View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Acronym

Analytische Maschinen und Berechenbarkeit analytischer Funktionen

von

Tobias Gärtner

Dissertation zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken 2008

Tag des Kolloquiums:	16.4.2009
Dekan:	Prof. Dr. Joachim Weickert
Berichterstatter:	Prof. Dr. Günter Hotz Prof. Dr. Frank-Olaf Schreyer Prof. Dr. Kurt Mehlhorn Prof. Dr. Friedhelm Meyer auf der Heide
Prüfungsausschuß:	Prof. Dr. Joachim Weickert (Vorsitzender) Prof. Dr. Günter Hotz Prof. Dr. Frank-Olaf Schreyer Prof. Dr. Kurt Mehlhorn Dr. Jan Schwinghammer

Analytische Maschinen und Berechenbarkeit analytischer Funktionen

von

Tobias Gärtner
(Dipl. Inform.)

Dissertation zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken 2008

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, daß ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Zusammenfassung

Gegenstand dieser Arbeit ist der Berechenbarkeitsbegriff über den reellen und komplexen Zahlen und die Charakterisierung der Berechenbarkeit analytischer Funktionen.
Dazu werden *analytische Maschinen* betrachtet, ein von HOTZ eingeführtes Maschinenmodell, das die von BLUM, SHUB und SMALE definierten Maschinen um unendliche konvergente Berechnungen (analytische Berechnungen) erweitert. Es werden Resultate über die Eigenschaften analytisch berechenbarer Funktionen präsentiert und Verallgemeinerungen des Darstellungssatzes von BLUM, SHUB und SMALE für $\mathbb{R}$-berechenbare Funktionen gegeben.
Das Maschinenmodell wird dann dazu benutzt, um Berechenbarkeit holomorpher (komplex-analytischer) Funktionen zu charakterisieren. Für die in der Arbeit definierte Klasse der koeffizientenberechenbaren analytischen Funktionen wird gezeigt, daß sie unter grundlegenden Operationen wie Komposition und lokaler Umkehr abgeschlossen ist. Es wird ferner gezeigt, daß die analytische Fortsetzung einer auf einem Gebiet $D \subseteq \mathbb{C}$ analytischen und berechenbaren Funktion auf ein Gebiet $G \supseteq D$ ebenfalls wieder berechenbar ist.
Zusätzlich zur Berechenbarkeit durch Maschinen wird auch Berechenbarkeit mittels rekursiver Funktionen betrachtet. Die linear primitiv-rekursiven Funktionen werden eingeführt und im Rahmen der μ-rekursiven Funktionen klassifiziert.

Abstract

The subject of this thesis is computability over the real and complex numbers, and the characterization of computable analytic functions. To this end, we consider *analytic machines*, a machine model introduced by HOTZ that extends the machines defined by BLUM, SHUB and SMALE with infinite convergent computations (analytic computations). Results concerning the properties of analytically computable functions are presented and generalizations of BLUM, SHUB and SMALE's representation theorem for $\mathbb{R}$-computable functions are given.
Then, the machine model is used for the characterization of computability of holomorphic (complex-analytic) functions. The class of coefficient-computable analytic functions is introduced and shown to be closed under basic operations such as composition and local inversion. Further, it is shown that, given a function that is analytic and computable on a region $D \subseteq \mathbb{C}$ and which possesses an analytic continuation on a region $G \supseteq D$, this analytic continuation is also computable.
In addition to computability by machines, computability by recursive functions is considered. The linear primitive-recursive function are introduced and classified within the μ-recursive functions.

Danksagung

An dieser Stelle möchte ich allen danken, die mich bei meiner Arbeit unterstützt haben. Prof. Dr. Günter Hotz danke ich für die Themenstellung und für die vielen Anregungen, die ich von ihm in unzähligen Diskussionen erhalten habe. Ich bedanke mich für seine hervorragende Unterstützung und Förderung, und für die große Begeisterungsfähigkeit, die er für jedes ihm vorgebrachte Thema gezeigt hat.
Bei meinem Zweitgutachter Prof. Dr. Frank-Olaf Schreyer bedanke ich mich für sein Interesse an meiner Arbeit und die Unterstützung, die ich von ihm durch meine Gespräche mit ihm erhalten habe.
Timo von Oertzen und Jan Schwinghammer danke ich für die große Hilfe, die sie mir während der Entstehung der Arbeit geleistet haben. Sie haben durch ihre stete Gesprächs- und Diskussionsbereitschaft fachlich und emotional wesentlich zum Gelingen der Arbeit beigetragen, und diese auch mit eigenen Ideen und Anregungen bereichert. Ich danke beiden auch für die gründliche und kritische Durchsicht der Arbeit. Axel Müller möchte ich danken für die große Unterstützung, die er mir bei der Gestaltung der Arbeit und ihrer Korrektur gegeben hat. Ohne ihn wäre das Erscheinungsbild der Arbeit nicht so schön, wie es jetzt ist, insbesondere die Abbildungen betreffend. Ich danke ihm auch für seine moralische Unterstützung.
Schließlich danke ich meiner Mutter, die mich in meinem gesamten bisherigen Leben in jeder Situation rückhaltlos unterstützt hat.

Inhaltsverzeichnis

Zur Notation

Die in dieser Arbeit verwendeten Notationen sind weitgehend Standard in Mathematik und Informatik.
Für Mengen A und B verwenden wir $A \subseteq B$ für die Teilmengenbeziehung. Wollen wir ausdrücken, daß die Teilmengenbeziehung echt ist, so schreiben wir $A \subsetneq B$. Partielle Funktionen von einer Menge A in eine Menge B werden durch $A \rightharpoonup B$ symbolisiert. Für eine Menge M sei $M^* = \bigcup_{n=0}^{\infty} M^n$ die Menge aller endlichen Folgen über M. Wenn wir von Zeichenketten sprechen, nennen wir eine Menge A ein *Alphabet* und A^* ist die Menge aller endlichen Wörter über A. Die beiden Definitionen sind äquivalent und unterscheiden sich nur in der Sprech- und Schreibweise. Für $a \in A^*$ sei $|a|$ die Länge des Wortes a. Die unendlichen Folgen über eine Menge M bezeichnen wir mit $M^{\mathbb{N}}$, dies ist ein Spezialfall der Bezeichnung A^B für die Menge aller Abbildungen von B nach A.
Die Mengen der natürlichen, ganzen, rationalen, reellen und komplexen Zahlen bezeichnen wir mit $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ und $\mathbb{C}$. Für $x \leq y$ bezeichne $[x, y]$ das abgeschlossene Intervall von x bis y, halboffene und offene Intervalle werden durch runde Klammern wie bei $(x, y]$ bezeichnet. Für eine Teilmenge D der komplexen oder reellen Zahlen bezeichne $\| \cdot \|_D$ die Supremumsnorm auf D.
Unter der r-Umgebung eines Punktes $z_0 \in \mathbb{C}$ verstehen wir die offene Menge $U_r(z_0) = \{z \in \mathbb{C} \,|\, |z - z_0| < r\}$, die abgeschlossene r-Umgebung bezeichnen wir mit
$\overline{U}_r(z_0) = \{z \in \mathbb{C} \,|\, |z - z_0| \leq r\}$. Generell bezeichne $\overline{D}$ den Abschluß der Menge D (wobei die Obermenge, in der der Abschluß gebildet wird, aus dem Zusammenhang hervorgeht).

Kapitel 1
Einleitung

Motivation

Die klassische Theorie der Berechenbarkeit, oder Rekursionstheorie, die ihre Ursprünge in Arbeiten von Turing, Hilbert, Gödel, Church, Kleene und vielen anderen hat, dient dazu, den intuitiven Begriff des Algorithmus und der Berechnung mathematisch präzise zu fassen. Zu diesem Zwecke wurden verschiedene Methoden entwickelt, mit denen sogenannte berechenbare Funktionen charakterisiert werden können, etwa die Turing-Maschinen, der λ-Kalkül, verschiedene Termersetzungssysteme oder die μ-rekursiven Funktionen. Während mit Hilfe von Turing-Maschinen auf eine operationale Weise der Begriff der Berechnung definiert wird, geschieht dies beim λ-Kalkül und auch bei den μ-rekursiven Funktionen auf eine funktionale Weise, das heißt hier werden Funktionen mit Hilfe von Gleichungen festgelegt. All diese Modelle wurden in den Jahrzehnten nach ihrer Entdeckung in den 30er Jahren tiefgehend untersucht, und schnell zeigte sich, daß alle zum gleichen Begriff der Berechenbarkeit bei Funktionen führen. Die Tatsache, daß alle Ansätze zum gleichen Ergebnis führen, gab Anlaß zur Formulierung der *Churchschen These*. Diese besagt, daß mit den Turing-berechenbaren Funktionen wirklich auch die "intuitiv berechenbaren" Funktionen erfaßt werden.
Durch die klassische Berechenbarkeitstheorie wurden die Grundlagen für die gesamte theoretische Informatik gelegt. Ein Großteil ihrer Erkenntnisse baut auf ihr auf und wird in ihrer Sprache formuliert.
Obwohl die klassische Theorie der Berechenbarkeit sehr umfassend ist, hat sie einen Nachteil: Die betrachteten Strukturen sind diskret, es werden nur Funktionen der natürlichen Zahlen in sich betrachtet. Man kann die Theorie mit Hilfe von Kodierungen leicht so verallgemeinern, daß man auch rationale berechenbare Funktionen definieren kann. Aber dennoch bleibt die Natur der Berechnung diskret, und reelle oder komplexe Funktionen können darin nicht charakterisiert werden. Die Frage nach berechenbaren kontinuierlichen Funktionen erscheint aber sinnvoll, sind doch zahlreiche Probleme aus den Naturwissenschaften und auch der Informatik, die algorithmischer Lösung bedürfen, in kontinuierlichen Strukturen wie den reellen Zahlen verwurzelt. In der algorithmischen Geometrie etwa wird mit Punktmengen reeller Zahlen gearbeitet, die Bildverarbeitung macht intensiv von analytischen Methoden wie etwa der Fourier-Transformation Gebrauch, und überall in den Naturwissenschaften, wo etwas berechnet werden muß, werden Algorithmen aus der numerischen Mathematik verwendet, die in der Regel reelle Zahlen verarbeiten.
Betrachtet man die Mathematik der letzten 2000 Jahre, so fällt auf, daß schon lange

vor der Entwicklung der Theorie der Berechenbarkeit und der Algorithmen nicht nur die Existenz von Objekten, sondern auch deren Konstruktion oder algorithmische Berechnung von Interesse waren. Polynome etwa kann man auch als konkrete Berechnungsvorschrift auffassen. Die Verallgemeinerung von Polynomen, die analytischen Funktionen, sind als Potenzreihen auch gleichzeitig eine Vorschrift zu ihrer Berechnung, wobei hier die Berechnung unendlich lang ist. Betrachtet man klassische Operationen auf analytischen Funktionen, wie etwa Integration, Differentiation, analytische Fortsetzung, lokale Umkehr, so kann man diese Operationen auch in Verbindung mit der Berechenbarkeit setzen. Welcher Zusammenhang besteht zwischen diesen Operationen und einem etwaigen Berechenbarkeitsbegriff für analytische Funktionen? Werden durch diese Operationen nicht auch berechenbare Funktionen definiert? Wir werden uns in dieser Arbeit mit einigen dieser Fragen beschäftigen.

