Komplexität von Zertifikaten, Heuristiken und Zähltypen mit Anwendungen in der Kryptographie und Schaltkreistheorie

Zusammenfassung der Habilitationsschrift

von
Dr. Jörg-Matthias Rothe

Die Komplexitätstheorie hat sich während der vergangenen drei Jahrzehnte zu einem reichen und eigenständigen Teilgebiet der Theoretischen Informatik entwickelt. Sie hat eine Vielzahl wichtiger Resultate hervorgebracht, die gleichzeitig durch ihre mathematische Schönheit und durch ihre Fähigkeit bestechen, andere – in vielen Fällen angewandte – Gebiete der Informatik zu stimulieren und mit ihnen in eine fruchtbare Wechselbeziehung zu treten. Dazu zählen Gebiete, die so verschiedenartig sind wie Kryptographie, Kodierungs- und Informationstheorie, die Theorie der Datenkomprimierung, Logik, der Entwurf und die Analyse von Algorithmen, Graphentheorie und Schaltkreistheorie.[1] Selbst in anscheinend so entfernten Gebieten wie der Politik- und Sozialwissenschaft konnten kürzlich komplexitätstheoretische Techniken erfolgreich angewandt werden [BTT89a, BTT89b, BTT92, HHR97a, HHR97b]. Historisch und konzeptuell gesehen ist die Komplexitätstheorie eng verwandt mit der Rekursionstheorie und der Theorie der formalen Sprachen und Automaten. Abschließend sei erwähnt, daß die ganz aktuellen Gebiete der Quantumberechnung [Ber97] und der biologischen Berechnung [KMRS97] – die möglicherweise einen entscheidenden Einfluß auf künftige Computer-Technologien haben können – als neue, interdisziplinäre Gebiete auch zur Komplexitätstheorie gehören und aus dieser ihre theoretische Fundierung schöpfen.

Eine zentrale Aufgabe in der Komplexitätstheorie ist die Klassifizierung von Problemen (aus einer Vielzahl von verschiedenen Gebieten) hinsichtlich ihrer Berechnungskomplexität, d.h. ihre Einordnung in Komplexitätsklassen. Eine Komplexitätsklasse enthält alle die Probleme, die mittels eines bestimmten Algorithmentyps bei vorgegebener Beschränkung der Berechnungsressourcen lösbar sind. Genauer gesagt ist mit einem „bestimmten Algorithmentyp“ das abstrakte Modell einer Turingmaschine [Tur36] gemeint, die ein spezifisches Berechnungsparadigma repräsentiert wie z.B. deterministische Berechnung, nichtdeterministische Berechnung, probabilistische Berechnung etc. Beispiele für Berechnungsressourcen sind die Komplexitätsmaße Zeit und Raum. Dabei bedeutet „Zeit“ die Anzahl der zur Lösung des Problems erforderlichen Berechnungsschritte der Turingmaschine, und unter „Raum“ versteht man die Anzahl der zur Problemlösung benötigten Speicherzellen der Turingmaschine.

Berechnungsressourcen werden als Funktionen, die von der Größe der Eingabe abhängen, angegeben. In der vorliegenden Arbeit wird ausschließlich das traditionelle Modell der *worst-case* Komplexität zugrundegelegt, d.h., Ressourcenfunktionen $r(n)$ beschränken von oben die *maximal* zur Problemlösung gestattete Ressource, wobei das Maximum über alle Eingaben der Größe n genommen wird. Die sogenannte *average-case* Komplexität – ein alternatives Modell, das ebenfalls ein nicht unbedeutendes Interesse her-

[1] Einige dieser Gebiete sind inzwischen so eng mit der Komplexitätstheorie verflochten, daß es schwerfällt, eine Grenze zwischen ihnen zu ziehen. Zuweilen hat diese Interaktion sogar neue, dazwischenliegende Teilgebiete geschaffen wie z.B. die komplexitätstheoretische Kryptographie.

vorgerufen hat – wird in dieser Arbeit nicht betrachtet.

Durch Klärung der Inklusionsbeziehungen zwischen den einzelnen Komplexitätsklassen versuchen Komplexitätstheoretiker, die relative Berechnungskraft der zugrundeliegenden Berechnungsparadigmen bzw. Ressourcenbeschränkungen zu ergründen. Der Nachweis, daß ein Problem in einer bestimmten Klasse liegt, ermöglicht die Angabe oberer Schranken für den Rechenaufwand, mit dem dieses Problem gelöst werden kann. Die Bestimmung unterer Schranken – d.h. des zur Problemlösung mindestens erforderlichen Rechenaufwands – kann durch den Nachweis erfolgen, daß das gegebene Problem *nicht* in einer bestimmten Komplexitätsklasse liegt; dies entspricht der Aufgabe, verschiedene Komplexitätsklassen voneinander zu separieren. Leider sind Separationsergebnisse in der Komplexitätstheorie selten und untere Schranken in der Regel sehr schwer zu beweisen.

Komplexitätstheorie ist jedoch nicht nur auf die Klassifizierung von Problemen beschränkt. Durch das Studium der Struktur und der Eigenschaften von Komplexitätsklassen versuchen Komplexitätstheoretiker, Erkenntnisse darüber zu gewinnen, *warum* manche Probleme schwerer zu lösen sind als andere – und nicht nur, *wie* schwer ihre Lösung hinsichtlich des Rechenaufwandes ist. Diese Aufgabe ist besonders angesichts der Tatsache, daß es für viele praktisch wichtige Probleme anscheinend keine effizienten Algorithmen gibt, von entscheidender Bedeutung.

Die fundamentalsten Komplexitätsklassen sind die Klassen P (deterministische Polynomialzeit) und NP (nichtdeterministische Polynomialzeit). Bereits Mitte der sechziger Jahre [Cob64, Edm65] wurde P als die geeignetste Formalisierung des informalen Begriffs der „effizient machbaren“ Berechnung erfasst: Mengen, die Probleme aus P kodieren, sind leicht entscheidbar, d.h., die durch sie repräsentierten Probleme sind leicht lösbar. Andererseits gelten die schwersten Mengen in NP [Coo71, Lev73] als „widerspenstig“ hinsichtlich der Berechnungseffizienz. Seit den Anfängen der Komplexitätstheorie ist das berühmte $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ Problem die zentrale Frage und die größte Herausforderung dieses Gebietes, wenn nicht gar die zentrale Frage und Herausforderung der gesamten Theoretischen Informatik, und es ist auch heute noch ungelöst. Die herausragende Bedeutung dieser Frage – sowohl in der Theorie als auch in der Praxis – ergibt sich aus der ärgerlichen Diskrepanz zwischen der hohen praktischen Signifikanz vieler der gegenwärtig bekannten NP-vollständigen Probleme gegenüber einem absoluten Mangel an effizienten Algorithmen für irgendeines von ihnen, trotz stetiger, intensiver, jahrzehntelanger Anstrengungen, solche Algorithmen in die Hand zu bekommen.

Die vorliegende Habilitationsschrift versteht sich als eine Fortsetzung der Untersuchungen zu Struktur und Eigenschaften der Klassen P und NP – und verwandter Klassen. Die in dieser Schrift angesprochenen Themenkreise können kurz durch die folgenden Stichworte benannt werden, welche im weiteren genauer erläutert werden: *Zertifikatkomplexität,*

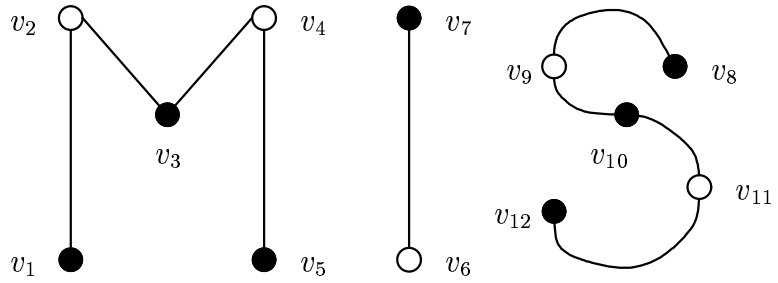

Abbildung 1: Graph G mit maximaler unabhängiger Menge, dargestellt durch volle Kreise

Einwegfunktionen, Heuristiken versus NP-*Vollständigkeit* und *Komplexität eingeschränkter Zähltypen*. Dabei hat das zuletzt genannte Stichwort drei relativ unabhängige Aspekte, die wie folgt kurz beschrieben werden können: (a) *Zähleigenschaften von Schaltkreisen*, (b) *Separationen mit Immunität für Zählklassen* und (c) *Zählen der Lösungen von über einem unären Alphabet kodierten* NP-*Mengen*.

