Über strukturelle Gemeinsamkeiten der Aufzählbarkeitsklassen von Turingmaschinen und endlichen Automaten

Till Tantau

International Computer Science Institute
1947 Center St. Suite 600
Berkeley, California 94704, USA

tantau@icsi.berkeley.edu

Abstract: Der Beitrag enthält eine Zusammenfassung der Dissertation »On Structural Similarities of Finite Automata and Turing Machine Enumerability Classes«.

Die Dissertation [Ta03] eröffnet mit folgenden Worten:

> **My Thesis:** *The enumerability and verboseness classes of finite automata on the one hand and of Turing machines on the other hand share numerous structural properties. They do not share these properties with intermediate resource-bounded computational models. Especially the finite automata versions of these classes have applications in areas unrelated to enumerability. Two methods that are used for the proofs of the structural similarities – elementary definitions of regular relations and branch diagonalisation – will be applicable to other proofs in automata, complexity, and recursion theory.*

Auf den folgenden 150 Seiten wird dann versucht, mittels etwa sechzig Sätzen, Korollaren und Lemmata diese These zu belegen und den Leser oder die Leserin von ihrer Richtigkeit zu überzeugen – nicht mehr und nicht weniger. Es wird sogar frech behauptet, man könne sich die weitere Lektüre der Arbeit sparen, sobald man von der »Thesis« überzeugt ist.

Mein Hauptanliegen im vorliegenden Beitrag ist zu erläutern, was die obige These eigentlich genau besagt. Mathematische Exaktheit steht dabei weniger im Vordergrund als eine möglichst allgemein verständliche Darstellung der Ideen. Dazu werden zunächst die *Konzepte* wie Aufzählbarkeit oder Verboseness-Klassen genauer erklärt, dann die zentralen *Resultate* beschrieben, eine neue *Beweismethode* kurz angedeutet und zum Schluss *Anwendungen* skizziert.

1 Die Konzepte

1.1 Turingmaschinen, endliche Automaten und Polynomialzeitmaschinen

Schon der Titel der Dissertation verrät, dass es in ihr um Gemeinsamkeiten der beiden ältesten Maschinenmodelle der Informatik geht: *Turingmaschinen* und *endliche Automaten*. Grob gesprochen ist das Turingmaschinenmodell das mächtigste »sinnvolle« Maschinenmodell, das es gibt. »Mächtig« bedeutet hier, dass Turingmaschinen alles berechnen können, was sich überhaupt berechnen lässt. Da unklar ist, was die richtige Definition von »was sich überhaupt berechnen lässt« ist, ist es kein mathematischer Satz, dass Turingmaschinen allmächtig sind, sondern lediglich eine These, die aber der klangvollen Namen »Church'sche These« trägt. Praktisch kann man sich eine Turingmaschine als einen Rechner vorstellen, der über beliebig viel Speicher verfügt und der beliebig lange rechnen darf.

Am anderen Ende des Spektrums der Maschinenmodelle finden sich die endlichen Automaten. Diese verarbeiten ihre Eingabe, indem sie sie einmal von links nach rechts lesen und am Ende sofort eine Ausgabe produzieren. Endliche Automaten benötigen keinerlei Speicher und ihre Rechenzeit ist proportional zur Eingabelänge.

Vom praktischen Standpunkt aus sind Turingmaschinen viel zu mächtig, da sie Probleme spielend lösen können, die nachweislich nicht in vertretbarer Zeit (wie beispielsweise der zu erwartenden Lebensdauer des Universums) zu lösen sind. Endliche Automaten sind nicht mächtig genug, da sie manche Probleme nachweislich nicht lösen können, die ein Taschenrechner mit links löst. In der Komplexitätstheorie werden deshalb Modelle wie *Polynomialzeitmaschinen* betrachtet, die »in der Mitte liegen« und die die Fähigkeiten realer Computer realistischer modellieren.