Beispiele

Wir wollen nun einige Beispiele einfacher Funktionen geben, die ihrer kontinuierlichen Natur nach nicht im Rahmen der klassischen Berechenbarkeitstheorie betrachtet werden können, von denen wir in der Regel aber dennoch eine intuitive Vorstellung haben, ob sie berechenbar sind oder nicht. Es soll aber anhand der Beispiele auch deutlich werden, daß der Berechenbarkeitsbegriff für reelle Funktionen auch in einem intuitiven Sinne nicht so eindeutig ist wie bei den natürlichen Zahlen. Es wird sich zeigen, daß der Berechenbarkeitsbegriff stark vom verwendeten Modell abhängt und die verschiedenen Methoden, Funktionen zu definieren, nicht wie im diskreten Fall die gleichen Mengen berechenbarer Funktionen hervorbringen.

1. *Konstante Funktionen* $f_c : \mathbb{R} \to \mathbb{R},\ x \mapsto c$ mit $c \in \mathbb{R}$. Ist die Konstante c eine natürliche oder rationale Zahl, so ist f_c sicher eine im intuitiven Sinne berechenbare Funktion, denn dann ist sie bereits Turing-berechenbar. Aber auch Funktionen wie $f_\pi : x \mapsto \pi$ kann man als berechenbar ansehen, da π mit beliebiger Genauigkeit approximiert werden kann. Problematisch wird es, wenn die Konstante c als "nicht berechenbar" angesehen wird. Man sieht, daß man unter Umständen auch für Zahlen einen Berechenbarkeitsbegriff geben muß.

2. *Polynomfunktionen* $p(x) = a_n x^n + \ldots + a_0$. Ähnlich wie bei den konstanten Funktionen wird man Polynomfunktionen als im intuitiven Sinne berechenbar ansehen, wenn man die Berechenbarkeit der Koeffizienten außer Acht läßt.

3. *Analytische Funktionen* $a(z) = \sum_{k=0}^{\infty} a_k z^k$. Im Unterschied zu den bisherigen Beispielen kann man diese Funktionen im allgemeinen nicht mehr in endlich vielen Schritten berechnen. Während man die vorigen Funktionen mit Maschinen beschreiben kann, die endlich viele arithmetische Operationen auf reellen Zahlen durchführen können, ist das hier im allgemeinen nicht mehr möglich. Dennoch ist es unsere Auffassung, daß Funktionen wie etwa die Exponentialfunktion oder die trigonometrischen Funktionen in einem intuitiven Sinne berechenbar sind.

4. *Heaviside-Funktion*

$$h : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 0 & : \quad x \leq 0 \\ 1 & : \quad x > 0 \end{cases}$$

Der Unterschied zu den bisher definierten Funktionen besteht darin, daß hier eine *unstetige* Funktion gegeben ist. Es besteht keine Einigkeit darüber, ob diese Funktion intuitiv berechenbar ist oder nicht. Es gibt den Standpunkt, daß nur stetige Funktionen (nicht notwendigerweise in der Standardtopologie stetig) berechenbar sind. Wir sind hingegen der Auffassung, daß die Heaviside-Funktion und ähnliche Funktionen in einem intuitiven Sinne berechenbar sind.

5. "*Lange Bartfunktion*"

$$\chi_{\mathbb{Q}} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 0 & : \quad x \in \mathbb{R} - \mathbb{Q} \\ 1 & : \quad x \in \mathbb{Q} \end{cases}$$

Hier werden die meisten darin übereinstimmen, daß diese Funktion nicht in einem intuitiven Sinne berechenbar ist. Wodurch ist sie aber durch die Heaviside-Funktion h abgegrenzt? Es gibt Maschinenmodelle, in denen $\chi_{\mathbb{Q}}$ berechenbar ist und solche, in denen sie es nicht ist.

Diese Beispiele zeigen schon, daß nicht unbedingt Einigkeit über den intuitiven Berechenbarkeitsbegriff herrschen muß.
Wie in der klassischen Theorie sind nicht nur berechenbare Funktionen, sondern auch entscheidbare Mengen interessant. Roger Penrose [Pen91] stellt etwa die Frage nach der Entscheidbarkeit der Mandelbrotmenge und diskutiert auch die Unzulänglichkeit des Turing-Maschinenmodells, formal über diese Entscheidbarkeitsfrage zu sprechen.

Einordung

Blum, Shub und Smale [BSS89] haben die Theorie der Berechenbarkeit ausgehend vom klassischen Modell ausgedehnt und ein Maschinenmodell angegeben, das reelle Zahlen als atomare Objekte betrachtet und mit diesen arithmetische Operationen durchführen kann. Eine Berechnung ist in in diesem Modell eine endliche Abfolge von arithmetischen Operationen und bedingten Verzweigungen. Dieses Modell ist der Erweiterung eines klassischen Registermaschinenmodells auf die reellen Zahlen sehr ähnlich. Formal definieren Blum, Shub und Smale sogenannte Graphenmaschinen, die durch Graphen gegeben sind, deren Knoten für einzelne Programmbefehle stehen. Sie entwickeln eine Komplexitätstheorie für dieses Maschinenmodell und formulieren das $P \neq NP$-Problem über $\mathbb{R}$.
Eine wesentliche Einschränkung des Maschinenmodells im Hinblick auf eine Berechenbarkeitstheorie über den reellen und komplexen Zahlen ist die Beschränkung auf endliche Berechnungen. Die reellen Zahlen sind fundamental mit unendlichen Grenzprozessen verknüpft, formal werden sie mit Hilfe solcher Grenzprozesse definiert. Von den diskutierten Beispielen sind die konstanten Funktionen und die Polynomfunktionen berechenbar, wenn man die Koeffizienten der Polynome bzw. Konstanten voraussetzt. Ebenso sind abschnittsweise definierte Funktionen wie die Heaviside-Funktion berechenbar, wenn die Anzahl der Abschnitte endlich ist oder diese Abschnitte aufzählbar sind. Im wesentlichen sind dies auch alle in diesem Modell berechenbaren Funktionen. Analytische Funktionen wie etwa die Exponentialfunktion oder auch nichtrationale algebraische Funktionen wie z.B. die Quadratwurzel sind nicht berechenbar, da diese Funktionen nicht mittels einer endlichen Abfolge arithmetischer Operationen und Vergleichsoperationen berechnet

werden können.
In der Typ-2 Theorie der Berechenbarkeit oder der rekursiven Analysis, wie sie etwa von Weihrauch [Wei00] vertreten wird, wird die Erweiterung des Berechenbarkeitsbegriffs von der anderen Seite aus angegangen: Das Modell der Turing-Maschine wird beibehalten, die Bänder können nach wie vor nur Nullen und Einsen beziehungsweise Elemente eines endlichen Alphabets tragen, aber die Maschinen können unendlich lange rechnen. Das Ergebnis einer Berechnung ist dann das Ergebnis eines Grenzprozesses. In der gewöhnlichen Ausführung dieses Modells sind die berechenbaren Funktionen automatisch stetig. Mittels der Typ-2 Turing-Maschinen sind durch die Möglichkeit der unendlich langen Berechnung auch analytische Funktionen berechenbar, soweit die Koeffizientenfolgen der Potenzreihenentwicklung berechenbare Zahlenfolgen sind. Unstetige Funktionen wie die Heaviside-Funktion sind im allgemeinen in diesem Modell nicht berechenbar.
Die prinzipielle Idee, mittels unendlich lange rechnenden Turing-Maschinen auch reelle Funktionen zu berechnen, existiert schon längere Zeit. In [Ko91] wird sie mittels Orakel-Turing-Maschinen entwickelt. Schon bei Grzegorczyk [Grz57] finden sich Ansätze einer Berechenbarkeitstheorie über den reellen Zahlen mittels Turing-Maschinen. Einen Überblick über die Typ-2 Theorie der Berechenbarkeit, der auch weit darüber hinausgeht und unter anderem die Maschinen von Blum, Shub und Smale (BSS-Maschinen) behandelt, findet man in [Zie07]. Brattka und Hertling entwickelten mit dem Modell der "feasible real RAM" ein Registermodell über den reellen Zahlen, das mit der Typ-2 Turing-Berechenbarkeit kombiniert wird.
Mit Hilfe von Repräsentationsfunktionen, die Ein- und Ausgabebänder interpretieren, wird die Expressivität der Typ-2 Theorie vergrößert und es ist möglich, nicht nur Funktionen über den reellen Zahlen zu betrachten, sondern beispielsweise auch Funktionen allgemeinerer metrischer Räume. Die grundsätzliche Charakteristik, daß in diesem Maschinenmodell mit einem endlichen Alphabet gerechnet wird, wird aber beibehalten.
Wir sind der Ansicht, daß auch unstetige Funktionen wie etwa die Heaviside-Funktion in einem intuitiven Sinne berechenbar sind, und daß das Maschinenmodell dies widerspiegeln sollte. Die Beschränkung auf Nullen und Einsen erscheint bei der Untersuchung der Berechenbarkeit reeller Funktionen vielleicht auch sehr restriktiv.
Nach einer Idee von Hotz [Hot94] haben Vierke [Vie96], Hotz, Vierke und Schieffer [HVS95], Chadzelek [Cha98] und Chadzelek und Hotz [CH99] den Grundstein für die Theorie der *Analytischen Maschinen* gelegt. Die Analytischen Maschinen sind ein Registermaschinenmodell, das bei endlich vielen Rechenschritten dem von Blum, Shub und Smale in seiner Berechnungsmächtigkeit äquivalent ist. Im Unterschied zu den BSS-Maschinen werden bei Analytischen Maschinen auch unendliche, konvergente Berechnungen zugelassen. Dieser Ansatz hat den Vorteil, daß er einerseits Grenzwertbildungen berücksichtigt, und andererseits die reellen oder komplexen Zahlen als Atome betrachtet, von bitweisen Operationen wie bei Turing-Maschinen also abstrahiert.
Die Analytischen Maschinen bilden auch das grundlegende Maschinenmodell dieser Arbeit.