Zur anschaulichen Beschreibung dieser Themen wird das folgende konkrete Beispiel eines Problems in NP verwendet. Zuvor wird der folgende graphentheoretische Begriff benötigt. Für einen gegebenen Graphen G heiße eine Teilmenge I der Knotenmenge von G eine *unabhängige Menge* von G, falls alle Knoten in I paarweise nicht benachbart sind. Die Größe – d.h. die Anzahl der Knoten – einer maximalen unabhängigen Menge von G wird mit $mis(G)$ bezeichnet; $mis(G)$ wird auch als die *Unabhängigkeitszahl* von G bezeichnet.

Abbildung 1 zeigt einen Graphen G, der zwei maximale unabhängige Mengen mit jeweils sieben Knoten besitzt, d.h., $mis(G) = 7$. `Independent Set` ist das folgende Entscheidungsproblem: Gegeben ein Paar $\langle G, k \rangle$, wobei G ein Graph ist und k eine positive natürliche Zahl, ist es wahr, daß $mis(G) \geq k$? `Independent Set` ist ein NP-vollständiges Problem und somit eines der schwersten Probleme in dieser Klasse, denn *jedes* NP-Problem läßt sich in Polynomialzeit in `Independent Set` transformieren. Technisch gesagt ist eine solche Polynomialzeit-Transformation eine polynomialzeitbeschränkte many-one-Reduktion: Für Mengen A und B ist A in Polynomialzeit auf B many-one-reduzierbar, falls es eine polynomialzeitberechenbare Funktion f gibt, so daß für alle $x \in \Sigma^*$ gilt:[2] $x \in A \iff f(x) \in B$. Für eine Komplexitätsklasse $\mathcal{C}$ heißt eine Menge A *$\mathcal{C}$-schwer*,[3] falls jede Menge B aus $\mathcal{C}$ in Polynomialzeit auf A many-one-reduziert werden kann. Ist A sowohl ein Element von $\mathcal{C}$ als auch $\mathcal{C}$-schwer, so heißt A *$\mathcal{C}$-vollständig*.

Für Tausende von Problemen ist mittlerweile gezeigt worden, daß sie NP-vollständig

[2] $\Sigma = \{0, 1\}$ ist das zugrundegelegte binäre Alphabet, und Σ^* bezeichnet die Menge aller Wörter über Σ.

[3] Man beachte aber die hiervon abweichende Verwendung der Sprechweise „$\mathcal{C}$-schwer“, die in Kapitel 6 vereinbart (und nur dort verwendet) wird.

sind; siehe Garey und Johnson [GJ79]. Eigentlich ist es unwesentlich, welches spezielle NP-vollständige Problem wir als Beispiel wählen, denn es ist bekannt, daß wenn irgendein NP-vollständiges Problem in P liegt, dann sind alle NP-vollständigen Probleme in P enthalten. Tatsächlich legt die berühmte Isomorphievermutung von Berman und Hartmanis [BH77] nahe, daß möglicherweise *sämtliche* NP-vollständige Probleme in Wirklichkeit nur ein und dasselbe Problem „in Verkleidung“ sind.

Wir wenden uns nun der detaillierten Beschreibung der einzelnen Themen zu, die in der vorliegenden Habilitationsschrift behandelt werden. Dabei wird insbesondere auf die Motivation und den historischen Hintergrund der aufgeworfenen Fragen und der eingeführten Begriffe eingegangen, und es werden die wichtigsten Ergebnisse der vorliegenden Schrift den aus der Literatur bekannten Ergebnissen gegenübergestellt, mit denen sie in Zusammenhang stehen.

(1) Zertifikatkomplexität – [HRW97a, HRW97b]. Gegeben seien ein Problem $A \in \mathrm{NP}$ und eine NP-Maschine N für A (d.h., $A = L(N)$ ist die von N akzeptierte Sprache), und $x \in A$ sei eine von N akzeptierte Instanz. Ein *Zertifikat* für „$x \in A$“ ist ein Wort $z \in \Sigma^*$, das einen akzeptierenden Berechnungspfad von N bei Eingabe x kodiert. Andere gebräuchliche Bezeichnungen für „Zertifikat“ sind „Zeuge“ und „Lösung“. Für jedes Zertifikat z für „$x \in A$“ gilt: (a) die Länge von z (bezeichnet mit $|z|$) ist polynomiell in $|x|$, und (b) z bezeugt die Zugehörigkeit von x zu A so, daß dies deterministisch in Polynomialzeit verifiziert werden kann.

Betrachten wir beispielsweise das Problem `Independent Set`; N sei die kanonische NP-Maschine für dieses Problem, d.h. der „natürliche“ NP-Algorithmus, der bei Eingabe $\langle G, k\rangle$ alle möglichen Teilmengen der Knotenmenge von G rät, die mindestens k Knoten enthalten, und für jede geratene Teilmenge prüft, ob sie eine unabhängige Menge von G ist, und entsprechend akzeptiert. Ist G beispielsweise der Graph aus Abbildung 1, so ist $\langle G, 7\rangle \in$ `Independent Set`, und die beiden Zertifikate dafür, daß N bei Eingabe $\langle G, 7\rangle$ akzeptiert, sind die (geeignet als Wörter über Σ kodierten) Knotenmengen $\{v_1, v_3, v_5, v_6, v_8, v_{10}, v_{12}\}$ und $\{v_1, v_3, v_5, v_7, v_8, v_{10}, v_{12}\}$. Um die Gültigkeit eines solchen Zertifikats zu prüfen, muß man lediglich verifizieren, daß es sieben Knoten enthält, die paarweise unabhängig sind, was nicht schwer ist, wenn die Nachbarschaftsmatrix des Graphen gegeben ist.

Die Nützlichkeit von Zertifikaten liegt auf der Hand – bzw. liegt begründet in ihren Eigenschaften (a) und (b). Doch wie kann man Zertifikate in die Hand bekommen, wie sie effizient finden? In Kapitel 3 der vorliegenden Schrift werden solche NP-Mengen untersucht, deren Maschinen schwer bzw. leicht zu berechnende Zertifikate haben. Ein Zertifikat heißt *leicht berechenbar*, falls es eine in Polynomialzeit berechenbare Funktion f gibt, so daß für jede Eingabe $x \in A$ der Wert $f(x)$ ein Zertifikat für „$x \in A$“ ist. Ein Zertifikatsystem

(d.h. eine NP-Maschine) N für A heißt *leicht*, falls N für jede akzeptierte Eingabe $x \in A$ leicht berechenbare Zertifikate besitzt.

Insbesondere wird in Kapitel 3 die Klasse $\mathrm{EASY}_{\forall}^{\forall}$ aller der Mengen studiert, deren *sämtliche* Zertifikatsysteme leicht sind. Außerdem wird die Klasse $\mathrm{EASY}_{\forall}^{\exists}$ aller der Mengen untersucht, für die wenigstens ein leichtes Zertifikatsystem *existiert*; es stellt sich heraus, daß die Gleichheit $\mathrm{EASY}_{\forall}^{\exists} = \mathrm{P}$ gilt, und somit gilt die Inklusion $\mathrm{EASY}_{\forall}^{\forall} \subseteq \mathrm{P}$. Analog zu diesen beiden Klassen werden die Klassen $\mathrm{EASY}_{\mathrm{io}}^{\forall}$ und $\mathrm{EASY}_{\mathrm{io}}^{\exists}$ aller der Mengen eingeführt, deren Zertifikatsysteme lediglich für *unendlich viele* Eingaben leicht sein müssen.

Haben leichte Mengen nur leichte Zertifikatsysteme? Mit anderen Worten, gilt $\mathrm{EASY}_{\forall}^{\forall} = \mathrm{P}$? Borodin und Demers [BD76] bewiesen bereits vor mehr als zwei Jahrzehnten, daß die Antwort auf diese Frage negativ ist, außer es tritt ein unwahrscheinlich erscheinender Kollaps von Komplexitätsklassen ein. In der hier verwendeten Notation kann der Satz von Borodin und Demers wie folgt formuliert werden: Wenn $\mathrm{NP} \cap \mathrm{coNP} \neq \mathrm{P}$, dann $\mathrm{P} \not\subseteq \mathrm{EASY}_{\forall}^{\forall}$, wobei coNP die Klasse der Mengen ist, deren Komplement in NP liegt. Das heißt, der Satz von Borodin und Demers sagt, daß es unter Annahme einer sehr plausiblen Hypothese leichte Mengen (d.h. Mengen in P) gibt, die nicht nur leichte Zertifikatsysteme haben. Dieses schöne Ergebnis motivierte die Untersuchungen, die zu den in Kapitel 3 dargestellten Ergebnissen führten.

