

Seminar „Komplexität und Kryptologie“:  
Weder  $P = NP$  noch  $P \neq NP$  kann mit  
relativierenden Techniken bewiesen werden

Martin Stigge  
mstigge@informatik.hu-berlin.de

21. Juni 2008

## 1 Einleitung

Eines der größten und nach wie vor offenen Probleme der Theoretischen Informatik ist das sogenannte „ $P \stackrel{?}{=} NP$ “ Problem. Es beschreibt intuitiv das Verhältnis zwischen Aussagen auf der einen Seite, für die Beweise effizient *gefunden* werden können, und auf der anderen Seite von Aussagen, für die solche Beweise effizient *überprüft* werden können. Beide Arten werden durch die Komplexitätsklassen  $P$  und  $NP$  repräsentiert.

Unmittelbar klar ist dabei nur, dass die erste Eigenschaft die zweite impliziert, also  $P$  in  $NP$  enthalten ist. Die Vermutung, dass die Umkehrung *nicht* gilt (also  $NP$  *nicht* in  $P$  enthalten ist) ist zwar allgemein akzeptiert – jedoch bleibt sie seit geraumer Zeit unbewiesen. Trotz intensiver Forschung während der vergangenen Jahrzehnte durch Komplexitätstheoretiker weltweit ist eine Lösung bisher nicht gefunden. Das Problem wird als derart wichtig und auch schwierig angesehen, dass das Clay Mathematics Institute diese auch als das  $P$ - $NP$ -Problem bekannte Fragestellung im Jahr 2000 in die Liste der 7 „Millenniums-Probleme“ aufgenommen und damit auf seine Lösung ein Preisgeld von 1 Million US-Dollar ausgelobt hat ([Cla]).

Zwar blieb die Suche nach einem Beweis für  $P \neq NP$  (oder auch dessen Gegenteil) bislang trotz zahlreicher vermeintlicher „Lösungen“ (siehe z.B. [Woe]) erfolglos, jedoch konnten im Laufe der Zeit Eigenschaften entdeckt werden, die ein solcher Beweis haben müsste. Es stellt sich nämlich heraus, dass viele klassische Beweistechniken, die in der Komplexitätstheorie für eine sogenannte Separation von Komplexitätsklassen eingesetzt wurden, für eine Lösung von  $P \stackrel{?}{=} NP$  definitiv nicht ausreichen werden.

In diesem Seminar haben wir uns mit diesen „Hürden“ zur Lösung des  $P$ - $NP$ -Problems beschäftigt. In meinem Vortrag und entsprechend dieser Ausarbeitung ging es dabei um das Konzept der *Relativierung*, und gerade dar-

um, dass ein wie auch immer gearteter Beweis für  $P \neq NP$  oder auch  $P = NP$  nicht die Eigenschaft haben kann, zu *relativieren*.

## 2 Relativierung

### 2.1 Orakel-Turingmaschinen

Um uns dem Konzept der Relativierung zu nähern, müssen wir uns zunächst das zugrunde liegende Berechnungsmodell ansehen. Dazu wird die klassische Turingmaschine um die Fähigkeit erweitert, Anfragen an ein sogenanntes *Orakel* stellen zu können. Ein solches Orakel ist einfach eine beliebige Sprache über einem Alphabet  $\Sigma$ . Die Fragen an das Orakel haben dann die Form der Frage nach der Zugehörigkeit eines Wortes  $w \in \Sigma^*$  zu der Orakel-Sprache. Eine Antwort auf eine solche Frage erhält die Orakel-Turingmaschine in einem Schritt:

**Definition 1** (OTM). *Sei  $A \subseteq \Sigma^*$ . Eine Orakel-Turingmaschine  $M^A$  ist eine TM, welche über ein spezielles Frageband Anfragen an die Zugehörigkeit von Wörtern  $w \in \Sigma^*$  zum Orakel  $A$  stellen kann und in einem Schritt die Antwort bekommt.*

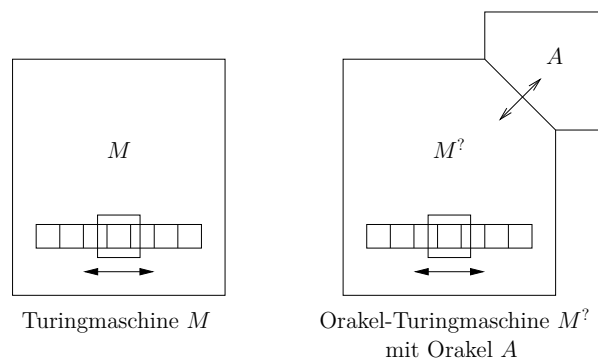


Abbildung 1: Eine klassische Turingmaschine  $M$  im Vergleich mit der Erweiterung zu einer Orakel-Turingmaschine  $M^?$  mit Orakel  $A$ .