Übersicht

In *Kapitel 2* führen wir das Modell der Analytischen Maschinen ein. Auf ähnliche Weise wie in [Vie96] und [Cha98] geben wir die Definition der Analytischen Maschinen und unendlicher Berechnungen. Die Definition des Maschinenmodells erfolgt zunächst allgemein für Ringe $\mathcal{R}$, aber abgesehen von diesen grundlegenden Definitionen werden Maschinen über den reellen Zahlen $\mathbb{R}$ und den komplexen Zahlen $\mathbb{C}$ betrachtet. Diese Maschinen verfügen über exakte Arithmetik über diesen Zahlkörpern und rechnen mit reellen bzw. komplexen Zahlen als Atomen. Jede Maschine korrespondiert zu einer durch sie berechneten Funktion; auf diese Weise werden berechenbare Funktionen über die Berechenbarkeit durch eine Maschine definiert.
Bei reellen Rechnungen stellt sich die Frage nach rationaler Approximierbarkeit. Die δ-$\mathbb{Q}$-Maschinen sind Registermaschinen über den rationalen Zahlen $\mathbb{Q}$, die gerundete reelle Eingaben verarbeiten und deren Ausgaben gegen reelle Zahlen konvergieren.
Mit dem Berechenbarkeitsbegriff von Funktionen ist auch der Begriff der Entscheidbarkeit von Mengen gegeben. Analog zum Halteproblem für Turing-Maschinen ist das Konvergenzproblem für analytische Maschinen. Wir verschärfen die Resultate aus [Cha98] diesbezüglich und lösen ein dort gestelltes offenes Problem. Darüber hinaus untersuchen wir das Beschränktheitsproblem für analytische Maschinen und zeigen, daß die Menge der Häufungswerte einer berechenbaren Folge nicht berechenbar ist.
In der klassischen Theorie ist das Konzept des Nichtdeterminismus vor allem bei komplexitätstheoretischen Betrachtungen wichtig. Während die Berechnungsmächtigkeit nichtdeterministischer Maschinen gleich der deterministischer Maschinen ist, gilt dies nicht für die Komplexität. Wir führen zwei verschiedene Formen des Nichtdeterminismus für analytische Maschinen ein, zum einen Nichtdeterminismus mittels nichtdeterministischer Verzweigungen und zum anderen das Raten reeller Zahlen. Wir zeigen, daß Nichtdeterminismus bei unendlichen Berechnungen zu einer größeren Berechnungsmächtigkeit führt als der Determinismus.
Bei Blum, Shub und Smale wird die Maschine unmittelbar über einen Graphen definiert, in dem sich die Berechnungen entlang von Pfaden in diesem Graphen nachvollziehen lassen. Für Analytische Maschinen definieren wir Berechnungsgraph und Berechnungsbaum, und führen das Konzept der Berechnungspfade ein, das bei den BSS-Maschinen wie bei den Analytischen Maschinen zum Darstellungssatz für $\mathbb{R}$-Maschinen führt. Dieser charakterisiert die $\mathbb{R}$-berechenbaren Funktionen dadurch, daß ihr Definitionsbereich in semi-algebraische Mengen zerfällt, auf denen sie Polynome darstellen. Dies ergibt sich daraus, daß die Maschinen arithmetische Operationen und Vergleichsoperationen endlich oft hintereinander ausführen können. Bei unendlich langen Berechnungen liegt die Verallgemeinerung nahe, daß hier nicht Polynome, sondern Potenzreihen von den Maschinen berechnet werden. Wir untersuchen diese Verallgemeinerungen des Darstellungssatzes und zeigen, daß der Darstellungssatz über den reellen Zahlen nicht auf unendliche Berechnungen verallgemeinert werden kann. Für Maschinen über den komplexen Zahlen zeigen wir, daß die von den Maschinen berechneten Funktionen immer auf einer offenen Teilmenge des Definitionsbereiches durch eine Potenzreihe dargestellt werden können, wenn auf jeder Berechnung der Maschine nur endlich viele Verzweigungsbefehle auftreten.

In *Kapitel 3* werden die in Kapitel 2 entwickelten Modelle verwendet, um die Berechenbarkeit komplex-analytischer Funktionen zu charakterisieren. Die analytische Funktionentheorie ist eine wesentliche Säule der Mathematik. Viele der Operationen, die in dieser Theorie betrachtet werden, sind eng mit der Berechenbarkeit verknüpft. Eine Potenzreihe kann bei gegebenen Koeffizienten unmittelbar als Berechnungsvorschrift aufgefaßt werden, aber mit Hilfe der klassischen Berechenbarkeitstheorie kann die Berechenbarkeit analytischer Funktionen nicht adäquat charakterisiert werden. Es wäre zwar möglich, eine Funktion als berechenbar anzusehen, wenn die Koeffizientenfolge ihrer Potenzreihenentwicklung eine rekursiv aufzählbare Folge bildet. Dadurch entsteht aber die Einschränkung, daß nur rationale Koeffizientenfolgen berücksichtigt werden können. Auch sind viele Operationen in dieser Sichtweise nicht berechenbar, die wir als "intuitiv berechenbar" ansehen, und die vom Standpunkt der klassischen Funktionentheorie von zentraler Bedeutung sind.
Diese Probleme werden gelöst, wenn man nicht die Turing-Maschinen, sondern analytische Maschinen als grundlegendes Berechenbarkeitsmodell für analytische Funktionen wählt. In diesem Fall gibt es zwei Möglichkeiten, Berechenbarkeit analytischer Funktionen zu charakterisieren. Die erste Möglichkeit besteht darin, die Berechenbarkeit der Funktion vorauszusetzen. Alternativ fordert man die Berechenbarkeit der Koeffizientenfolge der Potenzreihe mittels einer analytischen Maschine. Wir zeigen, daß jede Funktion, deren Potenzreihenentwicklung in einem Punkt berechenbar ist, bereits auf einer Umgebung des Punktes als Funktion berechenbar ist. Unter der Voraussetzung, daß die Funktion für jedes Argument mit endlich vielen Verzweigungen berechenbar ist, gilt auch die Umkehrung. Schließlich untersuchen wir klassische analytische Operationen wie lokale Umkehr und analytische Fortsetzung auf Berechenbarkeit. Dabei stellt sich heraus, daß eine Funktion, die auf einer Teilmenge der komplexen Ebene definiert ist und in einem kleinen Bereich dieser Menge berechenbar ist, auch schon auf ihrem gesamten Definitionsbereich berechnet werden kann.

Im *vierten Kapitel* schließlich betrachten wir nicht mehr die Berechenbarkeit durch Maschinen, sondern gehen auf Berechenbarkeit durch rekursive Funktionen ein. In [Gä01] wurden die linearen rekursiven Funktionen einer Variablen als Funktionen eingeführt, die einer linearen Rekursionsgleichung genügen. Diese Funktionen wurden zu analytischen linearen rekursiven fortgesetzt. Das heißt, es wurden analytische Funktionen definiert, die über den komplexen Zahlen einer linearen Rekursionsgleichung genügen. Ein einfaches Beispiel für eine solche Funktion ist die Γ-Funktion, die die Fakultät zu einer auf $\mathbb{C}$ meromorphen Funktion fortsetzt. Wir zeigen nun, daß über den natürlichen Zahlen die linear primitiv-rekursiven Funktionen eine Klasse von Funktionen bilden, die unter anderem alle polynomialzeitberechenbaren und exponentialzeitberechenbaren Funktionen enthält. Wir zeigen, daß die Funktionen dieser Klasse einer Wachstumsbeschränkung unterliegen und grenzen die Klasse von der Klasse der allgemeinen primitiv-rekursiven Funktionen ab, indem wir zeigen, daß die "Turmfunktion" $2^{2^{\cdot^{\cdot^{2}}}}$ nicht der Wachstumsbeschränkung unterliegt und somit nicht linear primitiv-rekursiv ist. Zum Abschluß der Arbeit skizzieren wir eine Methode, mit der auch gewisse nichtlineare rekursive Funktionen zu analytischen Funktionen fortgesetzt werden können.

Kapitel 2
Analytische Maschinen

2.1 Definition der Analytischen Maschinen

Das Maschinenmodell, welches der vorliegenden Arbeit zugrunde liegt, ist ein Registermaschinenmodell. Zunächst geben wir aber die abstrakte Definition einer mathematischen Maschine, in deren Rahmen Berechnungen und die von einer Maschine berechneten Funktionen definiert werden. Dieses Modell wird dann konkretisiert durch die Registermaschinen. Um möglichst allgemein zu bleiben, definieren wir diese $\mathcal{R}$-Maschinen für einen Ring $\mathcal{R}$, für den wir natürlich speziell die Ringe $\mathbb{R}$ der reellen und $\mathbb{C}$ der komplexen Zahlen im Auge haben. Für die allgemeine Definition fordern wir aber nur, daß $\mathcal{R}$ ein beliebiger Ring ist.
Die $\mathcal{R}$-Maschinen sind Maschinen, die exakte Arithmetik über $\mathcal{R}$ verwenden, die aber nur endliche Berechnungen zulassen. Sie werden dann zu den im Interesse dieser Arbeit stehenden analytischen Maschinen erweitert, die unendliche, konvergente Berechnungen erlauben. Die $\mathcal{R}$-Maschinen entsprechen den Maschinen über einem Ring $\mathcal{R}$ von Blum, Shub und Smale [BSS89], und das hier definierte spezielle Modell wurde bereits in ähnlicher Form in [Cha98, CH99, Vie96] verwendet. Die Einführung der Definitionen orientiert sich hier an [Cha98].

2.1.1 Abstrakte Maschinen

Wir geben zunächst die Definition einer mathematischen Maschine, die es uns erlaubt, formal über Berechnungen und berechnete Funktionen zu sprechen. Dazu sei A im folgenden ein Alphabet. Dieses Alphabet wird später dem Ring entsprechen, über dem die konkrete Maschine definiert ist, in unserem Falle also $\mathbb{R}$ oder $\mathbb{C}$.
In diesem Abschnitt werden nur deterministische Maschinen definiert. Die Definition der nichtdeterministischen Maschinen erfolgt in einem eigenen Abschnitt.

Definition 2.1.1 (Abstrakte Maschine)**.** *Eine* Abstrakte Maschine *über dem Alphabet A ist ein Tupel*

$$\mathcal{M} = (K, K_s, K_e, K_t, \Delta, A, \text{in}, \text{out})$$

Hierbei ist K die Menge der Konfigurationen *der Maschine, $K_s \subseteq K$, $K_e \subseteq K_t \subseteq K$* Start-, End- und Zielkonfigurationen*, die Funktion $\Delta : K \to K$ mit $\Delta|_{K_e} = \text{id}$ ist die* Übergangsfunktion *und die Funktionen* $\text{in} : A^* \to K_s$ *und* $\text{out} : K \to A^*$ *sind* Ein- *und* Ausgabefunktionen.