1.2 Die Aufzählbarkeit von Funktionen

Das erste in der These erwähnte Konzept ist die *Aufzählbarkeit* (enumerability) von Funktionen. Der Begriff »Funktion« dient hierbei, wie so oft in der theoretischen Informatik, lediglich als bequeme mathematische Formalisierung von »Problemen« verschiedenster Art. Funktionen bilden Eingabeinstanzen auf Lösungen ab. Ein sehr breites Spektrum an Problemstellungen lässt sich so formalisieren: Touristen (insbesondere Studenten) sind häufig an der Funktion interessiert, die Stadtpläne mit eingezeichneten Sehenswürdigkeiten auf den Preis der günstigsten Tour abbildet, die alle Sehenswürdigkeiten besucht. Kriminalisten interessieren sich für die Funktion, die DNS-Fingerabdrücke auf die Personen abbildet, denen sie gehören. Für einen Biologen oder Chemiker kann die Funktion interessant sein, die die Aminosäurensequenz eines Proteins auf die Faltungsstruktur des Proteins abbildet, genauer auf die Raumkoordinaten der Atome.

Unglücklicherweise sind viele wichtige Funktionen nach heutigem Wissensstand nur sehr schwer zu berechnen. Glücklicherweise ist dies oft »nicht so schlimm«, da die Funktionen gar nicht exakt berechnet werden müssen, sondern Approximationen der exakten Werte ausreichend sind – so interessieren den Touristen weniger die exakten Kosten der optima-

len Tour, er wird sich sicherlich auch mit einer um beispielsweise maximal zehn Prozent teureren Schätzung zufrieden geben.

In der klassischen Approximationstheorie muss eine Approximation »nahe« an der gesuchten Lösung in folgenden Sinne sein: Sie darf die exakte Lösung nur um einen, möglichst kleinen, konstanten Faktor übertreffen. Falls dieser Faktor beispielsweise 1,1 ist, was 10% entspricht, und falls der Algorithmus im Handgepäck des Touristen eine Approximation von 22 Euro liefert, so bedeutet dies, dass die echten Kosten irgendwo zwischen 20 und 22 Euro liegen. Um es ganz mathematisch auszudrücken: die (auf ganze Euro gerundeten) Kosten sind ein Element der Menge $\{20, 21, 22\}$.

An dieser Stelle kommt nun das Konzept der Aufzählbarkeit ins Spiel. Die Idee ist, allgemeinere Mengen als lediglich Zahlenintervalle als »Approximationen« zuzulassen. Ein *Aufzähler* für eine Funktion ist ein Algorithmus, der für Eingabeinstanzen möglichst kleine Mengen von möglichen Lösungen berechnet. Einzige Bedingungen sind, dass die optimale Lösung in der Menge ist und dass die Menge klein ist. Hat sie immer höchstens m Elemente, so spricht man von einem m-Aufzähler. Ein Vorteil dieses Konzepts gegenüber klassischen Approximationen ist, dass sich so auch Funktionen »approximieren« lassen, die ihre Eingaben gar nicht auf Zahlen abbilden. Für DNS-Fingerabdrücke macht eine »10%-Approximation« an den Besitzer des Fingerabdrucks wenig Sinn, eine dreielementige Menge von möglichen Besitzer hingegen durchaus. Genauso macht eine kleine Menge von möglichen Faltungen eines Proteins mehr Sinn als eine »10%-Approximation« an die Faltung.

Welche Funktionen lassen sich leicht aufzählen, welche sind schwierig? Intuitiv ist es um so leichter eine Funktion aufzählen, je eingeschränkter ihr Wertebereich ist. Ein (recht unnützes) Bespiel ist ein 2-Aufzähler für Funktionen, die überhaupt nur zwei unterschiedliche Werte annehmen können wie »Ja« oder »Nein«: Er gibt einfach immer »›Ja‹ oder ›Nein‹« aus – was mit Sicherheit richtig ist, mit Sicherheit eine kleine Menge von Möglichkeiten ist und mit Sicherheit völlig nutzlos ist. Spannender ist die Frage, welche Funktionen mit, sagen wir, vier unterschiedlichen möglichen Werten sich beispielsweise 3-aufzählen lassen – dazu später mehr. Fasst man alle Funktionen zu einer Klasse zusammen, die sich von einer bestimmten Art Maschine wie Turingmaschinen oder endliche Automaten m-aufzählen lassen, so entsteht eine *Aufzählbarkeitsklasse*. Das Verhältnis dieser Klassen untereinander und ihre Feinstruktur sind Hauptforschungsobjekte der Dissertation. Die Aufzählbarkeitsklassen von Turing- und Polynomialzeitmaschinen sind wohluntersucht. Neu ist die Benutzung von endlichen Automaten als Aufzähler.