Insbesondere werden die Klassen $\mathrm{EASY}_{\forall}^{\forall}$ und $\mathrm{EASY}_{\mathrm{io}}^{\forall}$ mit den Mitteln der Kolmogoroff-Komplexität charakterisiert, und somit wird ihre Robustheit gezeigt. Informell gesagt ist die Kolmogoroff-Komplexität ([Kol65, Cha66], siehe auch [LV93]) ein Maß für die „Zufälligkeit“ endlicher binärer Wörter in einem informationstheoretischen Sinn. Hier wird der Begriff der Kolmogoroff-Komplexität verwendet, um die Zufälligkeit von Zertifikaten zu beschreiben. „Zufälligkeit“ bedeutet dabei für ein Zertifikat, daß es „schwer zu finden“ ist, d.h., es kann nicht in Polynomialzeit berechnet werden. Im Gegensatz dazu haben leichte Zertifikatsysteme für jede akzeptierte Eingabe Zertifikate mit kleiner verallgemeinerter Kolmogoroff-Komplexität,[4] die leicht zu finden sind.

Wird beispielsweise ein Zertifikat für „$\langle G, 7\rangle \in$ `Independent Set`“ zufälligerweise durch das Wort kodiert, das nur aus Einsen besteht – 1^n für ein $n \in \mathbb{N}$ –, dann handelt es sich nicht um ein Kolmogoroff-zufälliges Zertifikat, denn dieses Wort läßt sich extrem stark komprimieren – nämlich in ein Wort, daß nur die Anzahl n der Einsen in Binärdarstellung angibt und somit logarithmisch kürzer ist als die ursprüngliche Darstellung des Zertifikats. Es ist jedoch sehr unwahrscheinlich, daß NP-vollständige Mengen wie `Independent Set`

[4] „Zufälligkeit“ im Sinne der Kolmogoroff-Komplexität bedeutet „Nichtkomprimierbarkeit“ im Sinne der Theorie der Datenkomprimierung. Der Begriff der *verallgemeinerten* Kolmogoroff-Komplexität wurde von Hartmanis [Har83] eingeführt; diese mißt nicht nur, wie stark ein binäres Wort komprimiert werden kann, sondern auch, welcher Zeitaufwand nötig ist, um das ursprüngliche Wort aus dem entsprechenden komprimierten Wort wiederzugewinnen.

leichte Zertifikatsysteme haben. Diese Eigenschaft würde wegen $\mathrm{EASY}_{\forall}^{\exists} = \mathrm{P}$ unmittelbar $\mathrm{P} = \mathrm{NP}$ implizieren.

Die vorliegende Arbeit liefert eine Reihe neuer Ergebnisse hinsichtlich der Wahrscheinlichkeit, mit der Separationen der Klassen $\mathrm{EASY}_{\beta}^{\alpha}$, mit $\alpha \in \{\exists, \forall\}$ und $\beta \in \{\forall, \mathrm{io}\}$, von Standard-Komplexitätsklassen wie P, NP, $\mathrm{NP} \cap \mathrm{coNP}$ oder der Klasse aller endlichen Mengen zu erwarten sind. Genauer gesagt werden solche Separationen in Zusammenhang zu anderen komplexitätstheoretischen Bedingungen gesetzt und somit Aussagen über die Größe der EASY-Klassen gemacht. Diese Bedingungen betreffen solche Eigenschaften wie Immunität und Bi-Immunität,[5] P-printability[6] und Kollapse von Komplexitätsklassen.

Diese Ergebnisse haben die Form von Implikationen wie der Satz von Borodin und Demers oder von Äquivalenzen wie der folgende Satz aus Kapitel 3: $\mathrm{EASY}_{\forall}^{\forall} \neq \mathrm{NP}$ genau dann, wenn $\mathrm{P} \neq \mathrm{NP}$. Eine konkrete Konsequenz dieses Satzes ist: Damit *jedes* Zertifikatsystem für `Independent Set` leicht ist, genügt es, daß *irgendein* Zertifikatsystem für `Independent Set` diese Eigenschaft besitzt. Eine andere Konsequenz dieses Satzes ist die Implikation $\mathrm{P} \neq \mathrm{EASY}_{\forall}^{\forall} \implies \mathrm{P} \neq \mathrm{NP}$.

Die bisher genannten Ergebnisse kann man als positive Resultate bezeichnen. Es werden auch negative Resultate bewiesen, die zeigen, daß manche der positiven Resultate optimal sind in dem Sinne, daß die Umkehrung der entsprechenden Implikation nicht in allen relativierten Welten gilt. Ein Beispiel für ein solches negatives Resultat ist der folgende Satz: Es gibt ein Orakel A, so daß $\mathrm{NP}^A \neq \mathrm{P}^A = (\mathrm{EASY}_{\forall}^{\forall})^A$. Dieser Satz sagt, daß die Umkehrung der oben erwähnten Implikation $\mathrm{P} \neq \mathrm{EASY}_{\forall}^{\forall} \implies \mathrm{P} \neq \mathrm{NP}$ nicht in allen Relativierungen gilt. Gleichzeitig verschärft dieser Satz das folgende klassische Resultat von Baker, Gill und Solovay [BGS75]: Es gibt ein Orakel B, so daß $\mathrm{NP}^B \neq \mathrm{P}^B = \mathrm{NP}^B \cap \mathrm{coNP}^B$. Dieses Ergebnis sagt, daß die (triviale) Implikation $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$ nicht robust (d.h. nicht in allen Relativierungen) umkehrbar ist. Da sich, dank des Satzes von Borodin und Demers, die Implikation $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{NP}$ zerlegen läßt in:

$$(1) \qquad \left(\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP} \implies \mathrm{P} \neq \mathrm{EASY}_{\forall}^{\forall}\right) \wedge \left(\mathrm{P} \neq \mathrm{EASY}_{\forall}^{\forall} \implies \mathrm{P} \neq \mathrm{NP}\right),$$

und da die beiden Implikationen in (1) in allen Relativierungen gelten, ist der oben erwähnte Satz tatsächlich stärker als der Satz von Baker, Gill und Solovay. Bemerkenswerterweise ist auch die andere der beiden Implikationen in (1) – d.h. die Aussage des Satzes von

[5] Immunität und Bi-Immunität sind sowohl in der Rekursionstheorie als auch in der Komplexitätstheorie wichtige Begriffe, siehe Rogers [Rog67]. Eine Menge L heißt *immun* gegen eine Komplexitätsklasse $\mathcal{C}$, falls L eine unendliche Menge ist, aber keine der unendlichen Teilmengen von L in $\mathcal{C}$ liegt. L heißt *bi-immun* gegen $\mathcal{C}$, falls sowohl L als auch das Komplement von L *immun* gegen $\mathcal{C}$ ist.

[6] P-printability [HY84] ist ein Begriff, der aus der Theorie der Kolmogoroff-Komplexität und Datenkomprimierung stammt. Eine Menge heißt P-*printable*, falls alle ihre Elemente bis zu einer gegebenen Länge in Polynomialzeit ausgedruckt werden können.

Borodin und Demers – nicht robust umkehrbar. Naor und Impagliazzo [IN88, Proposition 4.2] (siehe auch [CS93, FR94, FFNR96]) haben ein Orakel C konstruiert, so daß gilt: $(\mathrm{EASY}^{\forall}_{\forall})^C \neq \mathrm{P}^C = \mathrm{NP}^C \cap \mathrm{coNP}^C$. Auch dieses Resultat ist wegen (1) eine Verschärfung des Satzes von Baker, Gill und Solovay.