Technisch wird dies im Detail so gelöst, dass die Turingmaschine  $M^A$  drei zusätzliche Zustände bekommt:  $q?$ ,  $q_{yes}$  und  $q_{no}$ . Bei einer Frage an das Orakel wird das angefragte Wort  $w \in \Sigma^*$  auf ein spezielles Frageband geschrieben und die Orakel-Turingmaschine wechselt in den Zustand  $q?$ . Im nächsten Schritt ist die Maschine automatisch in  $q_{yes}$ , falls  $w \in A$ , andernfalls im Zustand  $q_{no}$ . (Siehe Abbildung 2.)

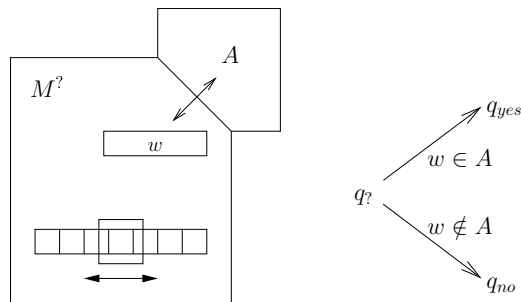


Abbildung 2: Die Orakel-Turingmaschine wechselt vom Fragezustand  $q?$  in einem Schritt in einen der Zustände  $q_{yes}$  und  $q_{no}$ , in Abhängigkeit der Orakelantwort.

Da  $A$  eine beliebige Sprache sein kann, insbesondere also sogar unentscheidbar, ist dieses Konzept ohne weitere Einschränkungen ein überaus mächtiges und besitzt keine praktische Aussagekraft. Sinnvoll ist es daher, neben den Eigenschaften von  $M^?$  (wie Laufzeit und Platzbedarf) auch die von  $A$  zu beschränken und entsprechende Sprachklassen zu definieren.

## 2.2 Relativierte Sprachklassen

Die klassischen Komplexitätsklassen wie  $\mathbf{P}$  oder auch  $\mathbf{PSPACE}$  werden nun *relativ* zu Orakeln betrachtet, d.h. die entsprechend eingeschränkten Turingmaschinen zum Entscheiden von Sprachen der klassischen Klassen haben nun Zugriff auf ihrerseits wiederum eingeschränkte Orakel:

**Definition 2.** Sei  $A \subseteq \Sigma^*$  eine Sprache und  $\mathcal{C}$  eine Komplexitätsklasse.

$$\mathcal{C}^A := \{L \subseteq \Sigma^* \mid L \text{ wird von einer } \mathcal{C}\text{-TM mit Orakel } A \text{ erkannt}\}$$

Dabei wird  $\mathcal{C}^A$  auch bezeichnet mit  $\mathcal{C}$  relativ zu  $A$ .

Damit ist dann zum Beispiel  $\mathbf{P}^A$  für eine Sprache  $A \subseteq \Sigma^*$  die Klasse aller Sprachen, die von Orakel-Turingmaschinen erkannt werden, für die gilt:

1. Sie sind deterministisch.
2. Sie sind in ihrer Laufzeit polynomiell beschränkt.
3. Sie benutzen  $A$  als Orakel.

Die ersten beiden Eigenschaften sind dabei von  $\mathbf{P}$  geerbt, die letzte ist die neue Eigenschaft. Analoge Eigenschaften hat  $\mathbf{NP}^A$ , welches sich im ersten Punkt unterscheidet (nichtdeterministische Orakel-Turingmaschinen, jedoch auch mit polynomieller Laufzeitschranke und  $A$  als Orakel).