Was »praktische« Funktionen wie die weiter oben erwähnten angeht, weiß man, dass viele sich aller Voraussicht nach nicht effizient aufzählen lassen. Genauer weiß man, dass für sie das Aufzählen zumindest nicht leichter ist, als den korrekten Funktionswert zu bestimmen, was wiederum allgemein als schwierig angesehen wird. So haben beispielsweise Cai und Hemaspaandra [CH89] gezeigt, dass es genauso schwierig ist, mit einer Polynomialzeitmaschine eine Menge von Möglichkeiten für die Anzahl der erfüllenden Belegungen einer aussagenlogischen Formel zu bestimmen, wie diese Anzahl exakt zu berechnen.

1.3 Verboseness

Das zweite zentrale Konzept, das auch im Titel der Dissertation vorkommt, ist *Verboseness*, was so viel wie »Geschwätzigkeit« bedeutet. Die Verboseness einer formalen Sprache beschreibt, wie gut sich ihre n-fache charakteristische Funktion m-aufzählen lässt. Um dies anschaulich zu machen, stelle man sich einen großen Sack vor, der mit beschrifteten Kugeln gefüllt ist. Jede Kugel ist mit einem Wort beschriftet und hat eine Farbe: schwarz oder weiß. Ob eine Kugel schwarz oder weiß ist, hängt (in eventuell sehr komplizierter Weise) von ihrer Beschriftung ob. So könnten alle Kugeln schwarz sein, deren Beschriftung eine Primzahl ist oder deren Beschriftung eine Aminosäurensequenz ist, die ein Enzym kodiert. Einen solchen Sack mit schwarzen und weißen Kugeln bezeichnet man in der theoretischen Informatik als *formale Sprache*, wobei der Sack für jedes überhaupt denkbare Wort eine Kugel enthält, was unendlich viele macht.

Um nun die »Geschwätzigkeit« einer formalen Sprache zu quantifizieren, stelle man sich vor, jemand würde n Kugeln aus dem Sack ziehen und uns die Beschriftungen mitteilen, nicht aber die Farben. Unsere Aufgabe als Aufzähler (beziehungsweise die Aufgabe einer aufzählenden Maschine) ist es, die richtigen Farben der Kugeln anzugeben oder, falls das zu schwierig ist, eine möglichst kleine Menge von Möglichkeiten für die Farbverteilung. Da es nur 2^n mögliche Farbverteilungen gibt, besteht eine intellektuelle (oder algorithmische) Herausforderung überhaupt nur dann, wenn wir höchstens $m < 2^n$ Möglichkeiten ausgeben dürfen. Formale Sprachen, für die Maschinen einer bestimmten Bauart zu je n Worten eine Menge von m Möglichkeiten berechnen können, heißen (m,n)-*verbose* in Bezug auf das Maschinenmodell.

Die Untersuchung von Verboseness birgt einige Überraschungen. Eine vom praktischen Standpunkt aus wichtige formale Sprache ist das aussagenlogische Wahrheitsproblem. Hier sind die Kugeln weiß, die mit einer tautologischen aussagenlogischen Formel beschriftet sind wie »$p \vee \neg p$«. Drei unabhängige Forschergruppen haben 1995 gezeigt, dass diese Sprache effizient entschieden werden kann, falls sie in Bezug auf Polynomialzeitmaschinen $(2^n - 1, n)$-verbose ist für irgendein n. Ebenso interessant ist die Verboseness des allgemeinen Wahrheitsproblems. In diesem »Sack voll Wahrheit« sind diejenigen Kugeln weiß, die mit einer wahren mathematischen Aussage beschriftet sind wie »$\forall n > 2\ \forall a,b,c > 0\colon a^n + b^n \neq c^n$«. Das Hauptresultat der Dissertation von Richard Beigel [Be87] ist, dass die Wahrheit für kein n in Bezug auf Turingmaschinen (n,n)-verbose ist, für alle n aber $(n+1,n)$-verbose.

1.4 Kardinalitätsberechnungen

Das dritte zentrale untersuchte Konzept sind Kardinalitätsberechnungen, die in ihrer allgemeinen Form von Gasarch [Ga91] eingeführt wurden und die direkt oder indirekt Anwendungen finden in verschiedenen Beweisen in der Rekursions- und Komplexitätstheorie. Es steht nicht im Vordergrund, *welche* Eingabewörter in einer formalen Sprache sind, sondern *wie viele*. Es werden also n Kugeln gezogen und es soll eine möglichst kleine Men-

ge an Möglichkeiten für die *Anzahl* an weißen Kugeln bestimmt werden. Verglichen mit dem obigen Problem, die exakten Farben zu bestimmen, erscheint dies zunächst einfacher: statt 2^n gibt es nun a priori lediglich $n+1$ Möglichkeiten (keine Kugel ist weiß, eine ist weiß, ..., n Kugeln sind weiß).