Wir führen nun den Begriff der *Berechnung* ein. Dabei unterscheiden wir zwischen haltenden Berechnungen, wie sie von klassischen Maschinen her bekannt sind und unendlichen, konvergenten Berechnungen, die wir im folgenden *analytische* Berechnungen nennen werden. Hier werden die End- und Zielkonfigurationen aus der obigen Definition eingesetzt: Befindet sich eine Maschine in einer Endkonfiguration, so hat sie eine endliche Berechnung ausgeführt und hält. Unendliche Berechnungen werden dadurch gekennzeichnet, daß eine Maschine bei einer solchen Berechnung unendlich oft eine Zielkonfiguration annehmen muß.
Um über konvergente Berechnungen sprechen zu können, muß der Konvergenzbegriff auf A^* zunächst überhaupt Sinn machen. Dazu setzen wir voraus, daß auf den $A^i, i \in \mathbb{N}$ Metriken gegeben sind und betrachten eine Folge $(a_n)_{n\in\mathbb{N}} \subseteq A^*$ als *konvergent*, wenn es ein $i \in \mathbb{N}$ gibt, so daß $a_n \in A^i$ für fast alle n gilt, und daß $\lim_{n\to\infty} a_n$ in A^i existiert.

Definition 2.1.2 (Berechnungen)**.** *Eine* Berechnung angesetzt auf x *auf einer Maschine ist eine Folge* $b = (k_i)_{i\in\mathbb{N}} \in K^{\mathbb{N}}$ *mit* $k_0 = \mathrm{in}(x)$ *und* $k_{i+1} = \Delta(k_i)$*. Sie heißt* regulär*, falls* $k_0 \in K_s$*. Eine* haltende *bzw.* endliche *Berechnung ist eine Berechnung mit* $k_n \in K_e$ *für ein* $n \in \mathbb{N}$*. Die* Länge *der Berechnung ist dann das kleinste solche* n_e *und das* Ergebnis *der Berechnung ist* $\mathrm{out}(k_{n_e})$*.*
Sei $b = (k_i)_{i\in\mathbb{N}} \in K^{\mathbb{N}}$ *eine reguläre Berechnung, bei der für unendlich viele* n *gilt:* $k_n \in K_t$*. Sei* $(k_{n_j})_{j\in\mathbb{N}}$ *die Folge der Zielkonfigurationen. Die Berechnung heißt* analytisch*, wenn die Folge der Ausgaben der Zielkonfigurationen konvergent ist, d.h.* $y = \lim_{j\to\infty} \mathrm{out}(k_{n_j})$ *existiert. In diesem Fall ist* y *das* Ergebnis *der Berechnung. Wir bezeichnen die Maschine dann als* konvergent*.*

Anmerkung. Da wir im folgenden nur reguläre Berechnungen betrachten, lassen den Zusatz regulär weg. Anstelle von analytischen Berechnungen sprechen wir oft auch von konvergenten unendlichen Berechnungen. Indem eine Maschine unendlich oft eine Zielkonfiguration annimmt, zeigt sie an, daß sie eine unendlich lange Berechnung durchführt. So gibt es also zwei verschiedene Möglichkeiten, daß eine unendliche Berechnung als nicht konvergent angesehen wird:

1. Sie nimmt nur endlich oft eine Zielkonfiguration an.
2. Die Ausgabefolge der (unendlich vielen) Zielkonfigurationen ist nicht konvergent.

Anstelle der Ausgabefolge der Zielkonfigurationen kann man auch unmittelbar die Ausgabefolge der Maschine betrachten, solange die Forderung, daß unendlich viele Zielkonfigurationen angenommen werden müssen, bestehen bleibt. Beide Möglichkeiten sind für unser Modell jedoch äquivalent.

Eingaben, auf denen Berechnungen halten bzw. konvergieren, bilden nun *Haltebereich* $H_{\mathcal{M}}$ und *Definitionsbereich* $D_{\mathcal{M}}$ der Maschine:

$$\begin{aligned} H_{\mathcal{M}} &= \{x \in A^* \,|\, \text{Berechnung von } \mathcal{M} \text{ angesetzt auf } x \text{ endlich}\} \\ D_{\mathcal{M}} &= \{x \in A^* \,|\, \text{Berechnung von } \mathcal{M} \text{ angesetzt auf } x \text{ analytisch}\} \end{aligned}$$

Da jede Endkonfiguration auch eine Zielkonfiguration ist, gilt $H_{\mathcal{M}} \subseteq D_{\mathcal{M}}$.
Eine Maschine $\mathcal{M}$ induziert nun auf natürliche Weise eine partielle Funktion

$$\Phi_{\mathcal{M}} : A^* \rightharpoonup A^*$$

Ist für $x \in A^*$ die Berechnung von $\mathcal{M}$ angesetzt auf $\mathrm{in}(x)$ endlich oder analytisch mit Ergebnis $y \in A^*$, so ist $\Phi_{\mathcal{M}}(x) := y$. Der Einfachheit halber führen wir folgende Schreibweise ein: Abkürzend für $\Phi_{\mathcal{M}}(x)$ schreiben wir auch $\mathcal{M}(x)$. Die *n-te Approximation* $\mathcal{M}^{(n)}(x)$ ist dann definiert als $\mathrm{out}(k_{j_n})$, wobei j_n der Index der n-ten Zielkonfiguration ist.

2.1.2 $\mathcal{R}$-Maschinen

Die konkreten $\mathcal{R}$-Maschinen, die wir im folgenden definieren werden, sind Registermaschinen, die mit Elementen aus $\mathcal{R}$ als Atomen rechnen können. Dies hat den Vorteil, daß von Bitoperationen wie etwa bei Turing-Maschinen abstrahiert wird und die arithmetischen Operationen als einzelne Rechenschritte betrachtet werden können. Der Ring $\mathcal{R}$ wird dann im konkreten Falle natürlich der Ring der reellen bzw. komplexen Zahlen sein.
Das Maschinenmodell ist schematisch dargestellt in Abbildung 2.1. Die Maschinen sind

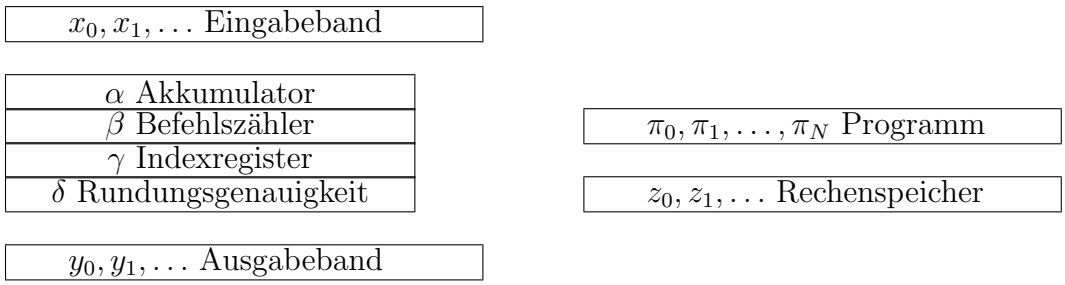


Abbildung 2.1: Das Schema der $\mathcal{R}$-Maschine

aufgebaut aus verschiedenen Registern, Ein- und Ausgabe, Speicher und einem Programm. Die (unendlichen) Ein- und Ausgabebänder $x = (x_0, x_1, \ldots)$, $y = (y_0, y_1, \ldots) \in \mathcal{R}^{\mathbb{N}}$ und der (unendliche) Rechenspeicher $z = (z_0, z_1, \ldots) \in \mathcal{R}^{\mathbb{N}}$ enthalten in jeder Zelle ein Element aus $\mathcal{R}$. Darüber hinaus gehört zu einer $\mathcal{R}$-Maschine ein endliches Programm $\pi = (\pi_1, \ldots, \pi_N)$ mit Befehlen aus dem Befehlssatz Prog. In Tabelle 2.1 ist der Aufbau von Prog angegeben, wobei wir hier darauf verzichten, eine formale Grammatik für Prog anzugeben. Weiterhin ist $\mathsf{Prog} = \mathsf{Prog}_{\mathcal{R}}$ natürlich abhängig vom betrachteten Ring, sofern dieser aus dem Zusammenhang jedoch hervorgeht, wird auf den Zusatz auch meist verzichtet. Die Menge aller Programme über einem Ring ist dann gegeben durch $\mathsf{Prog}^*_{\mathcal{R}}$.
Die zentrale Steuereinheit der Maschine besteht aus einem Akkumulator $\alpha \in \mathcal{R}$, über den die Rechenoperationen laufen, einem Befehlszähler $\beta \in \mathbb{N}$ und einem Indexregister $\gamma \in \mathbb{N}$.
Auf die Bedeutung des Genauigkeitsregisters δ wird bei der Einführung von δ-$\mathbb{Q}$-Maschinen eingegangen.

Anmerkung. Bei endlichen Berechnungen wird natürlich nur ein endlicher Teil des Eingabebandes gelesen und ebenfalls nur ein endlicher Teil des Rechenspeichers verwendet. Tatsächlich kann man leicht zeigen, daß in diesem Fall bei den Ringen $\mathbb{Q}, \mathbb{R}$ oder $\mathbb{C}$ ein Rechenspeicher von konstanter Größe genügt [Vie96]. Die Idee besteht dabei darin, die Information mehrerer Speicherzellen in eine einzige Zahl zu kodieren.