(2) Einwegfunktionen. Die oben gemachten Bemerkungen darüber, daß es für NP-vollständige Mengen trotz ihrer großen praktischen Bedeutung gegenwärtig keine effizienten Algorithmen gibt, könnten den (falschen) Eindruck erweckt haben, daß Berechnungseffizienz das einzige Kriterium dafür sei, was nützlich, wichtig, wünschenswert ist für praktische Anwendungen. Im Gegenteil: Für bestimmte Anwendungen ist gerade die Eigenschaft der Berechnungs*in*effizienz – besonders wenn sie beweisbare Ineffizienz ist – nützlich und wünschenswert. Insbesondere sind hier Anwendungen gemeint, die mit strategischen Erwägungen zu tun haben, mit Sicherheit, mit Gegenspielern, die über eine bestimmte Berechnungskraft verfügen, usw. Solche Anwendungen benötigen Berechnungsineffizienz und verlassen sich auf theoretisch wohlbegründete Resultate, die diese Ineffizienz als ein wesenseigenes Merkmal der betrachteten Berechnungsaufgabe nachweisen. Ein solches Anwendungsgebiet kommt einem sofort in den Sinn: Kryptographie.[7]

In der Kryptographie versucht man, sichere Wege für die Übertragung von verschlüsselten Daten über unsichere Kanäle zu finden. Damit versucht man zu garantieren, daß die übertragenen Informationen sicher sind gegen Lauschangriffe, Fälschung der Urheberschaft usw. Zu diesem Zweck müssen geeignete kryptographische Protokolle entworfen werden, deren Grundbausteine elementare kryptographische Objekte sind wie z.B. Pseudozufallsgeneratoren, Bit-Commitment-Schemata oder Einwegfunktionen. Einwegfunktionen sind Funktionen, die sich leicht berechnen, aber nur schwer invertieren lassen. Im traditionellen Modell der *worst-case* Komplexität versteht man unter „schwer invertierbar“, daß kein Polynomialzeit-Algorithmus in der Lage ist, für jedes Wort z aus dem Bild der Funktion eines der Urbilder von z zu berechnen.[8] Dieser Begriff der komplexitätstheoretischen Einwegfunktion geht auf Grollmann und Selman [GS88] und Ko [Ko85] zurück.

Eine Frage von zentraler Bedeutung in der komplexitätstheoretischen Kryptographie ist die Frage, ob Einwegfunktionen existieren. Es ist bekannt, daß Einwegfunktionen genau dann existieren, wenn $\mathrm{P} \neq \mathrm{NP}$. Deshalb ist eine Antwort auf die Frage der Existenz von

[7] Das ist allerdings nicht das einzige. Beispielsweise wurde ein sehr interessantes Anwendungsgebiet komplexitätstheoretischer Ineffizienzresultate von Bartholdi et al. [BTT89a] erschlossen: Sie zeigten, daß für das Copeland-Wahlsystem – ein in der Praxis heute verwendetes Wahlsystem – eine Manipulation der Wahl vom erforderlichen Rechenaufwand her sehr schwierig ist, nämlich NP-vollständig.

[8] Für die meisten kryptographischen Anwendungen genügt diese Definition nicht. Statt dessen wird das Modell der *average-case* Komplexität zugrundegelegt und gefordert, daß nicht einmal randomisierte Algorithmen in der Lage seien, Einwegfunktionen mit vernünftig beschränkter Fehlerwahrscheinlichkeit zu invertieren. Solche kryptographischen Einwegfunktionen werden hier nicht betrachtet.

Einwegfunktionen nicht zu erwarten, außer man könnte das P $\stackrel{?}{=}$ NP Problem lösen. Daher versucht man wenigstens, die Existenzfrage für verschiedene spezielle Typen von Einwegfunktionen durch geeignete komplexitätstheoretische Bedingungen – wie z.B. durch Separationen von Komplexitätsklassen – zu charakterisieren. Grollmann und Selman [GS88] zeigten, daß es injektive und surjektive Einwegfunktionen (d.h. Einwegpermutationen) genau dann gibt, wenn P $\neq$ UP $\cap$ coUP.[9]

In Kapitel 4 der vorliegenden Schrift werden solche Charakterisierungen für bestimmte wichtige Typen von Einwegfunktionen erhalten. Insbesondere werden Einwegpermutationen betrachtet und die assoziativen Einwegfunktionen, die kürzlich von Rabi und Sherman [RS97] eingeführt wurden.

(2a) Assoziative Einwegfunktionen – [HRa]. Rabi und Sherman [RS97] präsentierten kryptographische Protokolle für *digital signatures* und für *unauthenticated secret-key agreement*, welche von ihnen selbst und von Rivest und Sherman entwickelt wurden. Die kryptographischen Grundbausteine dieser Protokolle sind starke,[10] totale, kommutative (im Falle von *multi-party secret-key agreement*) und assoziative Einwegfunktionen. Obwohl Rabi und Sherman [RS97] bewiesen, daß assoziative Einwegfunktionen genau dann existieren, wenn P $\neq$ NP, ließen sie die Frage offen, ob irgendeine natürliche komplexitätstheoretische Bedingung hinreichend für die Existenz starker, totaler, kommutativer und assoziativer Einwegfunktionen ist. In Kapitel 4 wird diese Frage beantwortet: Starke, totale, kommutative und assoziative Einwegfunktionen existieren genau dann, wenn P $\neq$ NP.

Rabi und Sherman [RS93] fragen auch, ob aus jeder gegebenen Einwegfunktion eine starke, totale, kommutative und assoziative Einwegfunktion konstruiert werden kann. Der Beweis des oben genannten Satzes gibt auch auf diese Frage eine positive Antwort. Es ist bekannt (siehe [GS88]), wie man aus einer beliebigen Einwegfunktion eine Menge in NP $-$ P erhält. Der Beweis des oben genannten Satzes zeigt konstruktiv, wie aus einer beliebigen Menge in NP $-$ P eine starke, totale, kommutative und assoziative Einwegfunktion kreiert werden kann. Ganz konkret ist unter der Annahme P $\neq$ NP z.B. `Independent Set` eine Menge in NP $-$ P, und die Konstruktion einer Einwegfunktion mit den gewünschten Eigenschaften kann mittels einer konkreten – z.B. mittels der weiter oben beschriebenen kanonischen – NP-Maschine für `Independent Set` deterministisch in Polynomialzeit ausgeführt werden.

[9] UP [Val76] ist die Teilklasse von NP, die aus solchen Mengen besteht, deren Maschinen für jede Eingabe höchstens ein Zertifikat erlaubt ist, und coUP ist die Klasse der Mengen, deren Komplement in UP liegt. UP spielt seit langem eine zentrale Rolle in der komplexitätstheoretischen Kryptographie, siehe [GS88]. Kandidaten für Probleme in UP $\cap$ coUP $-$ P sind das (gegeignet als Menge kodierte) Problem, den diskreten Logarithmus zu berechnen [GS88], und das Problem `Primality Testing` [FK92].

[10] Informell gesagt heißt eine zweistellige Einwegfunktion *stark*, falls sie selbst dann schwer invertierbar ist, wenn eines ihrer Argumente gegeben ist.

Ausgehend von Kleenes [Kle52, pp. 327–328] sorgfältiger Unterscheidung zweier Typen von Gleichheit zwischen partiellen Funkionen,[11] die in der Rekursionstheorie bekannt sind als *schwache* bzw. *vollständige Gleichheit,* wird eine neue Definition für den Begriff der Assoziativität partieller zweistelliger Funktionen vorgeschlagen. Es wird argumentiert, daß diese neue Definition einen natürlicheren Begriff der Assoziativität liefert als die von Rabi und Sherman [RS97] verwendete Definition. Dennoch wird gezeigt, daß die Ergebnisse von Rabi und Sherman [RS97] sowie die Ergebnisse aus Kapitel 4 der vorliegenden Schrift sogar unter dieser strengeren Definition gelten.

Ferner wird das Problem der Existenz injektiver assoziativer Einwegfunktionen untersucht. Obwohl Rabi und Sherman [RS97] bewiesen, daß keine totale assoziative Einwegfunktion injektiv sein kann, wird in Kapitel 4 gezeigt, daß nicht-totale, injektive, assoziative Einwegfunktionen genau dann existieren, wenn $\mathrm{P} \neq \mathrm{UP}$. Außerdem wird ein Gegenbeispiel zu einer Konstruktion gegeben, von der Rabi und Sherman [RS97] behaupten, sie konvertiere jede nicht-totale assoziative Einwegfunktion in eine totale. Genauer gesagt wird gezeigt, daß ein Beweis dieser Behauptung sofort $\mathrm{UP} = \mathrm{NP}$ beweisen würde.