Das wichtigste Resultat über Kardinalitätsberechnungen ist der Kummer'sche *Kardinalitätssatz* [Ku92]. Danach lässt sich die Kardinalitätsfunktion der Menge der wahren mathematischen Aussagen durch Turingmaschinen nicht n-aufzählen. Mit anderen Worten: Kein Computer schafft es für irgendein n, bei Eingabe von je n mathematischen Aussagen eine Menge von n Möglichkeiten für die Anzahl der wahren Aussagen unter ihnen zu berechnen. Dies ist um so erstaunlicher, als es ja überhaupt nur $n+1$ Möglichkeiten gibt.

In der Dissertation werden folgende Fragen in Bezug auf »Aufzählbarkeit«, »Verboseness« und »Kardinalitätsberechnungen« untersucht:

1. Was sind die strukturellen Eigenschaften von Aufzählbarkeitsklassen für verschiedene Berechnungsmodelle? Wie hängt die Struktur vom Berechnungsmodell ab?
2. Wie sieht die Inklusionsstruktur der Verbosenessklassen für verschiedene Berechnungsmodelle aus? Für welche Zahlen m, n, h und k sind alle (m,n)-verbosen formalen Sprachen auch (h,k)-verbose?
3. Welche Sätze der Rekursionstheorie in Bezug auf Verboseness und Kardinalitätsberechnungen (wie der Kardinalitätssatz) gelten auch für andere Berechnungsmodelle?
4. Welche theoretischen und praktischen Anwendungen von Aufzählbarkeit gibt es?

2 Die Resultate

Eine zentrale Behauptung der Dissertationsthese ist, dass die Struktur der Aufzählbarkeitsklassen von Turingmaschinen und von endlichen Automaten gleich ist, die Struktur bei ressourcenbeschränkten Turingmaschinen aber andersartig ist als bei diesen beiden. Diese Behauptung wird in der Dissertation durch folgende Ergebnisse untermauert:

1. Sowohl für Turingmaschinen als auch für endliche Automaten gilt der *Kreuzproduktsatz*: Ist das Kreuzprodukt zweier Funktionen $(n+m)$-aufzählbar, so die erste Funktion n-aufzählbar oder die zweite m-aufzählbar. Das *Kreuzprodukt* von Funktionen nimmt zwei Eingaben entgegen und wendet die erste Funktion auf die erste Eingabe an und parallel die zweite Funktion auf die zweite. Der Kreuzproduktsatz gilt nicht für Polynomialzeitmaschinen: Es gibt Funktionen, deren Kreuzprodukt sich 2-aufzählen lässt durch eine Polynomialzeitmaschine, die sich aber jeweils einzeln nicht durch Polynomialzeitmaschinen 1-aufzählen lassen.
2. Die filigranen Inklusionsstrukturen der Verbosenessklassen von Turingmaschinen und der von endlichen Automaten sind gleich. Dieses Resultat folgt aus dem Kreuzproduktsatz und einer Astdiagonalisierung, siehe unten. Wiederum verhalten sich

Polynomialzeitmaschinen anders und ihre Verbosenessklassen weisen eine andere Inklusionsstruktur auf. Dies ist verblüffend in Anbetracht der folgenden Überlegung: Für Turingmaschinen hat die unendliche Inklusionsstruktur eine bestimmte, recht komplizierte Form. Senkt man die Rechenkraft der erlaubten Maschinen ab, so ändert sich die Form der Struktur stark, bleibt dann aber für alle in der Komplexitätstheorie betrachteten Maschinenmodelle gleich. Senkt man die Rechenkraft noch weiter ab, bis man bei endlichen Automaten ankommt, so ändert sich die Form der Struktur erneut. Dass sich gerade wieder dieselbe Struktur wie bei den so viel mächtigeren Turingmaschinen einstellt, wäre eigentlich nicht zu erwarten.

3. Verschiedene »schwache« Formen von Kummers Kardinalitätssatz gelten auch für endliche Automaten. So gilt der Satz beispielsweise für $n = 2$. Keine der schwachen Formen (geschweige denn der Satz selbst) gilt im Polynomialzeitfall.