Die Menge der Konfigurationen der Maschine mit Programm $\pi \in \mathsf{Prog}^*$ ist nun durch den Inhalt der Register und der Speicherzellen gegeben:

$$K := \left\{ k = (\alpha, \beta, \gamma, \pi, x, y, z) \,\middle|\, \alpha \in \mathcal{R},\ \beta, \gamma \in \mathbb{N}, x, y, z \in \mathcal{R}^{\mathbb{N}} \right\}$$

Tabelle 2.1 *Die Menge* $\mathsf{Prog}_{\mathcal{R}}$ *der Programmbefehle*

1. Zuweisungen
   - $\alpha := x_i$, $\alpha := z_i$, $y_i := \alpha$, $z_i := \alpha$, $\quad i \in \mathbb{N} \cup \{\gamma\}$
   - $\alpha := c$, $c \in \mathcal{R}$
   - $\gamma := 0$
2. Arithmetik
   - $\alpha := \alpha + z_i$, $\alpha := \alpha \cdot z_i$
   - $\alpha := -\alpha$, $\alpha := \frac{1}{\alpha}$
   - $\gamma := \gamma + 1$, $\gamma := \gamma \dot{-} 1$ Mit ' $\dot{-}$ ' ist die nichtnegative Differenz gemeint.
3. Verzweigungen
   `goto` m
   `if` $\alpha > 0$ `then goto` m `else goto` n ($\mathcal{R}$ angeordnet)
   `if` $\alpha \neq 0$ `then goto` m `else goto` n ($\mathcal{R}$ nicht angeordnet)
   `if` $|\alpha| > |z_i|$ `then goto` m `else goto` n ($\mathcal{R} = \mathbb{C}$)
4. Spezielle Befehle
   `end`
   `print`
   `exception`
   `next` δ

Anfangs-, End- und Zielkonfigurationen sind auf natürliche Weise gegeben durch:

$$\begin{aligned} K_s &:= \{k \in K \mid \alpha = \gamma = 0, \beta = 1, \forall j : y_j, z_j = 0\} \\ K_e &:= \{k \in K \mid \pi_\beta = \texttt{end}\} \\ K_t &:= \{k \in K \mid \pi_\beta = \texttt{print}\} \end{aligned}$$

Die Übergangsfunktion Δ ergibt sich nun in nahe liegender Weise aus dem Befehlssatz Prog. Wir geben sie hier deshalb nicht mehr explizit an. Auf den Befehl `next` δ wird in Abschnitt 2.1.3 über δ-$\mathbb{Q}$-Maschinen eingegangen; `exception` dient dazu, die Maschine zu stoppen, ohne sie in eine Endkonfiguration zu überführen. Dies wird dadurch realisiert, daß die Übergangsfunktion auf Konfigurationen mit diesem Befehl im Befehlsindex identisch operiert. Somit kann eine Berechnung, die auf ein `exception`-Kommando stößt, nicht halten und auch nicht analytisch sein, da keine Zielkonfigurationen mehr angenommen werden können. Berechnungen, für die $\alpha := \frac{1}{\alpha}$ nicht definiert ist, also etwa wenn das Inverse von α nicht definiert ist oder eine Division durch 0 stattfinden soll, werden gestoppt, indem die Übergangsfunktion identisch operiert und so keine End- bzw. Zielkonfiguration erreicht werden kann.
Die zur Verfügung stehende Verzweigungsbedingung hängt vom Ring $\mathcal{R}$ ab, über dem die Maschine definiert ist. Im allgemeinen Fall gibt es nur die Bedingung `if` $\alpha \neq 0$. Sollte der Ring $\mathcal{R}$ jedoch angeordnet sein, so lassen wir zusätzlich die Bedingung `if` $\alpha > 0$ zu. Im Falle des Körpers $\mathbb{C}$ der komplexen Zahlen schließlich steht noch die Bedingung `if` $|\alpha| > |z_i|$ zur Verfügung, die Vergleiche von Beträgen komplexer Zahlen gestattet. Trifft eine Maschine auf einen Sprung in eine Programmadresse, die nicht existiert, so wird dies wie die Division durch 0 behandelt.
Da $\mathcal{R}$-Maschinen Funktionen $\mathcal{R}^* \rightharpoonup \mathcal{R}^*$ berechnen, muß für Ein- und Ausgabe angegeben werden, wie lang diese jeweils sind. Dies geschieht mit Hilfe der Ein- und Ausgabefunktionen. Diese interpretieren das erste Element der Ein- bzw. Ausgabefolge als Längenangabe in folgendem Sinne:
Ist $r = (r_1, r_2, \ldots, r_n) \in \mathcal{R}^*$ gegeben, so ist $\text{in}(r) = (0, 1, 0, \pi, x, y, z)$ mit $x_0 = n$, $x_i = r_i\,(1 \leq i \leq n), x_i = 0\,(i > n)$ und $y_i = z_i = 0\,(i \in \mathbb{N})$. Für eine Konfiguration $k = (\alpha, \beta, \gamma, \pi, x, y, z)$ interpretiert out den ersten Eintrag y_0 als natürliche Zahl, die die Länge der Ausgabefolge angibt, es ist also $\text{out}(k) = (y_1, y_2, \ldots, y_{y_0})$.

Anmerkung. Für die Interpretation des ersten Elements des Ein- und Ausgabebandes als Längenangabe wird benötigt, daß die natürlichen Zahlen ein Untermonoid von $\mathcal{R}$ bilden. Dies kann man dadurch umgehen, daß man $x, y \in \mathbb{N} \times \mathcal{R}^{\mathbb{N}}$ fordert und entsprechende Inkrementierungs- und Dekrementierungsbefehle für diese Längenangaben zum Befehlssatz hinzufügt. Für die Zwecke dieser Arbeit ist das aber belanglos, da wir ausschließlich die Ringe $\mathbb{R}$ und $\mathbb{C}$ betrachten. Wenn man (partielle) Funktionen $\mathcal{R}^i \rightharpoonup \mathcal{R}^k$ für feste i, k betrachtet, kann man auf die Längenangabe auch verzichten. Das werden wir in den entsprechenden Fällen auch tun.

Definition 2.1.3. *Für einen Ring $\mathcal{R}$ und ein Programm $\pi \in \text{Prog}^*$ heißt die durch obige Konstruktion festgelegte abstrakte Maschine $\mathcal{M} = (K, K_s, K_e, K_t, \Delta, \mathcal{R}, \text{in}, \text{out})$ die* $\mathcal{R}$-Maschine mit Programm π.

Maschinen als Eingabe von Maschinen; Universelle Maschinen

Der Ring $\mathcal{R}$ sei in diesem Abschnitt der Körper der reellen Zahlen $\mathbb{R}$ oder der Körper der komplexen Zahlen $\mathbb{C}$. Wir führen die folgende Diskussion exemplarisch für $\mathbb{R}$ durch,

alles Gesagte gilt analog auch für $\mathbb{C}$.
Die Menge aller möglichen Programme einer $\mathbb{R}$-Maschine ist gegeben durch Prog^*. Wir schränken die Möglichkeit, allgemeine Konstantenzuweisungen der Form $\alpha := c$, $c \in \mathbb{R}$ durchzuführen, für $\mathbb{R}$ auf folgende Weise ein: Für eine Menge $C \subseteq \mathbb{R}$ sei $\mathsf{Prog}[C]$ die Menge der Programmbefehle, wobei die Konstantenzuweisungen nur noch aus $C \cup \mathbb{Q}$ erfolgen können, d.h. die Form $\alpha := c$, $c \in C \cup \mathbb{Q}$ haben müssen. Im weiteren Verlauf der Arbeit sind hauptsächlich Maschinen ohne reelle (irrationale) Konstantenzuweisungen Gegenstand; allgemeine Konstantenzuweisungen verzerren durch Kodierungen das Bild der Berechenbarkeit, siehe etwa Satz 2.3.2. Natürliche (und damit auch rationale) Konstantenzuweisungen werden jedoch stets zugelassen, sie könnten auch bei Vorhandensein der 1 als Konstante leicht durch arithmetische Befehle erzeugt werden. Wir sprechen dann von Maschinen ohne Konstanten.
Wir bezeichnen mit

$$\mathbf{M}_{\mathbb{R}}[C] = (\mathsf{Prog}[C])^*$$

die Menge aller $\mathbb{R}$-Maschinen mit Konstanten aus C und mit

$$\mathbf{M}_{\mathbb{R}} = (\mathsf{Prog})^*$$

die Menge aller $\mathbb{R}$-Maschinen ohne Konstanten.
Durch Wahl einer präfixfreien Kodierung $G : \mathbf{M}_{\mathbb{R}} \to \mathbb{N}$ können wir die Menge der Maschinen in die Menge der natürlichen Zahlen einbetten. Auf diese Weise erhalten wir eine Aufzählung aller $\mathbb{R}$-Maschinen. Dieses Vorgehen ist auch aus der klassischen Berechenbarkeitstheorie als *Gödelnumerierung* bekannt. Vermöge dieser Einbettung und der Einbettung von $\mathbb{N}$ in $\mathbb{R}$ können wir nun Maschinen als Eingabe von Maschinen auffassen, indem wir die Maschine mit ihrer Kodierung identifizieren. Wir schreiben $\mathcal{M}(\mathcal{N})$ für $\mathcal{M}(G(\mathcal{N}))$, also Verwendung der Maschine $\mathcal{N}$ als Eingabe für die Maschine $\mathcal{M}$. Damit eine Maschine eine andere Maschine als Eingabe verarbeiten kann, muß freilich die Abbildung G und auch ihre Umkehrung berechenbar sein. Dazu können etwa die bekannten berechenbaren Bijektionen $\mathbb{N}^* \to \mathbb{N}$ verwendet werden.
Durch Simulation sieht man nun, daß eine *universelle Maschine* $\mathcal{U}$ mit Definitionsbereich in $\mathbb{R} \times \mathbb{R}^*$ existiert mit

$$\mathcal{U}(\mathcal{M}, x) = \mathcal{M}(x),\ x \in D_{\mathcal{M}}$$

Wird nun festgelegt, daß die Maschine $\mathcal{U}$ in einen Ausnahmezustand übergeht, wenn sie als erste Eingabe eine Zahl erhält, die keine Kodierung einer Maschine ist, dann sieht man, daß der Definitionsbereich von $\mathcal{U}$ eine Teilmenge von $G(\mathbf{M}_R) \times \mathbb{R}^*$ ist bzw. $\mathbf{M}_{\mathbb{R}} \times \mathbb{R}^*$ nach Identifikation von $G(\mathbf{M}_{\mathbb{R}})$ mit $\mathbf{M}_{\mathbb{R}}$.
Auf ähnliche Weise werden wir im weiteren Verlauf der Arbeit Maschinen durch Identifikation mit ihrer Kodierung als Eingabe von Maschinen auffassen und auch Funktionen von Mengen $\mathbf{M} \subseteq \mathbf{M}_{\mathbb{R}}$ analytischer Maschinen betrachten.