### 2.3 Bedeutung von Relativierung

Eine Interpretation von Orakel-Turingmaschinen und den relativierten Komplexitätsklassen ist die, dass wir uns in einem erweiterten Berechnungsmodell bewegen: In diesem Modell haben die betrachteten Maschinen alle Eigenschaften einer klassischen Turingmaschinen – sie können Felder auf unendlichen Bändern lesen und schreiben und den Lese-/Schreibkopf auf dem Band bewegen – jedoch noch die zusätzliche Fähigkeit der Orakel-Entscheidungen in nur *einem* Schritt. Für klassische Beweismethoden aus der Komplexitätstheorie gilt dabei im Allgemeinen, dass diese *relativieren*: Das heißt, dass sie unverändert auch *relativ zu* beliebigen Orakeln gelten.

Als Beispiel betrachten wir das *spezielle Halteproblem*:

$$K := \{x \mid M_x(x) \text{ hält}\}$$

Dabei ist also  $K$  die Menge der Kodierungen aller derjenigen Turingmaschinen, die, wenn sie ihre eigene Kodierung als Eingabe bekommen, halten werden. Diese Sprache ist (durch Turingmaschinen) unentscheidbar. Die Beweistechnik hierfür ist *Diagonalisierung* und hat die Eigenschaft zu *relativieren*. Relativ zu beliebigen Orakeln  $A$  funktioniert der Beweis ganz analog, und liefert damit das Resultat, dass die folgende Sprache auch unentscheidbar ist:

$$K^A := \{x \mid M_x^A(x) \text{ hält}\}$$

Wohlgemerkt ist hier die Unentscheidbarkeit durch Orakel-Turingmaschinen mit Orakel  $A$  gemeint.

Resultate im klassischen Berechnungsmodell der Turingmaschinen, die sich nur relativierender Beweistechniken bedienen, können also direkt in das Modell relativ zu beliebigen Orakeln übertragen werden. Was hilft uns das nun? Eine Intuition kann dazu analog der Skizze in Abbildung 3 so formuliert werden: Falls ein Problem  $X$  in der „normalen Welt“ schwer ist, kann man es in einer „relativierten Welt“ betrachten. Gilt  $X$  dort, lässt es sich dort also insbesondere beweisen, ist dies zunächst ein *Indiz* dafür, dass  $X$  auch in der „normalen Welt“ gelten könnte. Gilt nämlich das Gegenteil (also  $\neg X$ ) in der „normalen Welt“, dürfte der Beweis dafür *nicht relativieren*.

Das Modell der Orakel-Turingmaschinen mag also zwar ein gänzlich „unrealistisches“ sein. So haben solche Orakel doch etwas eher mystisches, im Gegensatz zur sehr konkreten Arbeitsweise von klassischen Turingmaschinen, die Alan Turing schließlich dem intuitiven Berechenbarkeitsbegriff „mit Bleistift auf Papier“ nachbildete. Jedoch bilden sie ein wichtiges Hilfsmittel zur Lösung von schwierigen Problemen in der Komplexitätstheorie.

Ein solches schwieriges Problem ist das P-NP-Problem. Im klassischen Berechnungsmodell ist es bisher nicht gelöst worden, also ist es vielleicht lohnenswert, das Problem relativ zu Orakeln zu betrachten. Leider werden wir Orakel angeben können, relativ zu denen das P-NP-Problem entgegengesetzte Antworten erfährt, die wir auch beweisen werden.

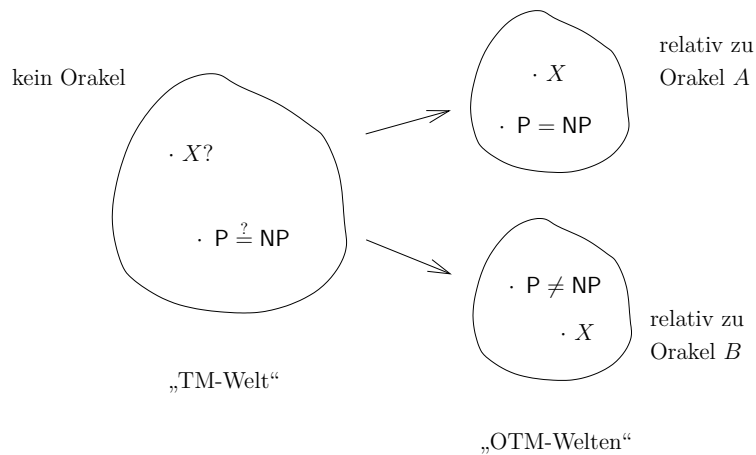


Abbildung 3: Eine relativierende Aussage  $X$  überträgt sich in alle relativierenden „Welten“. Das P-NP-Problem hat in verschiedenen „Orakel-Welten“ verschiedene Antworten, kann also nicht relativieren – egal wie Aussage letztlich lautet.