3 Die Methoden

Die Art und Weise wie ein Satz bewiesen wird ist oft genauso spannend wie der Satz selbst. Der Hauptsatz über die identischen Inklusionsstrukturen der Verbosenessklassen von Turingmaschinen und von endlichen Automaten könnte ohne eine Astdiagonalisierung wohl nicht bewiesen werden. Die Idee hinter diesem neuen Verfahren wird im Folgenden angedeutet.

Die allgemeine Idee der Diagonalisierung reicht weit zurück, letztendlich bis auf das Lügnerparadoxon. Wenn jemand sagt »ich lüge«, so ist es schwierig, dieser Aussage glauben zu schenken, aber ebenso möchte man ihr eigentlich zustimmen. Ganz ähnlich verstrickt man sich schnell in Widersprüche, wenn man mit dem Wissen, dass »der Barbier von Sevilla alle Männer rasiert, die sich nicht selbst rasieren«, schließen möchte, ob sich der Babier nun selbst rasiert oder nicht.

Alan Turing wendete ein solches Paradoxon auf Maschinen an. Dazu ließ er Maschinen »über sich selber reden«, indem er eine formale Sprache nach folgender Regel konstruierte: Um zu entscheiden, ob eine Kugel weiß oder schwarz sein soll, interpretiere man ihre Beschriftung als Programmtext. Nun führe man diesen Programmtext *mit sich selbst als Eingabe* aus und warte das Ergebnis ab. Wenn das Programm anhält und »weiß« ausgibt, so färbe man die Kugel schwarz, anderenfalls weiß. Für die so entstandene formale Sprache kann kein Programm die Farben der Kugeln korrekt berechnen, denn dies würde sofort zu einem Paradoxon à la »ich lüge« führen. Turings bahnbrechende Folgerung war, dass man mittels Computern nicht entscheiden kann, ob ein beliebiges Programm anhalten und »weiß« ausgeben wird oder nicht. Das Verfahren wird »Diagonalisierung« genannt in Gedenken an Cantors »Diagonalen-Beweis« der Überabzählbarkeit des Kontinuums.

Die Diagonalisierungsmethode ist im Laufe der Jahre weiter entwickelt worden und wird heute in der Rekursions- und Komplexitätstheorie vielseitig angewandt. Die modernen Verfahren beruhen letztendlich alle auf demselben Prinzip wie schon Turings erste Diagonalisierung. Die Weiterentwicklung besteht darin, dass bei den modernen Methoden die Maschinen oft nicht mehr »immer über sich selbst reden«, sondern beispielsweise

»über den iterierten Logarithmus ihres eigenen Kodes reden« (bei der Supersparse-Sets-Methode) oder »in der Regel über sich selbst reden, aber manchmal auch über andere Maschinen, wenn dies wichtiger ist, aber nur endlich oft« (bei der Finite-Injury-Methode) oder »in der Regel über sich selbst und manchmal auch über wichtigere Dinge reden, das auch vielleicht unendlich oft, aber irgendwann mal auch nicht« (bei der Infinite-Injury-Methode).

Astdiagonalisierung, das in der Dissertation neu eingeführte Verfahren, unterscheidet sich von den etablierten Verfahren dadurch, dass es in Verbindung mit endlichen Automaten verwendet werden kann. Bei den bekannten Verfahren reden die Maschinen in der einen oder anderen Form über sich selbst und damit insbesondere über den Kode von Maschinen. Endlichen Automaten fällt es hingegen schwer, auch nur einfache Berechnungen auszuführen. Insbesondere können sie *nicht* den Kode von Automaten wie sich selbst verarbeiten, um dann Aussagen darüber zu treffen.

Bei der Astdiagonalisierung wird dieses Problem umgangen, indem nicht der Kode einer Maschine verarbeitet wird, sondern Folgen von Diagonalisierungsentscheidungen. Dies funktioniert wie folgt: Man stelle sich alle Turingmaschinen nebeneinander in einer unendlichen Reihe aufgestellt vor. Wieder konstruieren wir eine formale Sprache, diesmal aber nach folgender Regel: Um die Farbe einer Kugel festzulegen, geben wir den ersten Buchstaben des Wortes auf der Kugel der ersten Maschine in der Reihe zur Berechnung; dann geben wir die ersten beiden Buchstaben der zweiten Maschine; und so weiter. Die Buchstaben des Wortes seien jeweils »s« oder »w«. Falls die erste Maschine in der Reihe etwas *anderes* berechnet als den ersten Buchstaben (also »schwarz« statt »w« oder »weiß« statt »s«) und die zweite Maschine etwas *anderes* als den zweiten Buchstaben und so fort, so färben wir die Kugel entsprechend des letzten Buchstabens. In allen anderen Fällen färben wir die Kugel schwarz.