(2b) Partielle und totale Einwegpermutationen – [RH96], siehe auch [HRW97b]. Eine Einwegpermutation ist eine injektive und surjektive Einwegfunktion. In Kapitel 4 werden sowohl partielle als auch totale Einwegpermutationen betrachtet.

Interessanterweise hat auch die in Kapitel 3 definierte Klasse $\mathrm{EASY}^{\forall}_{\forall}$ einen engen Bezug zur Existenz von Einwegfunktionen. Fenner et al. [FFNR96] charakterisierten die Existenz surjektiver Einwegfunktionen (welche nicht notwendig injektiv sind) durch die Separation $\mathrm{P} \neq \mathrm{EASY}^{\forall}_{\forall}$. In Kapitel 4 werden die UP und FewP Analoga[12] der Klasse $\mathrm{EASY}^{\forall}_{\forall}$ betrachtet. Diese beiden Klassen werden mit $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ bzw. $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{FewP})$ bezeichnet. Es wird gezeigt, daß Separationen dieser Klassen von P geeignet sind, die Existenz verschiedener Typen von Einwegfunktionen zu charakterisieren. Beispielsweise wird in Kapitel 4 die Existenz von partiellen Einwegpermutationen durch jede der folgenden drei Bedingungen charakterisiert: (1) $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{P}$; (2) $\Sigma^* \notin \mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$; und (3) $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP})$ ist nicht unter Komplementbildung abgeschlossen.

Eine Konsequenz dieser Charakterisierung der Existenz partieller Einwegpermutationen und der oben erwähnten Charakterisierung von Grollmann und Selman ist, daß $\mathrm{EASY}^{\forall}_{\forall}(\mathrm{UP}) \neq \mathrm{P}$ genau dann gilt, wenn $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$. Mit anderen Worten, im Falle des UP-Analogs des Satzes von Borodin und Demers gilt sogar die Umkehrung; diese Aussage wurde von Hartmanis and Hemaspaandra [HH88] direkt bewiesen.

Auch für die Existenz *totaler* Einwegpermutationen wird in Kapitel 4 eine notwendige

[11] Eine *partielle* Funktion ist eine nicht notwendig überall definierte, d.h. eine nicht notwendig totale Funktion.

[12] FewP [All86, AR88] ist die Teilklasse von NP, die aus solchen Mengen besteht, deren Maschinen höchstens polynomiell viele (in der Eingabegröße) Zertifikate für jede Eingabe erlaubt sind.

und hinreichende Bedingung angegeben.

(3) Heuristiken versus NP-Vollständigkeit. NP-Vollständigkeitsresultate werden oft als negative Ergebnisse angesehen, denn sie drücken die Unmöglichkeit aus, praktisch wichtige Probleme effizient zu lösen. In der Praxis ist es daher oft zweckmäßig, heuristische Algorithmen anzuwenden, die schnell sind und in vielen Fällen korrekte Lösungen liefern.

Betrachten wir noch einmal den Beispielgraphen G in Abbildung 1. G hat zufälligerweise nur Knoten mit maximalem Grad 2,[13] und es ist bekannt, daß das `Independent Set` Problem für Graphen, deren Knoten den Grad 2 nicht überschreiten, deterministisch in Polynomialzeit lösbar ist, siehe [GJ79]. Ebenso kann eine gegebene Heuristik dieses Problem korrekt lösen, falls der Eingabegraph eine für diese Heuristik gut geeignete Struktur hat. Doch nun stellt sich die Frage: Für *welche* Eingaben kann eine gegebene Heuristik das Problem lösen?

Der *Minimum-Degree-Greedy* Algorithmus (kurz: MDG-Algorithmus) ist eine bekannte und sehr simple Heuristik zum Finden maximaler unabhängiger Mengen in einem Graphen. Hat der Eingabegraph eine bestimmte einfache Struktur, dann arbeitet der MDG-Algorithmus korrekt. So findet er effizient eine maximale unabhängige Menge des gegebenen Graphen, falls dieser ein Baum ist, ein split-Graph, das Komplement eines k-Baumes, ein wohlüberdeckter Graph oder ein vollständig k-partiter Graph.

Selbst wenn eine Heuristik nicht das Optimum findet, kann sie dennoch praktisch nützlich sein, wenn sie die optimale Lösung des Problems hinreichend gut approximiert. Es ist bekannt, daß der MDG-Algorithmus für bestimmte Klassen von Graphen einen guten Approximationsratio hat [HR94].

Die zentrale Frage in Kapitel 5 ist: *Gegeben seien ein schweres Problem und ein heuristischer Algorithmus für dieses Problem; was ist die Komplexität des Problems, diejenigen Eingaben zu erkennen, für die die Heuristik das gegebene Problem korrekt lösen kann?* Diese Frage kann unter wenigstens drei Gesichtspunkten konkretisiert werden. Diese drei Aspekte werden hier anhand des `Independent Set` Problems und der MDG-Heuristik erläutert. Erstens kann man das *Entscheidungsproblem* `Independent Set` auf diejenigen Graphen einschränken, für die der MDG-Algorithmus eine maximale unabhängige Menge finden kann, und die Komplexität des eingeschränkten Problems untersuchen. Zweitens kann man nach der Komplexität des von einer fixierten Heuristik induzierten *Optimierungsproblems* für `Independent Set` fragen: Wie schwer ist es zu erkennen, für welche Graphen G der MDG-Algorithmus die Unabhängigkeitszahl von G (d.h. die Anzahl $mis(G)$ der Knoten einer maximalen unabhängigen Menge von G) exakt bestimmen kann? Dieselbe Frage kann man für das *Approximierbarkeitsproblem* stellen: Gegeben eine fixierte Konstante r, wie schwer ist es zu erkennen, für welche Graphen G der MDG-Algorithmus die

[13] Der *Grad* eines Knoten v ist die Anzahl seiner Nachbarknoten.

Unabhängigkeitszahl von G in einem Faktor von r approximieren kann?

In Kapitel 5 wird diese Frage für zwei wichtige NP-vollständige Graphenprobleme untersucht, für `Independent Set` und für `K-Colorability`, siehe [GJ79]. Es sei $\chi(G)$ die chromatische Zahl eines Graphen G, d.h. die kleinste Zahl von Farben, mit denen man die Knoten von G so färben kann, daß keine zwei benachbarten Knoten dieselbe Farbe erhalten. `K-Colorability` ist das folgende Entscheidungsproblem: Gegeben ein Paar $\langle G, k\rangle$, wobei G ein Graph ist und k eine positive natürliche Zahl, ist es wahr, daß $\chi(G) \leq k$? Bereits 3-`Colorability`, der Spezialfall dieses allgemeinen Problems für $k = 3$, ist NP-vollständig.

(3a) Unabhängige Mengen – [HR98a], siehe auch [HHR97c]. Bodlaender, Thilikos und Yamazaki [BTY97] untersuchten die Komplexität des oben erwähnten Approximierbarkeitsproblems für `Independent Set` und die MDG-Heuristik: Gegeben eine fixierte Konstante $r \geq 1$, wie schwer ist es zu erkennen, für welche Graphen G der MDG-Algorithmus die Unabhängigkeitszahl von G, $mis(G)$, in einem Faktor von r approximieren kann? (Für den Spezialfall $r = 1$ ergibt sich gerade das oben genannte Optimierungsproblem.) Sie bezeichneten dieses Problem mit $\mathcal{S}_r$ für ein beliebig fixiertes r. Sie zeigten, daß für jede rationale Zahl $r \geq 1$ das Problem $\mathcal{S}_r$ coNP-schwer ist. Sie fanden auch eine obere Schranke für die Komplexität dieses Problems: $\mathcal{S}_r$ ist in P^{NP}, der Klasse der Probleme, die mittels sequentiellem (d.h., „Turing“-) Zugriff auf ein NP-Orakel in Polynomialzeit gelöst werden können. Bodlaender et al. [BTY97] ließen die Frage offen, ob sich die Lücke zwischen oberer und unterer Schranke für die Komplexität von $\mathcal{S}_r$ schließen läßt. Für den Spezialfall $r = 1$ zeigten sie, daß $\mathcal{S}_1$ sogar DP-schwer ist, wobei DP [PY84] die Klasse der Mengen ist, die sich als Differenz zweier NP-Mengen darstellen lassen. Auch in diesem Fall ließen sie die Frage offen, ob $\mathcal{S}_1$ vollständig für DP oder eine größere Klasse wie P^{NP} ist.