### 3 Relativierung von $P \stackrel{?}{=} NP$

Wir werden nun das folgende zeigen: Es gibt Orakel  $A \subseteq \Sigma^*$  und  $B \subseteq \Sigma^*$  mit:

- $P^A = NP^A$  und
- $P^B \neq NP^B$ .

Wir geben also zwei „relativierte Welten“ an, in denen die  $P \stackrel{?}{=} NP$  Frage *verschiedene Antworten* hat. Damit kann es weder für  $P = NP$  noch für  $P \neq NP$  einen relativierenden Beweis geben.

#### 3.1 Ein Orakel $A$ für $P^A = NP^A$

**Theorem 1.** *Es gibt eine Sprache  $A \in \{0, 1\}^*$  mit  $P^A = NP^A$ .*

*Beweis.* Für die Gleichheit ist es einfach, ein Orakel  $A$  anzugeben: Wir wählen  $A$  als eine beliebige PSPACE-vollständige Sprache. Solche Sprachen gibt es, wie zum Beispiel das SPACE BOUNDED HALTING PROBLEM ( $SBHP$ ) oder auch QUANTIFIED SATISFIABILITY ( $QSAT$  bzw.  $TQBF$ ).

Wir zeigen nun die folgende Ring-Inklusion:

$$PSPACE \subseteq P^A \subseteq NP^A \subseteq NPSPACE \subseteq PSPACE$$

Schritt für Schritt:

**PSPACE**  $\subseteq$  **P<sup>A</sup>**: Wir hatten  $A$  als PSPACE-vollständige Sprache gewählt, jede Sprache  $L \in$  PSPACE lässt sich also auf  $A$  reduzieren. Diese Reduktion kann eine P-Maschine ausführen und anschließend das Orakel  $A$  fragen, um die Sprachzugehörigkeit zu  $A$  (und damit  $L$ ) zu entscheiden.

**P<sup>A</sup>**  $\subseteq$  **NP<sup>A</sup>**: Die Inklusion  $P \subseteq NP$  relativiert. Im speziellen ist jede **P<sup>A</sup>**-OTM auch eine **NP<sup>A</sup>**-OTM, analog zu klassischen Turingmaschinen.

**NP<sup>A</sup>**  $\subseteq$  **NPSPACE**: Eine NP-Maschine kann von einer NPSPACE-Maschine simuliert werden, da die NP-Maschine mit polynomiell vielen Schritten auch nur polynomiell viele Zellen erreichen kann. Sämtliche Orakel-Anfragen einer **NP<sup>A</sup>**-Maschine mit PSPACE-Orakel  $A$  kann die NPSPACE-Maschine dabei auch simulieren.

**NPSPACE**  $\subseteq$  **PSPACE**: Dies ist Savitch's Theorem bzw. folgt aus diesem direkt.

Damit sind also alle diese Inklusionen in der Tat Gleichheiten und es folgt die Behauptung  $P^A = NP^A$ .  $\square$

### 3.2 Ein Orakel $B$ für $P^B \neq NP^B$

**Theorem 2.** *Es gibt eine Sprache  $B \in \{0, 1\}^*$  mit  $P^B \neq NP^B$ .*

*Beweis (nach [Pap94]).* Klar ist zunächst wieder, dass  $P^B \subseteq NP^B$  ohnehin gilt. Wir möchten daher ein  $B$  konstruieren mit  $NP^B \setminus P^B \neq \emptyset$ . Insbesondere soll es also ein  $L \in NP^B$  geben mit  $L \notin P^B$ .

Zunächst geben wir dieses  $L$  an:

$$L = \{0^n \mid \exists x \in B : |x| = n\}$$

Offenbar ist  $L \in NP^B$ : Eine NP-Maschine kann bei Eingabe  $x$  überprüfen, ob es nur aus Nullen besteht. Im positiven Fall wird nicht-deterministisch ein passendes Wort  $w \in \{0, 1\}^*$  mit  $|w| = |x|$  geraten und dann die Frage  $w \stackrel{?}{\in} B$  an das Orakel gestellt. Die Antwort liefert die Entscheidung, ob  $x \in L$ .