Die so entstandene formale Sprache kann wiederum von Computern nicht korrekt entschieden werden. Entscheidend ist aber, dass sogar ein endlicher Automat interessante Aussagen über die Färbung von *mehreren* Kugeln machen kann. Nehmen wir beispielsweise zwei Kugeln her. Falls keines der beschriftenden Worte ein Präfix des anderen ist, was sich leicht mit einem endlichen Automaten überprüfen lässt, so wurde *höchstens eines der Worte* weiß gefärbt. Von den vier möglichen Färbungen kann also eine ausgeschlossen werde. Eine genauere Analyse zeigt, dass man auch leicht eine Möglichkeit ausschließen kann, wenn eines der Worte ein Präfix des anderen ist. Die entstandene formale Sprache ist folglich $(3,2)$-verbose in Bezug auf endliche Automaten – und das, obwohl sie von keinem Computer entschieden werden kann. Dies ist ein erstes starkes Resultat über die Inklusionsstruktur von Verbosenessklassen. Mit ähnlichen, komplizierteren Argumenten kann man »eine Hälfte« der Charakterisierung der Verbosenessklassen von endlichen Automaten beweisen; »die andere Hälfte« benötigt den Kreuzproduktsatz.

Astdiagonalisierung ist nicht universell einsetzbar. In Fällen, wo sie aber eingesetzt werden kann, liefert sie starke Trennungsresultate. Ein Beispiel aus einem anderen Bereich als Verboseness ist das folgende: Mit einer Astdiagonalisierung lässt sich zeigen, dass der Schnitt zweier p-selektiver Sprachen nicht semirekursiv sein muss. Ein früheres Resultat von Hemaspaandra und Jiang [HJ95] ist schwächer und wurde trotzdem wesentlich aufwendiger bewiesen mittels eines Supersparse-Sets-Arguments.

4 Die Anwendungen

Der Hauptfokus der Dissertation liegt auf einer mathematisch strikten Strukturanalyse von Aufzählbarkeitsklassen. Direkte Anwendungen solcher Grundlagenforschung darf man eigentlich nicht erwarten. Deshalb ist es interessant zu sehen, in welch unterschiedlichen Situationen Aufzählbarkeit auftaucht und welche unterschiedlichen Folgerungen man aus den Strukturresultaten ziehen kann.

Trennungssätze. Die erste Anwendung der Resultate über Aufzählbarkeit ist »innertheoretisch« und betrifft Trennbarkeit. Die Trennbarkeit von formalen Sprachen ist ein wohluntersuchtes Konzept. Die Idee hierbei ist, dass zwei disjunkte formale Sprachen A und B schwierig sein können, sich aber eventuell »einfachere« disjunkte formale Sprache A' und B' finden lassen, die sie »trennen«. Die trennenden Sprachen sind einfach Obermengen der Sprachen A und B. So könnten beispielsweise A und B beides sehr schwierige formale Sprachen sein, die Beschriftungen der weißen Kugeln in A aber nur Kleinbuchstaben benutzen, die der weißen Kugeln in B nur Großbuchstaben. Dann lassen sich die Sprachen mittels der sehr einfachen Sprachen trennen, bei denen *alle* Kugeln weiß sind, die mit Klein- beziehungsweise mit Großbuchstaben beschriftet sind. Ist hingegen A die formale Sprache der wahren mathematischen Aussagen und B die Sprache der falschen mathematischen Aussagen, so lassen sich A und B nicht trennen – sie sind zu »dicht«, es »passt nichts zwischen sie«.

Sowohl Kummers Kardinalitätssatz als auch die schwachen Formen des Kardinalitätssatzes für endliche Automaten lassen sich äquivalent als (verallgemeinerte) Trennbarkeitsaussagen formulieren. Interessant ist dabei, dass in den Umformulierungen »Aufzählbarkeit« oder »Kardinalitätsfunktionen« nicht mehr vorkommen.

