$

- Wie stellen wir sicher, dass keine Turing Maschine  $U_B$  in polynomieller Zeit entscheiden kann?
- 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 für  $B_i$  :

- 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

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

- 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

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

Ist 11011011 in B ?



Orakel  
B

Turing Maschine

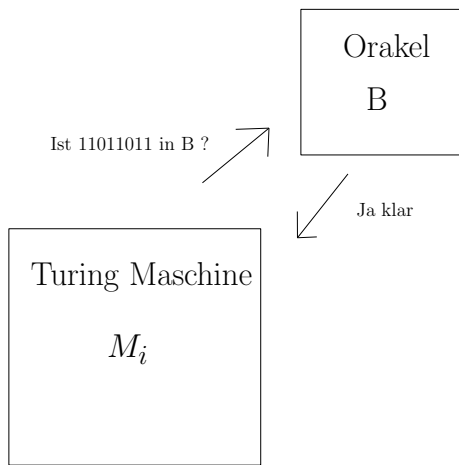
$M_i$

- 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

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

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

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

Ist  $1^i$  in B ?



Orakel  
B

Turing Maschine

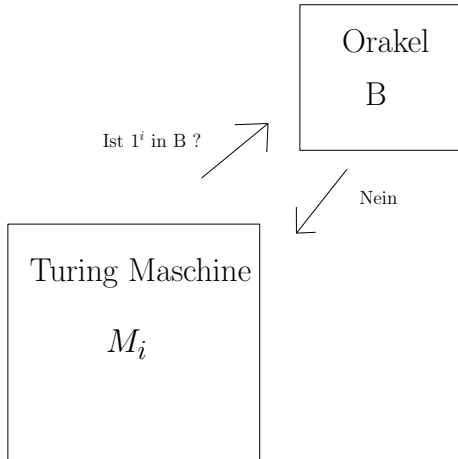
$M_i$

- 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

- 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  $x$ ?

- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □

- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □

- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □



- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □

- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □

- 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  $\mathbf{P}^B \neq \mathbf{NP}^B$  □

## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

- bisher die Komplexitätsklassen **P**, **NP**, **coNP**
- 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**

- bisher die Komplexitätsklassen **P**, **NP**, **coNP**
- 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**

- bisher die Komplexitätsklassen **P**, **NP**, **coNP**
- 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**

- bisher die Komplexitätsklassen **P**, **NP**, **coNP**
- 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**



### Definition INDSET

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

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

### Definition EXACTINDSET

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

### Definition INDSET

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

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

### Definition EXACTINDSET

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

### Definition INDSET

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

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

### Definition EXACTINDSET

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

## INDSET

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

## Wiederholung NP

**NP** ist die Menge aller Sprachen  $L$  für die gilt :

Es gibt eine deterministische polynomielle TM  $M$  und ein Polynom  $q$  so dass :

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

## Die Klasse $\Sigma_2^P$

### EXACTINDSET

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

### Definition $\Sigma_2^P$

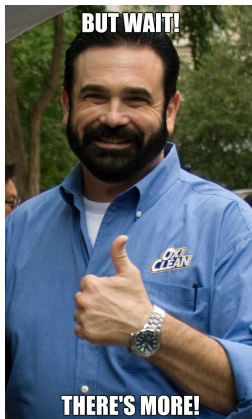
$\Sigma_2^P$  ist die Menge aller Sprachen  $L$  für die gilt :

Es gibt eine deterministische polynomielle TM  $M$  und ein Polynom  $q$  so dass :

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

# Motivation und Beispiele

Noch mehr Quantoren?



## Diagonalisierung

Was verstehen wir unter Diagonalisierung?