Nachdem nun  $L$  festgelegt ist, ist noch  $B$  offen. Wir werden jetzt  $B$  derart konstruieren, dass  $L \notin P^B$  gilt. Dabei werden wir  $B$  schrittweise für alle Längen  $i$  konstruieren, sodass:

$$B = \bigcup_{i \geq 0} B_i, \quad B_i \subseteq \Sigma^i$$

Die Konstruktion soll dabei sicherstellen, dass es *keine*  $P^B$ -OTM  $M^B$  geben kann mit  $L(M^B) = L$ . Um dies zu gewährleisten, wählen wir eine Aufzählung aller  $P$ -Orakel-Turingmaschinen  $\{M_i^B\}_{i \in \mathbb{N}}$ . (Dies ist mittels einer Standard-Kodierung möglich.) Jedes  $B_i$  wird nun so konstruiert, dass die „zugehörige“ OTM  $M_i^B$  die Zugehörigkeit von  $0^i$  zu  $L$  falsch entscheidet, und damit nicht  $L(M_i^B) = L$  gelten kann.

Wir definieren die  $B_i$  induktiv für alle  $i$ . Begonnen wird mit  $B_0 := \emptyset$ . Für den Induktionsschritt benötigen wir außerdem eine Menge  $X$ , in der als „Menge der Ausnahmen“ alle Wörter gespeichert werden, die nicht in  $B$  landen dürfen. Initial ist auch  $X$  leer.

Nehmen wir nun also an, alle  $B_0, \dots, B_{i-1}$  seien schon konstruiert. Zur Konstruktion von  $B_i$  simulieren wir  $M_i^B(0^i)$  für  $i^{\log i}$  Schritte. (Dieser Funktion  $i \mapsto i^{\log i}$  kommt später besondere Bedeutung zu, da sie superpolynomiell wächst, jedoch weit schwächer als exponentiell.) Das Ziel ist,  $B_i$  derart zu konstruieren, dass es der späteren Antwort von  $M_i^B(0^i)$  widerspricht, d.h. wenn  $M_i^B(0^i) = 1$  sein sollte, möchten wir  $B_i = \emptyset$  erreichen und umgekehrt.

Die Simulation von  $M_i^B(0^i)$  stellt zunächst kein Problem dar, denn bei dieser Konstruktion sind wir auf keinerlei Platz- oder Zeitbeschränkungen angewiesen. Einzig offene Frage bleibt, wie bei Orakel-Anfragen an das erst teilweise definierte  $B$  zu verfahren ist. Stellt die OTM eine Frage „Ist  $w \in B$ ?“ an das Orakel, so können die folgenden Fälle eintreten:

$|w| < i$ : In dem Fall können wir bereits antworten, denn nach Induktionsvoraussetzung sind die  $B_j$  mit  $j < i$  bereits konstruiert. Wir schauen also in  $B_{|w|}$  nach und antworten entsprechend: Das heißt, dass wir die Simulation von  $M_i^B(0^i)$  im Zustand  $q_{yes}$  weiterführen, falls  $w \in B_{|w|}$ , andernfalls im Zustand  $q_{no}$ .

$|w| \geq i$ : In diesem Fall kennen wir noch keine Antwort und entscheiden uns für „nein“. Wir führen die Simulation also im Zustand  $q_{no}$  weiter. Um bei späteren Anfragen (von dieser Maschine  $M_i^B$  oder auch von anderen Maschinen  $M_j^B$  mit  $j > i$ ) konsistent antworten zu können, fügen wir außerdem  $w$  in die „Ausnahmemenge“  $X$  hinzu. Damit werden wir sicherstellen, dass  $w$  wirklich nicht in  $B$  auftauchen wird.

Auf diese Weise kann die Simulation von  $M_i^B(0^i)$  für  $i^{\log i}$  Schritte abgeschlossen werden. Abhängig davon, ob sie innerhalb dieser Zeit gehalten hat, und falls ja, ob dies in einer akzeptierenden Konfiguration geschah, werden wir nun  $B_i$  definieren:

**1. Fall:  $M_i^B(0^i)$  hielt akzeptierend.** Da die Maschine akzeptiert, wir jedoch eine falsche Entscheidung erzeugen wollen, müssen wir dafür sorgen, dass  $B_i$  kein Wort der Länge  $i$  enthält. Wir setzen also  $B_i := \emptyset$ .