Betrachten wir als Beispiel eine äquivalente Formulierung des so genannten eingeschränkten Kardinalitätssatzes. Für eine formale Sprache A bezeichne $A^{\binom{n}{k}}$ die Menge aller Tupel von n verschiedenen Worten, so dass genau k von ihnen in der Sprache sind (n unterschiedliche Kugeln, von denen genau k weiß sind). Der eingeschränkte Kardinalitätssatz lässt sich nun wie folgt formulieren: Eine Sprache ist rekursiv (von Computern entscheidbar), falls für irgendein n die Sprachen $A^{\binom{n}{0}}$ und $A^{\binom{n}{n}}$ durch rekursive Sprachen getrennt werden können. Nach der Dissertationsthese, die die strukturelle Gleichheit von Turingmaschinen und endlichen Automaten in Bezug auf Aufzählbarkeit postuliert, dürfen wir erwarten, dass die Aussage auch wahr ist, wenn überall »rekursiv« durch »regulär« (von einem endlichen Automaten entscheidbar) ersetzt wird. Genau dies ist auch der Fall. Ebenso im Einklang mit der Dissertationsthese steht, dass dieses Trennungsresultat nicht für Polynomialzeitmaschinen gilt: Für $n \geq 2$ gibt es beliebig komplexe formale Sprachen A, für die $A^{\binom{n}{0}}$ und $A^{\binom{n}{n}}$ von Polynomialzeitmaschinen getrennt werden können.

Als zweites Beispiel betrachten wir den Kardinalitätssatz für endliche Automaten für $n = 2$. Dieser lässt sich wie folgt äquivalent formulieren: Eine formale Sprache A ist regulär, falls es reguläre Obermengen von $A \times A$, $A \times \bar{A}$ und $\bar{A} \times \bar{A}$ gibt, deren Schnitt leer ist. Hierbei ist $\bar{A}$ das Komplement von A (schwarze Kugeln sind weiß und umgekehrt) und eine Menge von Wortpaaren heißt regulär, wenn sie von einem Zweibandautomaten

entschieden werden kann, der seine Eingaben synchron liest.

Zunächst hört sich die Umformulierung des Kardinalitätssatzes wie ein typischer Satz der Automatentheorie an – schließlich geht es um recht einfache und grundlegende Eigenschaften regulärer Mengen. Der Beweis ist aber erstaunlich kompliziert und schon eine kleine Änderung in der Behauptung hat eine große Wirkung: Fügt man die »fehlende« Menge $\bar{A} \times A$ hinzu, so gilt der Satz nicht mehr – und dies auf recht spektakuläre Weise: Es gibt von Computern nicht entscheidbare formale Sprachen, für die reguläre Obermengen von $A \times A$, $A \times \bar{A}$, $\bar{A} \times A$ und $\bar{A} \times \bar{A}$ existieren, deren Schnitt leer ist.

Protokollkontrolle. Man stelle sich einen endlichen Automaten vor, der n Datenleitungen kontrolliert. Er soll überprüfen, ob ein bestimmtes einfaches Protokoll eingehalten wird, und am Ende einer Datenübertragung mitteilen, auf welchen Datenleitungen dies nicht der Fall war. Ist das Protoll sehr einfach, wie »kein Symbol darf mehr als viermal in einer Reihe vorkommen«, so kann ein endlicher Automat diese Kontrolle durchführen. Bei komplizierteren Protokollen wie »die Anzahl der ›Open‹ Symbole muss gleich der Anzahl der ›Close‹ Symbol sein« reicht ein endlicher Automat nicht aus. Dies ist schlecht, da die hohe Geschwindigkeit auf Datenleitungen es nötig macht, einfache Onlinemaschinen zu benutzen – endliche Automaten eben.

Eine Möglichkeit, den Endliche-Automaten-Ansatz zu retten, ist, dem Automaten Spezialhardware in Form eines Zählers oder eines Stacks zur Seite zu stellen. Diese Hardware kontrolliert ebenfalls die Datenströme und teilt dem Automaten k Bit an Information mit. Die Theorie der Aufzählbarkeit für endliche Automaten zeigt nun, dass k mindestens $\lfloor \log_2 n \rfloor + 1$ sein muss; sonst könnte der Automat auf die Hilfe auch gleich verzichten. Insbesondere kann es für *kein* nichtreguläres Protokoll ausreichend sein, lediglich eine Leitungsnummer an den Automaten zu übermitteln. Diese untere Schranke ist optimal.