2.1.3 δ-$\mathbb{Q}$-Maschinen

In diesem Abschnitt erfährt das Maschinenmodell eine Erweiterung. Wählt man als Ring $\mathcal{R}$ den Ring der rationalen Zahlen $\mathbb{Q}$, dann sieht man schnell, daß die $\mathbb{Q}$-Maschinen bei

endlichen Berechnungen die gleiche Berechnungsmächtigkeit wie Turing-Maschinen haben, sofern man die Berechnungen von Turing-Maschinen auf nahe liegende Weise auf $\mathbb{Q}$ ausdehnt. Bei der Betrachtung unendlicher Berechnungen tritt hier wegen der Unvollständigkeit von $\mathbb{Q}$ jedoch der Fall auf, daß es unendliche Berechnungen gibt, bei denen die Ausgaben zwar als Cauchy-Folgen konvergieren, die aber keinen Grenzwert in $\mathbb{Q}$ haben. Ein Maschinenmodell, bei dem nur rationale Zahlen in den Registern und im Speicher auftreten können, das aber gleichwohl Funktionen $\mathbb{R} \to \mathbb{R}$ charakterisiert, ist aber als sinnvolle Einschränkung der allgemeinen analytischen Maschinen wünschenswert. Dazu erweitert man die $\mathbb{Q}$-Maschinen dahingehend, daß reelle Eingaben zugelassen werden und Resultate von konvergierenden rationalen Berechnungen reellwertig sein können. Um reelle Eingaben verarbeiten zu können, werden diese mit einer vorgegebenen Präzision gerundet.
Die folgende Konstruktion wird für die Körper $\mathbb{Q}$ und $\mathbb{R}$ der rationalen bzw. reellen Zahlen gemacht. Analog verfährt man, will man mit komplexen Zahlen arbeiten; hier verwendet man entsprechend die komplexen rationalen Zahlen $\mathbb{Q}[i]$ und die komplexen Zahlen $\mathbb{C}$.
Damit reellwertige Eingaben mit $\mathbb{Q}$-Maschinen verarbeitet werden können, müssen diese Eingaben gerundet werden. Dazu führt man *Rundungsfunktionen* ein:

Definition 2.1.4. *Eine Funktion* $\rho : \mathbb{N} \times \mathbb{R} \to \mathbb{Q}$ *heißt* Rundungsfunktion*, falls für alle* $x \in \mathbb{R}$ *und* $n \in \mathbb{N}$ *gilt:* $|\rho(n, x) - x| < 2^{-n}$.

Bei der Definition der δ-$\mathbb{Q}$-Maschinen wird zunächst von einer $\mathbb{Q}$-Maschine $\mathcal{M}$ mit Programm π ausgegangen, wobei zusätzlich eine Rundungsfunktion ρ gewählt wird, und das Eingabeband x nunmehr reelle Zahlen enthalten kann, also $x \in \mathbb{R}^{\mathbb{N}}$. Weiterhin können im Programm π auch reelle Konstanten verwendet werden.
Die Interpretation der Zuweisungen der Form $\alpha :=x_i$ bzw. $\alpha :=c$ für irrationale Konstanten c durch die Übergangsfunktion Δ wird nun durch die Zuweisungen $\alpha := \rho(\delta, x_i)$ bzw. $\alpha := \rho(\delta, x_i)$. Hierbei ist die Rundungsgenauigkeit δ gegeben durch den Inhalt des δ-Registers der Maschine. Mit Hilfe des Befehls `next` δ wird der Inhalt des δ-Registers um 1 erhöht, und die Maschine wird neu gestartet.

Definition 2.1.5. *Für ein Programm* $\pi \in \mathsf{Prog}^*$ *und eine Rundungsfunktion* ρ *heißt die durch obige Konstruktion festgelegte abstrakte Maschine*

$$\mathcal{M}_\rho^{\delta\text{-}\mathbb{Q}} = (K, K_s, K_e, K_t, \Delta, \mathbb{R}, \mathrm{in}, \mathrm{out})$$

die δ-$\mathbb{Q}$-Maschine mit Programm π und Rundung ρ.

Eine δ-$\mathbb{Q}$-Maschine ist nach dieser Definition stets an eine spezielle Rundungsfunktion gebunden. Maschinen, deren berechnete Funktionen unabhängig von der speziellen Rundungsfunktion sind, bezeichnen wir als *robust*. Bei robusten δ-$\mathbb{Q}$-Maschinen ergibt sich also für jede Eingabe und für jede Rundungsfunktion die gleiche berechnete Funktion. Da Eigenschaften der Rundungsfunktionen nicht im zentralen Interesse dieser Arbeit stehen, werden vor allem robuste δ-$\mathbb{Q}$-Maschinen betrachtet, und der Zusatz *robust* wird weggelassen. Chadzelek [Cha98] hat den Einfluß der Rundung auf die Berechnungsmächtigkeit der Maschinen eingehend untersucht.

2.2 Berechenbare Funktionen und grundlegende Eigenschaften

2.2.1 Berechenbare Funktionen und entscheidbare Mengen

Wir sind nun soweit, daß wir berechenbare und analytisch berechenbare Funktionen definieren können. Generell ist in dieser Arbeit der Sprachgebrauch der folgende: Unter *berechenbaren* Funktionen sind immer solche Funktionen zu verstehen, die mittels einer endlichen Berechnung berechnet werden können. Unter *analytisch berechenbaren* Funktionen verstehen wir Funktionen, die durch unendliche, konvergente Berechnungen berechnet werden können. Dabei ist natürlich stets noch das spezielle Maschinenmodell zu beachten.

Berechenbare Funktionen

Bei der Definition (analytisch) berechenbarer Funktionen über dem Ring $\mathcal{R}$ setzen wir voraus, daß die notwendigen Eigenschaften wie etwa das Vorhandensein einer Metrik auf $\mathcal{R}$ bei analytischer Berechenbarkeit gegeben sind. Das Interesse dieser Arbeit liegt ohnehin bei den Körpern $\mathbb{R}$ und $\mathbb{C}$, wo alle nötigen Voraussetzungen gegeben sind.

Definition 2.2.1. *Es sei $D \subseteq \mathcal{R}^*$, und es sei $f : D \to \mathcal{R}^*$ eine Funktion. Wir nennen f*

- $\mathcal{R}$-berechenbar*, falls es eine $\mathcal{R}$-Maschine $\mathcal{M}$ gibt mit $D \subseteq H_{\mathcal{M}}$ und $\Phi_{\mathcal{M}}|_D = f$,*
- *auf D* analytisch $\mathcal{R}$-berechenbar*, falls es eine $\mathcal{R}$-Maschine $\mathcal{M}$ gibt mit $D \subseteq D_{\mathcal{M}}$ und $\Phi_{\mathcal{M}}|_D = f$.*

Anmerkung. Bei der allgemeinen Definition berechenbarer Funktionen über dem Ring $\mathcal{R}$ sind auch beliebige Konstantenzuweisungen mit Konstanten aus $\mathcal{R}$ zugelassen. Für die Körper $\mathbb{R}$ und $\mathbb{C}$ haben wir die Problematik der Konstantenzuweisung bereits diskutiert, und verstehen unter (analytisch) berechenbaren Funktionen solche ohne nichtrationale Konstantenzuweisungen. Wenn Konstanten aus einer Konstantenmenge C verwendet werden, dann sprechen wir von (analytisch) $\mathbb{R}[C]$-berechenbaren Funktionen. Bei endlichen Mengen $\{c_1, \ldots, c_n\}$ schreiben wir $\mathbb{R}[c_1, \ldots, c_n]$.

Beispiel 2.2.2.

1. Die $\mathbb{Z}$-berechenbaren Funktionen sind gerade die Turing-berechenbaren Funktionen. Definiert man mit Hilfe klassischer Turing-Maschinen auf nahe liegende Weise Berechenbarkeit auf den rationalen Zahlen, so entsprechen die $\mathbb{Q}$-berechenbaren Funktionen gerade diesen (auf $\mathbb{Q}$ verallgemeinerten) Turing-berechenbaren Funktionen.

2. Polynome und gegebenenfalls rationale Funktionen sind auf natürliche Weise $\mathcal{R}$-berechenbar.

3. Da für eine $\mathcal{R}$-Maschine $H_{\mathcal{M}} \subseteq D_{\mathcal{M}}$ gilt, ist die Menge der berechenbaren Funktionen in der Menge der analytisch berechenbaren Funktionen enthalten.

Die $\mathcal{R}$-berechenbaren Funktionen werden durch den Darstellungssatz 2.2.22 in Abschnitt 2.2.2 vollständig charakterisiert.
Für den Rest des Abschnittes betrachten wir wieder die Körper $\mathbb{R}$ und $\mathbb{C}$. Zunächst definiert man analog für δ-$\mathbb{Q}$-Maschinen:

Definition 2.2.3. *Es sei $D \subseteq \mathbb{R}^*$, und es sei $f : D \to \mathbb{R}^*$ eine Funktion. Wir nennen f* (robust) δ-$\mathbb{Q}$-analytisch berechenbar*, wenn es eine (robuste) δ-$\mathbb{Q}$-Maschine $\mathcal{M}$ gibt mit $D_{\mathcal{M}} = D$ und $\Phi_{\mathcal{M}} = f$.*

Vor allem für δ-$\mathbb{Q}$-Maschinen, aber auch für $\mathbb{R}$-Maschinen führen wir eine weitere Klasse berechenbarer Funktionen ein. Im Zusammenhang mit dem Halte- und Konvergenzproblem wird sich zeigen, daß die analytisch berechenbaren Funktionen nicht unter Komposition abgeschlossen sind. Will man zwei analytische Maschinen hintereinander ausführen, so ist ein nahe liegender Ansatz, zunächst die erste Maschine eine bestimmte Anzahl von Schritten rechnen lassen und dann die zweite mit dem erzeugten Zwischenergebnis und dann das Verfahren mit einer immer größeren Anzahl von Schritten zu wiederholen. Dies scheitert jedoch an zwei Gründen: Zunächst ist es im allgemeinen wegen fehlender Stetigkeitsvoraussetzungen nicht möglich, mit einer approximierten Eingabe die Berechnung einer analytischen Maschine zu approximieren. Bei δ-$\mathbb{Q}$-Maschinen ist die Eingabe durch die Rundung auf natürliche Weise nur approximativ gegeben. Aber da über die Genauigkeit der Zwischenergebnisse keine Aussage gemacht werden kann, ist es nicht möglich, die Rundungsgenauigkeit zu bestimmen, mit der die zweite Maschine jeweils rechnen muß. Daher liegt es nahe, zusammen mit der Zwischenausgabe auch eine Aussage über die Genauigkeit dieser Ausgabe zu machen. Die Genauigkeitsangabe muß zusammen mit der Ausgabe geliefert werden. In der folgenden Definition wird daher die erste Komponente der Ausgabe als Genauigkeitsschranke interpretiert.