In Kapitel 5 werden alle offenen Fragen der Arbeit [BTY97] beantwortet. Es wird gezeigt, daß für jede rationale Zahl $r \geq 1$ das Problem $\mathcal{S}_r$ vollständig für $\mathrm{P}_{||}^{\mathrm{NP}}$ ist, die Klasse der Probleme, die mittels parallelem (d.h., „truth-table“-) Zugriff auf ein NP-Orakel in Polynomialzeit gelöst werden können. Es gilt: $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{DP} \subseteq \mathrm{P}_{||}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Die Klasse $\mathrm{P}_{||}^{\mathrm{NP}}$ ist kürzlich wieder stark in den Blickpunkt des Interesses getreten und hat sich als sehr wichtig für die Bestimmung der Komplexität natürlicher Probleme erwiesen, für die zuvor nur bekannt war, daß sie NP-schwer oder coNP-schwer sind, siehe die Arbeiten [HHR97a, HHR97b, HW97] und den Übersichtsartikel [HHR97c].

(3b) Graphfärbbarkeit – [Rot98a]. Ferner wird in Kapitel 5 die Komplexität gewisser Einschränkungen des Entscheidungsproblems 3-`Colorability` untersucht. Diese Einschränkungen werden durch verschiedene Graphfärbungsheuristiken induziert, beispielsweise durch den sequentiellen Algorithmus, der die Knoten des Graphen in verschiede-

nen Reihenfolgen durchläuft – z.B. in der Reihenfolge, die die Knoten nach fallendem Grad ordnet, oder in der Reihenfolge, die gemäß der rekursiven *smallest-last* Ordnungsprozedur von Matula et al. [MMI72] entsteht. Zu den sieben in Kapitel 5 untersuchten Graphfärbungsheuristiken zählt auch der Algorithmus von Wood [Woo69]. Für jede dieser Heuristiken zum Färben von Graphen wird gezeigt, daß die durch sie induzierte Einschränkung von 3-`Colorability` immer noch NP-vollständig ist.

(4) Komplexität eingeschränkter Zähltypen. Um die Komplexität des Problems, die Permanente einer gegebenen Matrix zu berechnen, präzise beschreiben zu können, führte Valiant [Val79a] die Zählklasse #P ein. Seither sind Zählklassen ein zentraler Untersuchungsgegenstand in der Komplexitätstheorie, und eine Vielzahl nützlicher, wichtiger, manchmal auch unerwarteter Resultate ist erzielt worden, siehe z.B. [Hem87, Hem89, Tod91a, Tod91b, TO92, Tor88, Tor91] und die exzellenten Übersichtsartikel [Sch90, For97]. Traditionell wird NP als eine Klasse angesehen, die den Begriff der existientiellen Quantifizierung verkörpert: Gibt es eine Lösung (m.a.W. ein Zertifikat) für eine gegebene Probleminstanz? Valiants Arbeit und die ihr nachfolgenden Arbeiten fragen dagegen nach der exakten *Anzahl* der Lösungen für eine Probleminstanz. D.h., #P ist die Klasse der Funktionen, die angeben, wieviele Zertifikate eine NP-Maschine bei jeder Eingabe hat. Beispielsweise entspricht der kanonischen NP-Maschine N für `Independent Set`, die weiter oben beschrieben wurde, eine Funktion $f_N \in \#\mathrm{P}$; ist G der Graph aus Abbildung 1, so ergibt sich z.B. $f_N(\langle G, 7\rangle) = 2$. Viele wichtige Zählklassen und probabilistische Klassen wie PP, BPP, $\mathrm{G}\!=\!\mathrm{P}$, $\oplus$P und SPP können elegant mittels #P-Funktionen definiert werden. Das Studium der Klasse #P und der verwandten Klassen hat signifikante Anwendungen in der Schaltkreistheorie und in anderen Gebieten.

(4a) Zähleigenschaften von Schaltkreisen – [HRb, HR98b]. Die Mutter der Komplexitätstheorie ist die Rekursionstheorie. Eines der schönsten und wichtigsten Ergebnisse der Rekursionstheorie ist der Satz von Rice [Ric53, Ric56], der sagt, daß jede nichttriviale Spracheigenschaft der rekursiv aufzählbaren Sprachen unentscheidbar ist. Borchert und Stephan [BS97] gaben den Anstoß für die Suche nach komplexitätstheoretischen Analoga des Satzes von Rice, und sie erzielten erste Ergebnisse hinsichtlich der Komplexität nichttrivialer Zähleigenschaften von Schaltkreisen.

Zur Definition der verwendeten Begriffe: Jede Teilmenge $A \subseteq \mathbb{N}$ der natürlichen Zahlen ist eine *Zähleigenschaft von Schaltkreisen*; gilt $\emptyset \neq A \neq \mathbb{N}$, so heißt A *nichttrivial*. Zu jeder Zähleigenschaft A von Schaltkreisen wird das *Zählproblem für* A definiert und mit $\mathtt{Counting}(A)$ bezeichnet. $\mathtt{Counting}(A)$ ist die Menge aller der (geeignet kodierten) Booleschen Schaltkreise C, für die gilt: Die Anzahl der von C akzeptierten Eingabewörter (aus den 2^n möglichen Eingabewörtern für C, wenn C die Stelligkeit n hat) ist ein Element von A. Wie Borchert und Stephan [BS97] verwenden wir (ausschließlich in Abschnitt

(4a)) die folgende Sprechweise: Wir sagen, eine Zähleigenschaft A von Schaltkreisen ist *$\mathcal{C}$-schwer* für eine Komplexitätsklasse $\mathcal{C}$, falls das Zählproblem für A $\mathcal{C}$-schwer unter Turingreduktionen ist; d.h., falls $\mathcal{C} \subseteq \mathrm{P}^{\mathtt{Counting}(A)}$.

Insbesondere zeigten Borchert und Stephan [BS97], daß jede nichttriviale Zähleigenschaft von Schaltkreisen UP-schwer ist, und daß bestimmte eng verwandte Probleme SPP-schwer sind. SPP [OH93, FFK94], eine Verallgemeinerung der Klasse UP, spielt eine zentrale Rolle in der Komplexitätstheorie und ist insbesondere eng verbunden mit den Abschlußeigenschaften der Klasse #P [OH93].

In Kapitel 6 wird gezeigt, daß die zuerst genannte untere Schranke von Borchert und Stephan – jede nichttriviale Zähleigenschaft von Schaltkreisen ist UP-schwer – nicht zu einer unteren Schranke bezüglich SPP verbessert werden kann, außer es tritt ein unwahrscheinlich erscheinender Kollaps von Komplexitätsklassen auf: Wenn jede nichttriviale Zähleigenschaft von Schaltkreisen SPP-schwer ist, dann gilt $\mathrm{SPP} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Dennoch wird die oben genannte untere Schranke von Borchert und Stephan in Kapitel 6 verbessert: Jede nichttriviale Zähleigenschaft von Schaltkreisen ist Const-schwer. Die Klasse $\mathrm{Const} = \mathrm{P}^{(\#_{\mathrm{const}} \cdot \mathrm{P})[\mathcal{O}(1)]}$ wird in Kapitel 6 als ein Analog der bekannten Klasse Few [CH90] eingeführt. Grob gesagt ist Const die Klasse aller der Sprachen, die in Polynomialzeit mittels konstant vieler Fragen an ein #P-Funktionenorakel entschieden werden können, wobei das #P-Orakel der Einschränkung unterworfen ist, daß es höchstens eine konstante Anzahl verschiedener Werte annehmen darf. Für jedes $k > 0$ ist $\mathrm{UP}_{\leq k}$ definiert als die Klasse aller der NP-Mengen, deren Maschinen für jede Eingabe höchstens k Zertifikate haben dürfen. Weiter sei $\mathrm{UP}_{\mathcal{O}(1)} = \bigcup_{k \geq 1} \mathrm{UP}_{\leq k}$. Es gilt:

$$\mathrm{UP} = \mathrm{UP}_1 \subseteq \mathrm{UP}_2 \subseteq \cdots \subseteq \mathrm{UP}_{\mathcal{O}(1)} \subseteq \mathrm{Const} \subseteq \mathrm{Few} \subseteq \mathrm{SPP}.$$

Außerdem wird in Kapitel 6 gezeigt, daß jede „P-konstruierbar bi-unendliche“ Zähleigenschaft von Schaltkreisen SPP-schwer ist. Eine Menge $A \subseteq \mathbb{N}$ ist *bi-unendlich*, falls sowohl A als auch das Komplement von A in $\mathbb{N}$ unendlich ist. Grob gesagt ist A P-*konstruierbar bi-unendlich*, falls die Eigenschaft der Bi-Unendlichkeit auf konstruktive Weise und effizient (d.h. in Polynomialzeit) verifiziert werden kann.