Eine zweite Idee, den Ansatz zu retten, ist, nicht zu verlangen, dass alle fehlerhaften Datenleitungen erkannt werden, sondern nur ein Totalausfall. Der Automat soll also »Ok« ausgeben, wenn auf allen Datenleitungen das Protokoll eingehalten wurde, und soll »Fehler« ausgeben, wenn auf keiner das Protokoll eingehalten wurde und alle Datenströme unterschiedlich waren. In allen anderen Fällen (identische Datenströme oder nur einige Fehler) ist die Ausgabe des Automaten beliebig. Die Theorie der Aufzählbarkeit für endliche Automaten zeigt hier, dass dieser Ansatz nichts bringt: Das Erkennen von Totalausfällen mittels endlicher Automaten ist nicht einfacher als das exakte Erkennen der fehlerhaften Leitungen. Wieder verhalten sich allgemeine Turingmaschinen gleich, Polynomialzeitmaschinen aber anders: Es gibt Protokolle, für die Polynomialzeitmaschinen Totalausfälle erkennen können, die fehlerhaften Datenströme einzeln aber nicht.

Klassifikation mit Beispielen. Die letzte Anwendung betrifft die Klassifikation von Objekten. Als Eingabe dienen Objekte, die eine Maschine nach einer bestimmten Eigenschaft wie ihrer Farbe klassifizieren soll. Um diese Aufgabe zu vereinfachen, stellen wir neben dem zu klassifizierenden Objekt noch n Beispiele zur Verfügung, die gleichartig zu klassifizieren sind. Neben einer schwarzen Kugel als Eingabe könnten wir auch noch einen schwarzen Hut und eine schwarze Katze als Beispiele geben.

Die Theorie der Aufzählbarkeit liefert auch hier interessante Ergebnisse: Für Turingmaschinen helfen Beispiele bei der Klassifikation nicht. Jedes Klassifikationsproblem, das sich mit Beispielen lösen lässt, lässt sich auch ohne Beispiele lösen. Für endliche Automaten gilt dasselbe, zumindest für binäre Klassifikation (»Ja« / »Nein« oder »schwarz« / »weiß«). Bei Polynomialzeitmaschinen helfen Beispiele hingegen: Es gibt beliebig komplexe Klassifikationsprobleme, die sich effizient lösen lassen, wenn nur eine bestimmte genügend große Anzahl von Beispielen gegeben wird.

Werdegang

1975	Geboren in Berlin.
1984–1991	Familie lebt in Windhuk, Namibia.
1992, 1993	Bundessieger Informatik, Silbermedaille bei der International Olympiad of Informatics; Stipendium der Studienstiftung des Deutschen Volkes.
1994	Abitur in Berlin; Dr.-Habenna-Preis des Bezirks Wilmersdorf.
1994	Beginn des Studiums der Informatik und ab 1996 der Mathematik an der TU Berlin.
1995	Wehrdienst, Studiumsunterbrechung.
1996–1999	Studentischer Systementwickler bei Lufthansa Systems Berlin.
1999	Diplom in Informatik; Dr.-Erwin-Stephan-Preis der TU Berlin.
1999–2003	Wissenschaftlicher Mitarbeiter an der TU Berlin bei Prof. Dirk Siefkes.
2001	Diplom in Mathematik.
2001	Forschungsaufenthalt in Rochester, USA, bei Prof. Lane Hemaspaandra.
2003	Promotion an der TU Berlin.
2003–2004	Post-Doc-Stipendium des DAAD. Forschungsaufenthalt am International Computer Science Institute in Berkeley, USA, bei Prof. Richard Karp.

Literatur

[Be87] Beigel, R.: *Query-Limited Reducibilities*. PhD thesis. Stanford University. USA. 1987.

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

[Ga91] Gasarch, W. I.: Bounded queries in recursion theory: A survey. In: *Proc. 6th Structure in Complexity Theory Conf.* S. 62–78. IEEE Computer Society Press. 1991.

[HJ95] Hemaspaandra, L. A. und Jiang, Z.: P-selectivity: Intersections and indices. *Theoretical Computer Science*. 145(1–2):371–380. 1995.

[Ku92] Kummer, M.: A proof of Beigel's cardinality conjecture. *Journal of Symbolic Logic*. 57(2):677–681. 1992.

[Ta03] Tantau, T.: *On Structural Similarities of Finite Automata and Turing Machine Enumerability Classes*. Wissenschaft und Technik Verlag. 2003. Auch: Dissertation, Technische Universität Berlin, Februar 2003.