Time Hierarchy

Satz von Ladner

Orakelmaschinen - Grenzen der Diagonalisierung

## Die polynomielle Hierarchie

Motivation und Beispiele

Die Klasse **PH**

### Definition $\Sigma_i^P$

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

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

wobei  $Q_i$  entweder  $\forall$  oder  $\exists$  beschreibt, abhängig davon ob  $i$  gerade oder ungerade ist

### Definition PH

Die polynomielle Hierarchie ist  $\mathbf{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^P$



### Definition $\Sigma_i^p$

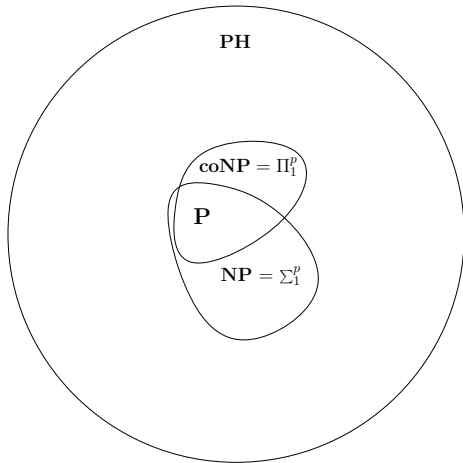
$\Sigma_i^p$  ist die Menge aller Sprachen  $L$  für die gilt :  
Es gibt deterministische polynomielle TM  $M$  und ein Polynom  $q$  so dass :

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

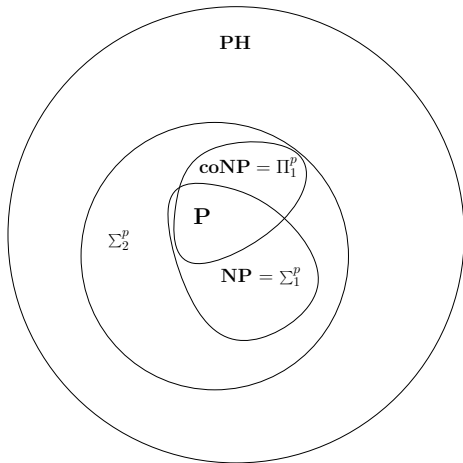
wobei  $Q_i$  entweder  $\forall$  oder  $\exists$  beschreibt, abhängig davon ob  $i$  gerade oder ungerade ist

### Definition PH

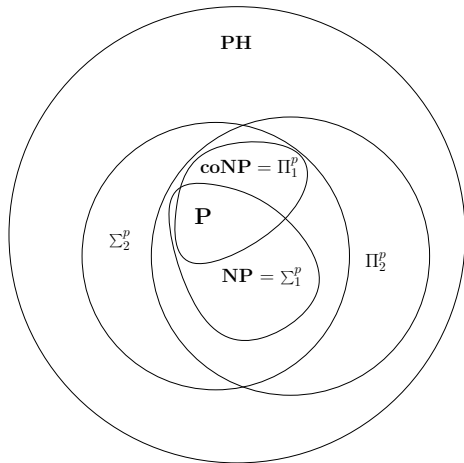
Die polynomielle Hierarchie ist  $\mathbf{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^p$



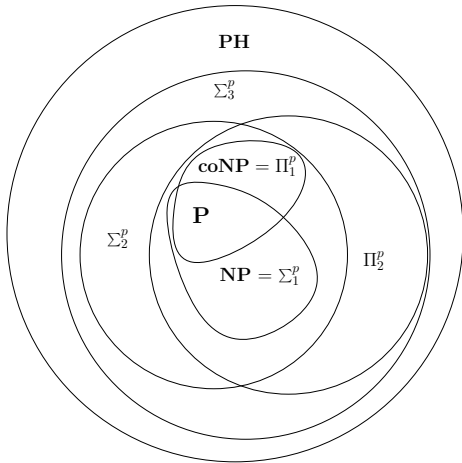
- Man sieht :  $\Sigma_1^P = \mathbf{NP}$
- $\Pi_i^P := \text{co } \Sigma_i^P$
- $\Sigma_i^P \subseteq \Pi_{i+1}^P \subseteq \Sigma_{i+2}^P$



- Man sieht :  $\Sigma_1^p = \text{NP}$
- $\Pi_i^p := \text{co} \Sigma_i^p$
- $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$



- Man sieht :  $\Sigma_1^P = \text{NP}$
- $\Pi_i^P := \text{co } \Sigma_i^P$
- $\Sigma_i^P \subseteq \Pi_{i+1}^P \subseteq \Sigma_{i+2}^P$



- Man sieht :  $\Sigma_1^p = \mathbf{NP}$
- $\Pi_i^p := \text{co} \Sigma_i^p$
- $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$