(4b) Separationen mit Immunität für Zählklassen – [Rot98b, Rot98c]. Ko [Ko90] und Bruschi [Bru92] zeigten unabhängig voneinander, daß in einer geeigneten relativierten Welt PSPACE, die Klasse der in polynomialem Raum entscheidbaren Mengen, eine Menge enthält, die immun ist gegen PH, die Polynomialzeit-Hierarchie von Meyer und Stockmeyer [MS72, Sto77]. Eine Separation von Komplexitätsklassen, die durch eine immune Menge bezeugt wird, ist eine besonders starke Separation, denn eine immune Menge enthält nach Definition höchstens *endliche* Teilmengen aus der Klasse, gegen die separiert wird. Die Eigenschaft der Immunität „schützt“ diese Menge folglich dagegen, „von innen

her approximiert zu werden“ durch unendliche Mengen aus der Klasse, gegen die separiert wird.

In Kapitel 7 der vorliegenden Schrift wird die Frage nach relativierten Separationen *mit Immunität* für PH und die Zählklassen PP, $\mathrm{G{=}P}$ und $\oplus\mathrm{P}$ in allen denkbaren paarweisen Kombinationen gestellt und beantwortet. Diese Ergebnisse verschärfen zuvor bekannte einfache Separationsergebnisse von Torán [Tor91], Green [Gre91] und Berg und Ulfberg [BU], welche nicht über den Vorteil der Immunität verfügen.

Außerdem wird eine relativierte „$\mathrm{G{=}P}$-einfache“ Menge konstruiert. Eine $\mathrm{G{=}P}$-*einfache* Menge ist eine Menge in $\mathrm{G{=}P}$, deren Komplement immun gegen $\mathrm{G{=}P}$ ist. Ähnliche Ergebnisse wurden von Balcázar et al. [Bal85, BR88] für andere Klassen als $\mathrm{G{=}P}$ erhalten.

Die in Kapitel 7 verwendete Beweistechnik erfordert eine exponentielle untere Schranke für die Schaltkreiskomplexität der Booleschen Funktion $\mathrm{EQU}_n^{\mathrm{half}}$, die der Zählklasse $\mathrm{G{=}P}$ entspricht. Die benötigte untere Schranke für $\mathrm{EQU}_n^{\mathrm{half}}$ kann aus einer bekannten unteren Schranke für die Schaltkreiskomplexität der Majoritätsfunktion hergeleitet werden, die von Razborov [Raz87] bewiesen wurde.

(4c) Zählen der Lösungen von über einem unären Alphabet kodierten NP-Mengen – [GOR98a, GOR98b]. Kapitel 8 der vorliegenden Schrift untersucht $\#\mathrm{P}_1$ [Val79b], die Klasse der Funktionen, die angeben, wieviele Zertifikate eine NP-Maschine bei jeder Eingabe hat, falls das Eingabealphabet der NP-Maschine unär ist, d.h., aus nur einem Symbol besteht. $\#\mathrm{P}_1$ ist eine Einschränkung der Klasse $\#\mathrm{P}$ und enthält praktisch wichtige Probleme wie z.B. das Problem `Self-Avoiding Walk`, ein klassisches Problem der Statistischen Physik und der Polymerchemie, siehe Welsh [Wel93]. Valiant zeigte für eine Reihe von Problemen, daß sie $\#\mathrm{P}_1$-vollständig sind. Diese Probleme haben typischerweise die Form: Berechne für eine gegebene natürliche Zahl n in Unärdarstellung die Anzahl der Graphen mit n Knoten, die eine fixierte Grapheigenschaft π erfüllen. Ob `Self-Avoiding Walk` $\#\mathrm{P}_1$-vollständig ist, ist eine seit langem offene Frage.

Die entscheidende Frage bezüglich $\#\mathrm{P}_1$ ist, ob $\#\mathrm{P}_1$ in FP, der Klasse der in Polynomialzeit berechenbaren Funktionen, enthalten ist. In Kapitel 8 wird diese Frage in Bezug zu anderen komplexitätstheoretischen Bedingungen gesetzt. Es folgt unmittelbar aus der Definition, daß alle unär kodierten NP-Mengen in P liegen, falls $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. In Kapitel 8 wird gezeigt, daß unter der Annahme $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ weitere unwahrscheinlich erscheinende Kollapse von Komplexitätsklassen auftreten: $\mathrm{PH} \subseteq \oplus\mathrm{P}$ und $\mathrm{P} = \mathrm{BPP}$.

Außerdem wird gezeigt, daß $\#\mathrm{P}_1$ genau dann in FP enthalten ist, wenn jede Menge in P eine leicht (d.h. in Polynomialzeit) berechenbare „census“-Funktion hat. Die *census*-Funktion einer Menge L, census_L, ordnet jeder natürlichen Zahl n in Unärdarstellung die Anzahl der Wörter in L mit Länge n zu. Der Begriff der census-Funktion ist ein zentraler Begriff in der Komplexitätstheorie, und die Untersuchung von census-Funktionen hat sich in vielfacher Hinsicht als sehr nützlich erwiesen und zu überraschenden Anwendungen

geführt.

Das technische Hauptergebnis in Kapitel 8 ist, daß jede Funktion aus $\#\mathrm{P}_1^{\mathrm{PH}}$ in $\mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}}$ berechnet werden kann. Als Konsequenz ergibt sich, daß die census-Funktion einer jeden Menge in P genau dann leicht berechenbar ist, wenn jede Menge in der Polynomialzeit-Hierarchie diese Eigenschaft besitzt.

Weiterhin wird in Kapitel 8 die Eigenschaft einer Menge, eine leicht berechenbare census-Funktion zu haben, in Bezug zu anderen wichtigen Eigenschaften von Mengen gesetzt. Dazu gehören Eigenschaften wie P-printability, „rankability“ [GS91] (ebenfalls ein Begriff, der aus der Theorie der Kolmogoroff-Komplexität und Datenkomprimierung stammt) und „scalability“ [GH96] (der Abschluß der „rankable“ Mengen unter P-Isomorphie).

Schließlich wird in Kapitel 8 gezeigt, daß die präzise Berechnung der Funktionswerte der census-Funktion einer beliebigen Menge in P nicht wesentlich mehr Berechnungsaufwand erfordert, als diese Werte in einem bestimmten Sinn zu approximieren. Diese Aussage folgt aus dem folgenden Satz: Für fixierte Konstanten α und β gilt, daß wenn jede $\#\mathrm{P}_1$-Funktion n^α-numerierbar ist in Zeit n^β, so ist $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. Der Begriff der $g(n)$-Numerierbarkeit (für Funktionen $g : \mathbb{N} \to \mathbb{N}$) wurde von Cai und Hemaspaandra [CH89] eingeführt, um die Werte von #P-Funktionen approximativ, aber effizient zu berechnen. Grob gesagt ist eine Funktion $f : \Sigma^* \to \Sigma^*$ $g(n)$-numerierbar in Zeit t, falls es einen Turingtransducer gibt, der für alle $n \in \mathbb{N}$ bei Eingabe $x \in \Sigma^n$ in Zeit t eine Liste von höchstens $g(n)$ möglichen Kandidaten für den Funktionswert von $f(x)$ ausgibt, von denen einer der korrekte Wert ist.

Literatur

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 1–11. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[Bal85] J. Balcázar. Simplicity, relativizations and nondeterminism. *SIAM Journal on Computing*, 14(1):148–157, 1985.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Cornell Department of Computer Science, Ithaca, NY, July 1976.

[Ber97] A. Berthiaume. Quantum computation. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 23–51. Springer-Verlag, 1997.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[BR88] J. Balcázar and D. Russo. Immunity and simplicity in relativizations of probabilistic complexity classes. *R.A.I.R.O. Theoretical Informatics and Applications*, 22(2):227–244, 1988.