**2. Fall:**  $M_i^B(0^i)$  **hielt ohne zu akzeptieren.** Um wieder eine falsche Entscheidung zu erzeugen, wollen wir ein nicht leeres  $B_i$  erhalten. Dabei ist zu beachten, dass alle in  $X$  gesammelten Wörter der Länge  $i$  nicht in  $B_i$  landen dürfen. Wir setzen also:

$$B_i := \Sigma^i \setminus X$$

Um den Widerspruch schließlich zu erhalten, ist noch zu zeigen, dass tatsächlich  $B_i \neq \emptyset$  in diesem Fall gilt, denn nur dann entscheidet  $M_i^B$  das Wort  $0^i$  tatsächlich falsch, bezüglich der Zugehörigkeit zu  $L$ . Einen Beweis hierzu schieben wir kurz auf.

**3. Fall:**  $M_i^B(0^i)$  **hielt nicht.** Dieser Fall kann durchaus eintreten, falls die polynomielle Schranke der Laufzeit von  $M_i^B(0^i)$  hinreichend groß ist, größer als  $i^{\log i}$ . In diesem letzten Fall wissen wir leider nicht, wie  $M_i^B$  sich entscheiden wird und können an dieser Stelle die falsche Entscheidung noch nicht erzwingen.

Die entscheidende Beobachtung hier ist jedoch, dass in der Aufzählung  $\{M_i^?\}_{i \in \mathbb{N}}$  aller P-Orakel-Turingmaschinen jede Maschine nicht nur einmal, sondern (in äquivalenter Form) sogar unendlich oft vorkommt. Dies liegt daran, dass durch triviale Modifikationen äquivalente Maschinen mit anderen Kodierungen erzeugt werden können (z.B. Hinzufügen eines nie verwendeten Dummy-Zustandes). In ihrem Verhalten sind diese Maschinen vollkommen gleich. Wir können also zwar  $M_i^B$  auf  $0^i$  nicht zu einer falschen Entscheidung zwingen, jedoch wird die polynomielle Laufzeitschranke für eine später simulierte Eingabe  $0^j$ ,  $j > i$ , irgendwann unter die Simulationsschranke von  $j^{\log j}$  fallen. Für dieses  $j$  wird die (äquivalente) Maschine dann bis zu einer haltenden Konfiguration simuliert, und gemäß der ersten beiden Fälle zu einer falschen Entscheidung bezüglich  $L$  geführt.

Wir haben nun also gesehen, dass zu jeder Maschine  $M_i^B$  ihre Akzeptanz auf zumindest einem  $0^j$  durch entsprechende Konstruktion von  $B$  derart erzwungen werden kann, dass keine von ihnen genau  $L$  entscheidet. Offen blieb, dass im Fall 2 beim Setzen von  $B_i := \Sigma^i \setminus X$  eine nicht-leere Menge entsteht.

Die entscheidende Beobachtung hierzu ist die folgende: Die Menge  $X$  wird anfangs als leer initialisiert. Für jedes  $j$  wird die Maschine  $M_j^B$  auf Eingabe  $0^j$  für  $j^{\log j}$  Schritte simuliert. Da in jedem dieser Schritte maximal eine Orakel-Anfrage an  $B$  gestellt werden kann, kann pro simulierter Maschine  $M_j^B$  die „Ausnahmemenge“  $X$  maximal um  $j^{\log j}$  Elemente anwachsen.

Damit gilt im Schritt  $i$ :

$$|X| \leq \sum_{j=1}^i j^{\log j} < 2^i$$

(Letztere Ungleichheit folgt z.B. induktiv.) Damit folgt für diesen Fall 2 direkt:

$$|B_i| = |\Sigma^i \setminus X| \geq |\Sigma^i| - |X| > 0$$

Also ist  $B_i$  in diesem Fall nicht leer und damit die falsche Entscheidung auch dort erzwungen.