- 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

Für alle  $i \geq 0$  gilt:  $\Sigma_i^P = \Pi_i^P \Rightarrow PH = \Sigma_i^P$

Wenn  $P = NP$ , dann folgt  $PH = P$

- Vermutung:  $\mathbf{P} \neq \mathbf{NP}$  und  $\mathbf{NP} \neq \mathbf{coNP}$
- 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

Für alle  $i \geq 0$  gilt:  $\Sigma_i^P = \Pi_i^P \Rightarrow \text{PH} = \Sigma_i^P$

Wenn  $\mathbf{P} = \mathbf{NP}$ , dann folgt  $\text{PH} = \mathbf{P}$

- Vermutung:  $\mathbf{P} \neq \mathbf{NP}$  und  $\mathbf{NP} \neq \mathbf{coNP}$
- 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 $\mathbf{P} = \mathbf{NP}$

Für alle  $i \geq 0$  gilt:  $\Sigma_i^P = \Pi_i^P \Rightarrow \mathbf{PH} = \Sigma_i^P$

Wenn  $\mathbf{P} = \mathbf{NP}$ , dann folgt  $\mathbf{PH} = \mathbf{P}$



- Vermutung:  $\mathbf{P} \neq \mathbf{NP}$  und  $\mathbf{NP} \neq \mathbf{coNP}$
- 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

Für alle  $i \geq 0$  gilt:  $\Sigma_i^P = \Pi_i^P \Rightarrow \text{PH} = \Sigma_i^P$

Wenn  $\mathbf{P} = \mathbf{NP}$ , dann folgt  $\text{PH} = \mathbf{P}$

- Vermutung:  $\mathbf{P} \neq \mathbf{NP}$  und  $\mathbf{NP} \neq \mathbf{coNP}$
- 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 $\mathbf{P} = \mathbf{NP}$

1. Für alle  $i \geq 0$  gilt:  $\Sigma_i^P = \Pi_i^P \Rightarrow \mathbf{PH} = \Sigma_i^P$

2. Wenn  $\mathbf{P} = \mathbf{NP}$ , dann folgt  $\mathbf{PH} = \mathbf{P}$

- Vermutung:  $\mathbf{P} \neq \mathbf{NP}$  und  $\mathbf{NP} \neq \mathbf{coNP}$
- 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 $\mathbf{P} = \mathbf{NP}$

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

### Beweis von $P = NP \Rightarrow PH = P$

- Sei  $P = NP$ , beweisen über Induktion  $\Sigma_i^P, \Pi_i^P \subseteq P$  für alle  $i$
- IA:  $i = 1$ , nach Voraussetzung:  $\Sigma_1^P = NP$ ,  $\Pi_1^P = coNP$   
und  $P = coP = NP = coNP$  gilt
- IV: Es gelte  $\Sigma_{i-1}^P \subseteq P$  für  $i - 1 \in \mathbb{N}$
- Anm:  $\Pi_{i-1}^P$  besteht aus Komplementsprachen der Sprachen in  $\Sigma_{i-1}^P$   
 $P$  ist abgeschlossen unter Komplementbildung  $\Rightarrow \Pi_{i-1}^P \subseteq P$  unter IV.

### Beweis von $P = NP \Rightarrow PH = P$

- Sei  $P = NP$ , beweisen über Induktion  $\Sigma_i^P, \Pi_i^P \subseteq P$  für alle  $i$
- **IA:**  $i = 1$ , nach Voraussetzung:  $\Sigma_1^P = NP$ ,  $\Pi_1^P = coNP$   
und  $P = coP = NP = coNP$  gilt
- **IV:** Es gelte  $\Sigma_{i-1}^P \subseteq P$  für  $i - 1 \in \mathbb{N}$
- Anm:  $\Pi_{i-1}^P$  besteht aus Komplementsprachen der Sprachen in  $\Sigma_{i-1}^P$   
 $P$  ist abgeschlossen unter Komplementbildung  $\Rightarrow \Pi_{i-1}^P \subseteq P$  unter IV.