[Bru92] D. Bruschi. Strong separations of the polynomial hierarchy with oracles: Constructive separations by immune and simple sets. *Theoretical Computer Science*, 102(2):215–252, 1992.

[BS97] B. Borchert and F. Stephan. Looking for an analogue of Rice's Theorem in circuit complexity theory. In *Proceedings on the 1997 Kurt Gödel Colloquium*, pages 114–127. Springer-Verlag *Lecture Notes in Computer Science #1289*, 1997.

[BTT89a] J. Bartholdi III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

[BTT89b] J. Bartholdi III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[BTT92] J. Bartholdi III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical Comput. Modelling*, 16(8/9):27–40, 1992.

[BTY97] H. Bodlaender, D. Thilikos, and K. Yamazaki. It is hard to know when greedy is good for finding independent sets. *Information Processing Letters*, 61:101–106, 1997.

[BU] C. Berg and S. Ulfberg. A lower bound for perceptrons and an oracle separation of the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy. *Journal of Computer and System Sciences*. To appear. A preliminary version appeared in the *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 165–172. IEEE Computer Society Press, 1997.

[CH89] J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, 1989.

[CH90] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic Methodology and Philosophy of Science*, pages 24–30. North Holland, 1964.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[CS93] P. Crescenzi and R. Silvestri. Sperner's lemma and robust machines. In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 194–199. IEEE Computer Society Press, 1993.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 213–222. IEEE Computer Society Press, May 1996.

[FK92] M. Fellows and N. Koblitz. Self-witnessing polynomial-time complexity and prime factorization. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 107–110. IEEE Computer Society Press, June 1992.

[For97] L. Fortnow. Counting complexity. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer-Verlag, 1997.

[FR94] L. Fortnow and J. Rogers. Separability and one-way functions. In *Proceedings of the 5th International Symposium on Algorithms and Computation*, pages 396–404. Springer-Verlag *Lecture Notes in Computer Science #834*, August 1994.

[GH96] J. Goldsmith and S. Homer. Scalability and the isomorphism problem. *Information Processing Letters*, 57(3):137–143, 1996.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[GOR98a] J. Goldsmith, M. Ogihara, and J. Rothe. Tally NP sets and easy census functions. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 483–492. Springer-Verlag *Lecture Notes in Computer Science #1450*, August 1998.

[GOR98b] J. Goldsmith, M. Ogihara, and J. Rothe. Tally NP sets and easy census functions. Technical Report TR 684, University of Rochester, Rochester, NY, March 1998.

[Gre91] F. Green. An oracle separating $\oplus$P from PP^{PH}. *Information Processing Letters*, 37(3):149–153, 1991.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.

[GS91] A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20(3):524–536, 1991.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445. IEEE Computer Society Press, 1983.

[Hem87] L. Hemachandra. *Counting in Structural Complexity Theory*. PhD thesis, Cornell University, Ithaca, NY, May 1987. Available as Cornell Department of Computer Science Technical Report TR87-840.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HH88] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.

[HHR97a] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

[HHR97b] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming*, pages 214–224. Springer-Verlag *Lecture Notes in Computer Science #1256*, July 1997.

[HHR97c] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Raising NP lower bounds to parallel NP lower bounds. *SIGACT News*, 28(2):2–13, June 1997.

[HRa] L. Hemaspaandra and J. Rothe. Creating strong, total, commutative, associative one-way functions from any one-way function in complexity theory. *Journal of Computer and System Sciences*. To appear. Available as: University of Rochester Department of Computer Science Technical Report TR-98-688, May 1998.

[HRb] L. Hemaspaandra and J. Rothe. A second step towards complexity-theoretic analogs of Rice's Theorem. *Theoretical Computer Science*. To appear. Available as: University of Rochester Department of Computer Science Technical Report TR-98-662, July 1997.

[HR94] M. M. Halldorsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 439–448. ACM Press, 1994.

[HR98a] E. Hemaspaandra and J. Rothe. Recognizing when greed can approximate maximum independent sets is complete for parallel access to NP. *Information Processing Letters*, 65(3):151–156, February 1998.

[HR98b] L. Hemaspaandra and J. Rothe. A second step towards circuit complexity-theoretic analogs of Rice's Theorem. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 418–426. Springer-Verlag *Lecture Notes in Computer Science #1450*, August 1998.

[HRW97a] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[HRW97b] L. Hemaspaandra, J. Rothe, and G. Wechsung. On sets with easy certificates and the existence of one-way permutations. In *Proceedings of the Third Italian Conference on Algorithms and Complexity*, pages 264–275. Springer-Verlag *Lecture Notes in Computer Science #1203*, March 1997.

[HW97] E. Hemaspaandra and G. Wechsung. The minimization problem for boolean formulas. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 575–584. IEEE Computer Society Press, November 1997.

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34(1/2):17–32, 1984.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 29–38. IEEE Computer Society Press, June 1988.

[Kle52] S. Kleene. *Introduction to Metamathematics*. D. van Nostrand Company, Inc., New York and Toronto, 1952.

[KMRS97] S. Kurtz, S. Mahaney, J. Royer, and J. Simon. Biological computing. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 179–195. Springer-Verlag, 1997.

[Ko85] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37(1):1–30, 1985.

[Ko90] K. Ko. A note on separating the relativized polynomial time hierarchy by immune sets. *R.A.I.R.O. Theoretical Informatics and Applications*, 24(3):229–240, 1990.

[Kol65] A. Kolmogorov. Three approaches for defining the concept of information quantity. *Prob. Inform. Trans.*, 1:1–7, 1965.

[Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

[LV93] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1993.

[MMI72] D. Matula, G. Marble, and J. Isaacson. Graph coloring algorithms. In R. Read, editor, *Graph Theory and Computing*, pages 109–122. Academic Press, 1972.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory for feasible closure properties. *Journal of Computer and System Sciences*, 46(3):295–325, 1993.

[PY84] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[Raz87] A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mat. Zametki*, 41(4):598–607, 1987. In Russian. English Translation in *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RH96] J. Rothe and L. Hemaspaandra. Characterizations of the existence of partial and total one-way permutations. Technical Report Math/Inf/96/7, Friedrich-Schiller-Universität Jena, Institut für Informatik, Jena, Germany, April 1996.

[Ric53] H. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953.

[Ric56] H. Rice. On completely recursively enumerable classes and their key arrays. *Journal of Symbolic Logic*, 21:304–341, 1956.

[Rog67] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[Rot98a] J. Rothe. Heuristics versus completeness for graph coloring. Technical Report Math/Inf/98/29, Friedrich-Schiller-Universität Jena, Institut für Informatik, Jena, Germany, November 1998.

[Rot98b] J. Rothe. Immunity and simplicity for exact counting and other counting classes. In *Proceedings of the 9th International Conference on Computing and Information*, pages 279–286, June 1998. The conference proceedings are to appear as a special issue of the *Journal of Computing and Information*.

[Rot98c] J. Rothe. Immunity and simplicity for exact counting and other counting classes. Technical Report TR 679, University of Rochester, Rochester, NY, January 1998.

[RS93] M. Rabi and A. Sherman. Associative one-way functions: A new paradigm for secret-key agreement and digital signatures. Technical Report CS-TR-3183/UMIACS-TR-93-124, Department of Computer Science, University of Maryland, College Park, Maryland, 1993. Available on-line at http://www.cs.umbc.edu/pub/REPORTS/cs-93-18.ps.

[RS97] M. Rabi and A. Sherman. An observation on associative one-way functions in complexity theory. *Information Processing Letters*, 64(2):239–244, 1997.

[Sch90] U. Schöning. The power of counting. In A. Selman, editor, *Complexity Theory Retrospective*, pages 204–223. Springer-Verlag, 1990.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1977.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Tod91a] S. Toda. *Computational Complexity of Counting Complexity Classes*. PhD thesis, Tokyo Institute of Technology, Department of Computer Science, Tokyo, Japan, 1991.

[Tod91b] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tor88] J. Torán. *Structural Properties of the Counting Hierarchies*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1988.

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):753–774, 1991.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2*, 42:230–265, 1936. Correction, *ibid*, vol. 43, pp. 544–546, 1937.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20–23, 1976.

[Val79a] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[Val79b] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[Wel93] D. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[Woo69] D. Wood. A technique for coloring a graph applicable to large scale timetabling problems. *The Computer Journal*, 12:317–319, 1969.