Definition 2.2.4. *Sei $\mathcal{M}$ eine $\mathbb{R}$-Maschine bzw. δ-$\mathbb{Q}$-Maschine und $b = (k_i)_{i\in\mathbb{N}}$ eine analytische Berechnung, die unendlich oft eine Zielkonfiguration annimmt. Die Ausgaben der Zielkonfigurationen seien gegeben durch $(y_0^{(j)}, \ldots, y_{n_j}^{(j)}) \in \mathbb{Q}^*$, und die k-te Komponente des Ergebnisses sei $y_k := \lim_{n\to\infty} y_k^{(n)}$. Dann heißt die Berechnung* stark*, falls*

1. *$|y_0^{(j)}| \to 0$ für $j \to \infty$ und*
2. *$\max_{1\le k\le n_j} |y_k^{(j)} - y_k| \le |y_0^{(j)}|$.*

Sie heißt quasi-stark*, falls die zweite Bedingung für* fast alle *j gilt.*
Eine Funktion f heißt nun stark δ-$\mathbb{Q}$-analytisch berechenbar *bzw.* quasi-stark δ-$\mathbb{Q}$-analytisch berechenbar*, falls es eine δ-$\mathbb{Q}$-Maschine $\mathcal{M}$ gibt, die f berechnet und jede analytische Berechnung von $\mathcal{M}$ stark bzw. quasi-stark ist. Wir lassen hier meist den Zusatz "berechenbar" weg und schreiben* (quasi-)stark δ-$\mathbb{Q}$-analytische Funktionen*. Wir sprechen gelegentlich auch von δ-$\mathbb{Q}$-*approximierbaren Funktionen.
Analog definieren wir (quasi-)stark $\mathbb{R}$-analytisch berechenbare Funktionen.

Der Unterschied zwischen stark berechenbar und quasi-stark berechenbar besteht also darin, daß die Genauigkeitsschranke bei starken Berechnungen immer gelten muß und bei quasi-starken Berechnungen erst von einem gewissen Berechnungszeitpunkt an. Die Idee der quasi-stark δ-$\mathbb{Q}$-analytischen Berechnung stammt von Vierke, die Eigenschaften der quasi-stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen wurden zuerst von Chadzelek [Cha98] ausgearbeitet.

Berechenbare Operatoren

Wir haben bereits gesehen, wie Maschinen als Eingaben von Maschinen und Funktionen aufgefaßt werden können. Ist zusätzlich bekannt, daß das Ergebnis einer solchen Funktion stets eine Kodierung einer Maschine ist, können wir von einem berechenbaren Operator sprechen:

Definition 2.2.5 (Berechenbare Operatoren)**.** *Es seien* $\mathbf{F}$ *und* $\mathbf{G}$ *Mengen analytisch berechenbarer Funktionen auf* $\mathbb{R}$ *und* $f : \mathbf{F} \to \mathbf{G}$ *ein Operator. Seien* $\mathbf{M}$ *und* $\mathbf{N}$ *Mengen analytischer Maschinen, die die Funktionsmengen abdecken. Dann heißt* f analytisch berechenbarer Operator*, wenn es eine analytische Maschine* $\mathcal{F}$ *gibt mit* $\mathbf{M} \subseteq D_{\mathcal{F}}$ *und Ausgaben in* $\mathbf{N}$*, so daß gilt*

$$\Phi_{\mathcal{F}(\mathcal{M})} = f(\Phi_{\mathcal{M}}) \text{ für alle } \mathcal{M} \in \mathbf{M}$$

Anmerkung. Wir werden im weiteren Verlauf den Begriff des berechenbaren Operators etwas weiter fassen als in der obigen Definition. So werden wir nicht nur Mengen analytisch berechenbarer Funktionen betrachten, sondern auch Mengen δ-$\mathbb{Q}$ analytisch berechenbarer Funktionen und Funktionsklassen für weitere Maschinenmodelle. Auch in diesen Fällen werden wir von berechenbaren Operatoren sprechen, und die Art der Berechenbarkeit des Operators ist gegeben durch die Maschine, die den Operator realisiert.

Entscheidbare Mengen

Ist $D \subseteq \mathcal{R}^*$, so ist die *charakteristische Funktion* χ_D *von* D definiert durch

$$\chi_D : \mathcal{R}^* \to \{0,1\}, x \mapsto \begin{cases} 1 & x \in D \\ 0 & x \notin D \end{cases}$$

Ist $D \subseteq \mathcal{R}^n$, so ist natürlich auch eine charakteristische Funktion $\mathcal{R}^n \to \{0,1\}$ definiert. Es geht jedoch stets aus dem Zusammenhang hervor, auf welcher Menge eine verwendete charakteristische Funktion definiert ist, und daher wird darauf nicht mehr explizit eingegangen.

Definition 2.2.6. *Sei* $D \subseteq \mathcal{R}^*$*.* D *heißt*

- $\mathcal{R}$-entscheidbar*, falls* χ_D $\mathcal{R}$*-berechenbar ist.*
- analytisch $\mathcal{R}$-entscheidbar*, falls* χ_D *analytisch* $\mathcal{R}$*-berechenbar ist.*

Anmerkung. Bei Turing-Maschinen sind Berechenbarkeit von Funktionen und Entscheidbarkeit von Mengen äquivalente Konzepte. Die Turing-Entscheidbarkeit einer Menge ist äquivalent zur Berechenbarkeit ihrer charakteristischen Funktion. Umgekehrt ist eine Funktion $f : \mathbb{N} \to \mathbb{N}$ genau dann Turing-berechenbar, wenn ihr Graph $\{(n, f(n)) \,|\, n \in \mathbb{N}\} \subseteq \mathbb{N}^2$ eine Turing-entscheidbare Menge ist. Dies ist bei $\mathcal{R}$-Maschinen nicht mehr der Fall. Beispielsweise ist die Funktion $\sqrt{\cdot} : \mathbb{R}^+ \to \mathbb{R}$ nicht $\mathbb{R}$-berechenbar (vgl. Abschnitt 2.2.3), aber die Menge $\{(x, \sqrt{x}) \,|\, x \in \mathbb{R}^+\}$ ist $\mathbb{R}$-entscheidbar.

Abschlußeigenschaften berechenbarer Funktionen

Ist $f : \mathbb{R}^k \to \mathbb{R}^n$ eine $\mathbb{R}$-berechenbare Funktion, so ist es auch $p_j f$, die Projektion auf die j-te Komponente. Dazu muß eine Maschine einfach nur stets die j-te Zelle des Ausgabebandes ausgeben. Das gleiche gilt für $\mathbb{R}$-analytische Maschinen.

Satz 2.2.7. *Die Mengen der $\mathbb{R}$-berechenbaren Funktionen und der analytisch $\mathbb{R}$-berechenbaren Funktionen sind abgeschlossen unter*

- *Projektion und*
- *kartesischem Produkt.*

Die Menge der $\mathbb{R}$-berechenbaren Funktionen ist zudem abgeschlossen unter Komposition.

Beweis. Die Abgeschlossenheit der $\mathbb{R}$-berechenbaren Funktionen unter kartesischem Produkt ergibt sich sehr einfach: Bei einer Eingabe $x_1, \ldots, x_n$ wird nacheinander das Ergebnis der am Produkt beteiligten Funktionen berechnet, und alle Ergebnisse auf dem Ausgabeband ausgegeben.
Für analytisch $\mathbb{R}$-berechenbare Funktionen ist der Beweis nicht ganz unmittelbar, da man die Maschinen aufgrund ihrer unendlichen Berechnungen nicht sequentiell ausführen kann. Mit Hilfe von *parallelen Berechnungen*, einer auch in der klassischen Informatik häufig angewandten Technik kann die Berechnung mehrerer analytischer Maschinen jedoch simuliert werden. Der folgende Beweis und der Beweis zu Satz 2.2.11 werden exemplarisch etwas ausführlicher gestaltet, um dem Leser ein Gefühl für die Technik zu vermitteln. Der Übersicht halber wird der Beweis nur für zwei Funktionen bzw. Maschinen geführt, er läßt sich auf offensichtliche Weise auf endlich viele und sogar auf unendlich viele Maschinen verallgemeinern. Seien f, g analytisch $\mathbb{R}$-berechenbar mit berechnenden Maschinen $\mathcal{M}_f, \mathcal{M}_g$. Wir beschreiben im folgenden die Maschine $\mathcal{M}_{f\times g}$, die das kartesische Produkt $f \times g : (x, y) \mapsto (f(x), g(y))$ berechnet. $\mathcal{M}_{f\times g}$ berechnet die Approximationen von $f(x)$ und $g(y)$ parallel. Dazu wird eine allgemeine Zählvariable n in einer Schleife inkrementiert, und in jedem Durchlauf werden n Schritte der Berechnung von $\mathcal{M}_f$ auf x und n Schritte der Berechnung von $\mathcal{M}_g$ auf y simuliert. Die n-ten Approximationen $f^{(n)}(x)$ und $g^{(n)}(y)$ werden nun auf die ersten beiden Zellen des Ausgabebandes geschrieben, n um 1 inkrementiert und die Schleife von neuem durchlaufen. Auf diese Weise konvergiert der Inhalt des Ausgabebandes gegen $(f(x), g(y))$.
Die Abgeschlossenheit der $\mathbb{R}$-berechenbaren Funktionen unter Komposition ergibt sich auf naive Weise durch Simulation: Sind f, g $\mathbb{R}$-berechenbar mit $D_g \subseteq f(D_f)$, so berechnet die Maschine $\mathcal{M}_{g\circ f}$ für jede Eingabe x zunächst den Funktionswert $f(x)$ und wendet dann g auf diesen Wert an. Dies ist möglich, da bei jeder Eingabe die Maschine $\mathcal{M}_f$, die f berechnet, nur endlich viele Schritte benötigt. □

Im allgemeinen sind analytisch berechenbare Funktionen nicht unter Komposition abgeschlossen. Dies zeigt sich im Zusammenhang mit dem Konvergenzproblem, siehe Satz 2.2.12. Die Komposition ist auch dann nicht allgemein möglich, wenn die zweite ausgeführte Funktion lediglich endlich berechenbar ist. Es wird zusätzlich die Stetigkeit der zweiten Funktion benötigt:

Lemma 2.2.8. *Sei $g : D \to \mathbb{R}^i$ analytisch berechenbar und $g : \mathbb{R}^i \to \mathbb{R}^k$ eine $\mathbb{R}$-berechenbare und stetige Funktion. Dann ist $g \circ f$ analytisch berechenbar.*

Beweis. Es seien $\mathcal{M}_f$ bzw. $\mathcal{M}_g$ Maschinen für f bzw. g. Auf Eingabe x simuliert die Maschine für die Komposition $\mathcal{M}_h$ für wachsendes n die Maschine $\mathcal{M}_f$ bis zur n-ten Ausgabe und berechnet dann $g(\mathcal{M}_f^{(n)})$. Dies ist möglich, da $\mathcal{M}_g$ nur endliche Zeit rechnet. Mit der Stetigkeit von g folgt, daß die Ausgabe von $\mathcal{M}_h$ gegen $g(f(x))$ konvergiert.
Daß die Stetigkeit auch eine notwendige Bedingung ist, zeigt Korollar 2.2.15. □

Anmerkung. Dieses Lemma zeigt, daß endliche, stetige Operationen nach einer unendlichen Berechnung ausgeführt werden können. Es läßt sich auf den Fall erweitern, daß f Werte in $\mathbb{R}^*$ hat und g dort definiert ist. Nach unserem Konvergenzbegriff auf $\mathbb{R}^*$ muß bei Konvergenz die Länge der Ausgabefolge der Maschine für f konstant werden, und sobald dieser Berechnungszeitpunkt erreicht ist, funktioniert die Argumentation wie im ursprünglichen Fall.