### Beweis von $P = NP \Rightarrow PH = P$

- Sei  $P = NP$ , beweisen über Induktion  $\Sigma_i^P, \Pi_i^P \subseteq P$  für alle  $i$
- **IA:**  $i = 1$ , nach Voraussetzung:  $\Sigma_1^P = NP$ ,  $\Pi_1^P = coNP$   
und  $P = coP = NP = coNP$  gilt
- **IV:** Es gelte  $\Sigma_{i-1}^P \subseteq P$  für  $i - 1 \in \mathbb{N}$
- Anm:  $\Pi_{i-1}^P$  besteht aus Komplementsprachen der Sprachen in  $\Sigma_{i-1}^P$   
 $P$  ist abgeschlossen unter Komplementbildung  $\Rightarrow \Pi_{i-1}^P \subseteq P$  unter IV.

### Beweis von $P = NP \Rightarrow PH = P$

- Sei  $P = NP$ , beweisen über Induktion  $\Sigma_i^P, \Pi_i^P \subseteq P$  für alle  $i$
- **IA:**  $i = 1$ , nach Voraussetzung:  $\Sigma_1^P = NP$ ,  $\Pi_1^P = coNP$   
und  $P = coP = NP = coNP$  gilt
- **IV:** Es gelte  $\Sigma_{i-1}^P \subseteq P$  für  $i - 1 \in \mathbb{N}$
- Anm:  $\Pi_{i-1}^P$  besteht aus Komplementsprachen der Sprachen in  $\Sigma_{i-1}^P$   
 $P$  ist abgeschlossen unter Komplementbildung  $\Rightarrow \Pi_{i-1}^P \subseteq P$  unter IV.

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

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

gilt

- Definiere Sprache  $L'$

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



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

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

gilt

- Definiere Sprache  $L'$

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

- $L'$  ist in  $\Pi_{i-1}^P$  (für  $\overline{L'}$  alle Quantoren und  $M$  negieren  $\Rightarrow \overline{L'} \in \Sigma_{i-1}^P$ )
- Nach IV gilt:  $\Pi_{i-1}^P \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- 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  $L \in \mathbf{P}$  □

- $L'$  ist in  $\Pi_{i-1}^p$  (für  $\overline{L'}$  alle Quantoren und  $M$  negieren  $\Rightarrow \overline{L'} \in \Sigma_{i-1}^p$ )
- Nach IV gilt:  $\Pi_{i-1}^p \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$

- 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  $L \in \mathbf{P}$  □

- $L'$  ist in  $\Pi_{i-1}^p$  (für  $\overline{L'}$  alle Quantoren und  $M$  negieren  $\Rightarrow \overline{L'} \in \Sigma_{i-1}^p$ )
- Nach IV gilt:  $\Pi_{i-1}^p \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- 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  $L \in \mathbf{P}$  □

- $L'$  ist in  $\Pi_{i-1}^p$  (für  $\overline{L'}$  alle Quantoren und  $M$  negieren  $\Rightarrow \overline{L'} \in \Sigma_{i-1}^p$ )
- Nach IV gilt:  $\Pi_{i-1}^p \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- 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  $L \in \mathbf{P}$  □

- $L'$  ist in  $\Pi_{i-1}^p$  (für  $\overline{L'}$  alle Quantoren und  $M$  negieren  $\Rightarrow \overline{L'} \in \Sigma_{i-1}^p$ )
- Nach IV gilt:  $\Pi_{i-1}^p \in \mathbf{P} \Rightarrow L' \in \mathbf{P}$
- 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  $L \in \mathbf{P}$  □

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  $\mathbf{PH} = \sum_i^p$

Beweis :

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

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  $\mathbf{PH} = \Sigma_i^P$

Beweis :

- Da  $\mathbf{PH} = \bigcup_{k \in \mathbb{N}} \Sigma_k^P \exists i$  so dass  $L \in \Sigma_i^P$
- Können durch **PH** Vollständigkeit jedes  $L' \in \mathbf{PH}$  in pol. Zeit auf  $L$  reduzieren
- und damit also auch  $L' \in \Sigma_i^P$  □



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  $\mathbf{PH} = \Sigma_i^P$

Beweis :

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



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  $\mathbf{PH} = \Sigma_i^P$

Beweis :

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



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  $\mathbf{PH} = \Sigma_i^P$

Beweis :

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



- Bild Anfangsseite :  
<https://jeremykun.files.wordpress.com/2012/02/pvsnp.jpg>
- Einleitung Halteproblem :  
<http://s1060.photobucket.com/user/LandruBek/media/dkos/bitter-b8.jpg.html>