Es folgt nach Konstruktion, dass  $L$  von keiner P-Orakel-Turingmaschine mit Orakel  $B$  entschieden werden kann, also gilt für dieses  $B$ :  $P^B \not\subseteq NP^B$ .  $\square$

### 3.3 Bemerkungen

Aus dem letzten Teil des Beweises ist also klar geworden, warum die Funktion  $f : i \mapsto i^{\log i}$  gewählt wurde:

- Sie steigt stärker als jedes Polynom:  $f \in n^{\omega(1)}$   
Damit wird jede polynomielle Zeitschranke überschritten.
- Ihre Summe steigt schwächer als exponentiell:  $\sum_{i=1}^n f(i) \in 2^{o(n)}$   
Damit bleibt die Ausnahmemenge  $X$  „klein“.

Jede andere Funktion mit diesen Eigenschaften hätte ebenso gewählt werden können.

Eine weitere Bemerkung betrifft die Eigenschaft  $P^B \neq NP^B$  der Menge  $B$  die wir konstruierten. Von Bennett und Gill konnte in [BG81] gezeigt werden, dass dies auch mit Wahrscheinlichkeit 1 gilt, wenn  $B$  ein Zufallsorakel ist:

$$Pr_B[P^B \neq NP^B] = 1$$

Damit gilt für „fast alle“ Mengen  $B$  die Ungleichung  $P^B \neq NP^B$ . Dies wird manchmal auch als starkes Indiz dafür gesehen, dass in der Tat  $P \neq NP$  gilt – wenngleich das auch noch kein Beweis ist.

Die letzte Bemerkung betrifft die Relativierung von  $P \neq NP$ : Dass diese Ungleichung eben *nicht* relativiert, haben wir bereits im ersten Teil, also in Theorem 1 gezeigt. Würde sie relativieren, dürfte es dieses  $A$  mit  $P^A = NP^A$  nicht geben. Für die Erkenntnis also, dass ein Beweis der allgemeinen Vermutung  $P \neq NP$  „neuartige“ Beweistechniken erfordert (die nicht relativieren), war Theorem 1 bereits ausreichend.

## 4 Zusammenfassung und Ausblick

In diesem Vortrag beschäftigten wir uns mit dem Konzept der Relativierung im Allgemeinen, und wendeten dies im Speziellen auf die Problemstellung des P-NP-Problems an.

Dazu wurde zunächst das Berechnungsmodell der Orakel-Turingmaschinen eingeführt, motiviert und definiert. Es kann angesehen werden als ein „mächtigeres“ Berechnungsmodell als die klassischen Turingmaschinen, um einen anderen Blickwinkel auf Fragestellungen zu ermöglichen. Darauf basierend wurden relativierte Komplexitätsklassen eingeführt und ihre Bedeutung für die Untersuchung von Klassen-Separation veranschaulicht.

Angewendet auf das P-NP-Problem wurde anschließend gezeigt, dass die beiden möglichen Antworten auf diese Fragestellung nicht relativieren. Im Speziellen wurden hierfür zwei Orakel  $A$  und  $B$  konstruiert, relativ zu denen die  $P \stackrel{?}{=} NP$  Frage unterschiedliche Antworten hat.

Im Kontext des Seminars bildete dieser Vortrag damit den Auftakt. Nachdem nun klar war, dass nicht-relativierende Techniken zur Lösung des P-NP-Problems notwendig sind, wurden Resultate betrachtet, die nicht relativieren – so insbesondere Resultate aus dem Bereich der interaktiven Beweissysteme. Schließlich wurde eine weitere Hürde beleuchtet, die als Verallgemeinerung von Relativierung betrachtet werden kann: Die als „Algebrierung“ bezeichnete algebraische Relativierung, die der aktuelle Artikel [AW08] von Aaronson und Wigderson vorstellt.

## A Anhang

### Literatur

- [AW08] Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In Richard E. Ladner and Cynthia Dwork, editors, *STOC*, pages 731–740. ACM, 2008.
- [BG81] C. H. Bennett and J. Gill. Relative to a Random Oracle  $A$ ,  $P^A \neq NP^A \neq co - NP^A$  with Probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.
- [Cla] Clay Mathematics Institute. P vs NP Problem. Available at [http://www.claymath.org/millennium/P\\_vs\\_NP/](http://www.claymath.org/millennium/P_vs_NP/).
- [Pap94] Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
- [Woe] Gerhard J. Woeginger. The P-versus-NP page. Available at <http://www.win.tue.nl/~gwoegi/P-versus-NP.htm>.