Für die quasi-stark δ-$\mathbb{Q}$-analytisch berechenbarern Funktionen ergibt sich

Lemma 2.2.9. *Die quasi-stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen sind unter Komposition abgeschlossen. Dies gilt auch für die stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen.*

Beweis. Wir verweisen auf [Cha98]. Die Idee besteht darin, die zweite Maschine fortlaufend zu simulieren und immer, wenn sie auf eine Eingabe zurückgreift (die Ergebnis der Berechnung der ersten Maschine ist), die erste Maschine so lange zu simulieren, bis die Ausgabegenauigkeit gleich der Rundungsgenauigkeit δ der zweiten Maschine ist. Dabei wird nur endlich oft ein Fehler gemacht, da irgendwann die Genauigkeitsschranke exakt ist.
Für die stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen sieht man dies noch leichter, da die Genauigkeitsschranke hier immer korrekt ist. □

Halte- und Konvergenzproblem

Für die klassische Berechenbarkeitstheorie ist das *Halteproblem* von zentraler Bedeutung: Gegeben eine Turing-Maschine T, kann man entscheiden, ob T auf einer Eingabe n hält oder nicht? Bekanntlich ist das *Halteproblem* für Turing-Maschinen unlösbar. Dabei wird das Problem zunächst als Entscheidbarkeitsproblem für Turing-Maschinen formuliert. Dann wird mit Hilfe eines Diagonalisierungsargumentes gezeigt, daß das Problem unlösbar ist.
Das Problem kann auf einfache Weise auf $\mathbb{R}$-Maschinen übertragen werden. Gibt es eine $\mathbb{R}$-Maschine, die bei Eingabe eines Tupels $(\mathcal{M}, x)$ einer kodierten $\mathbb{R}$-Maschine $\mathcal{M}$ und einer Eingabe x mit Hilfe einer endlichen Berechnung entscheidet, ob $\mathcal{M}$ auf der Eingabe x hält oder nicht? Das *Halteproblem für $\mathbb{R}$-Maschinen* ist also die Frage, ob die Funktion

$$h : \mathbf{M}_{\mathbb{R}} \times \mathbb{R} \to \mathbb{R},\ f(\mathcal{M}, x) = \begin{cases} 1 & : \quad \mathcal{M} \text{ hält auf Eingabe } x \\ 0 & : \quad \text{sonst} \end{cases}$$

$\mathbb{R}$-berechenbar ist. Genau wie im Fall der Turing-Maschinen zeigt man mittels Diagonalisierung, daß diese Funktion nicht berechenbar ist:
Gäbe es eine Maschine $\mathcal{H}$, die h berechnet, so könnte man daraus eine Maschine $\mathcal{W}$ konstruieren, die bei der Eingabe x $\mathcal{H}$ auf Eingabe (x, x) simuliert und in eine Endlosschleife

verzweigt (also nicht hält), wenn $\mathcal{H}$ 1 ausgibt und 1 ausgibt, wenn $\mathcal{H}$ 0 ausgibt. Nun überlegt man, ob $\mathcal{W}$ mit Eingabe $\mathcal{W}$ hält oder nicht, und sieht, daß beide Alternativen zu einem Widerspruch führen.
Mittels analytischer Berechnungen ist das Halteproblem für $\mathbb{R}$-Maschinen jedoch entscheidbar: Die Maschine $\mathcal{H}$ simuliert bei Eingabe eines Tupels $(\mathcal{M}, x)$ wie oben die Maschine $\mathcal{M}$ auf der Eingabe x unendlich lange. Dabei gibt sie auf dem Ausgabeband stets 0 für "hält nicht" aus. Erst, wenn $\mathcal{M}$ in eine Haltekonfiguration übergeht, gibt $\mathcal{H}$ eine 1 auf ihrem Ausgabeband aus, und dies setzt sie in diesem Fall unendlich lange fort. Auf diese Weise konvergiert die Ausgabe von $\mathcal{H}$ gegen 1, falls $\mathcal{M}$ auf x hält, und andernfalls gegen 0. Durch Simulation sieht man auch, daß das Halteproblem für klassische Turing-Maschinen analytisch entscheidbar ist.
Während das *Halte*problem für $\mathbb{R}$-Maschinen noch analytisch entscheidbar ist, ist es das *Konvergenz*problem für $\mathbb{R}$-Maschinen nicht mehr: Es gibt keine $\mathbb{R}$-Maschine $\mathcal{K}$, die bei Eingabe eines Tupels $(\mathcal{M}, x)$ gegen 1 konvergiert, falls $\mathcal{M}$ auf Eingabe x eine konvergente analytische Berechnung durchführt und andernfalls gegen 0.
Konkret heißt dies:

Satz 2.2.10. *Die Funktion*

$$f : \mathbf{M}_{\mathbb{R}} \times \mathbb{R} \to \mathbb{R},\ f(\mathcal{M}, x) = \begin{cases} 1 & : \quad \mathcal{M} \textit{ konvergiert auf Eingabe } x \\ 0 & : \quad \textit{sonst} \end{cases}$$

ist nicht analytisch berechenbar.

Beweis. Dieses Problem wurde bereits von Vierke [Vie96] und Chadzelek [Cha98] untersucht. Wir geben zur Veranschaulichung einen einfachen Beweis an:
Angenommen, es gäbe eine Maschine $\mathcal{K}$, die das Konvergenzproblem für analytische Maschinen entscheidet. Wir konstruieren daraus eine Maschine $\mathcal{W}$, die bei Eingabe x zunächst $\mathcal{K}$ auf (x, x) simuliert. Bei jeder Ausgabe von $\mathcal{K}$, die größer als $\frac{2}{3}$ ist, gibt $\mathcal{W}$ hintereinander eine 0 und eine 1 aus und bei jeder Ausgabe von $\mathcal{K}$, die kleiner als $\frac{1}{3}$ ist, gibt $\mathcal{W}$ 1 aus. Ist nun $\mathcal{W}$ mit Eingabe $\mathcal{W}$ konvergent? Ist dies der Fall, dann muß die Ausgabe von $\mathcal{K}$ bei Eingabe $(\mathcal{W}, \mathcal{W})$ gegen 1 konvergieren. Dann ist diese Ausgabe fast immer größer als $\frac{2}{3}$. In diesem Fall erzeugt $\mathcal{W}$ aber eine (nicht konvergente) Folge von 0 und 1 als Ausgabe, es liegt also ein Widerspruch vor. Im anderen Fall gelangt man analog zu einem Widerspruch. □

In Analogie zu den Turing-Maschinen folgt nun, daß auch das *Konvergenzproblem bei leerer Eingabe* nicht analytisch entscheidbar ist, d.h. daß die Funktion

$$f : \mathbf{M}_{\mathbb{R}} \to \mathbb{R},\ f(\mathcal{M}) = \begin{cases} 1 & : \quad \mathcal{M} \text{ konvergiert} \\ 0 & : \quad \text{sonst} \end{cases}$$

nicht analytisch berechenbar ist. Wäre die Funktion nämlich berechenbar, so könnte das Halteproblem mit Eingabe x gelöst werden, indem aus einer Maschine $\mathcal{M}$ mit Eingabe x eine Maschine $\mathcal{M}_x$ konstruiert wird, deren Programm die Eingabe x als Konstante enthält und die mit der Konstanten operiert wie $\mathcal{M}$ mit ihrer Eingabe.

Anmerkung. Bei Verzicht auf allgemeine reelle Konstanten ist die Analogie zu den Turing-Maschinen jedoch nicht ganz unmittelbar, da eine beliebige reelle Eingabe x nicht als natürliche Zahl kodiert werden kann. Im Beweis zu Satz 2.2.10 wird jedoch für die Diagonalisierung als Eingabe nur die Kodierung

einer Maschine verwendet, und bei Beschränkung auf natürliche Konstanten erfolgt die Kodierung der Maschinen in die natürlichen Zahlen.

Vierke konnte zeigen [Vie96], daß das Konvergenzproblem für analytische Maschinen durch drei hintereinandergeschaltete analytische Maschinen lösbar ist, woraus auch unmittelbar folgt, daß die Menge der *analytisch berechenbaren Funktionen nicht unter Komposition abgeschlossen* ist. Die Weiterführung dieser Überlegung führt zum Hierarchiesatz für $\mathbb{R}$-analytische Maschinen 2.2.13.

Chadzelek stellte die Frage, ob das Konvergenzproblem statt mit dreier auch mit Hilfe zweier hintereinander geschalteter Maschinen lösbar ist. Er konnte dies nur unter der Zuhilfenahme von unendlichen Zwischenergebnissen bejahen. Wir gehen hier etwas weiter und zeigen, daß sogar zwei hintereinander geschaltete analytische Maschinen genügen, deren Ausgabe sämtlich in $\mathbb{R}$ liegt. Durch dieses Resultat wird auch das erste der in [Cha98] offenen Probleme gelöst.

Satz 2.2.11. *Das Konvergenzproblem für analytische Maschinen ist durch zwei hintereinander ausgeführte analytische Maschinen entscheidbar.*

Beweis. Wir bezeichnen die zuerst ausgeführte Maschine mit $\mathcal{M}_1$. Es sei $\mathcal{M}$ die Eingabemaschine, von der die Konvergenz entschieden werden soll. Die Folge $(a_n)_{n\in\mathbb{N}}$ sei die Ausgabefolge der Zielkonfigurationen von $\mathcal{M}$. Sei für $k \in \mathbb{N}$

$$b_k := \sup_{n>m\geq k} |a_n - a_m| \tag{2.1}$$

Es gilt nun:

$$(a_n)_{n\in\mathbb{N}} \text{ konvergent} \quad \Leftrightarrow \quad \lim_{k\to\infty} b_k = 0.$$

Um nun zu überprüfen, ob die b_k eine Nullfolge bilden, wird nun lediglich die nächsthöhere Zweierpotenz als obere Schranke gespeichert. Da uns nur das Verhalten der Folge für kleine Werte von b_k interessiert, werden b_k mit $b_k \geq 1$ durch 1 ersetzt. Damit der Fall $b_k = 0$, der bei konstanten und fast überall konstanten Folgen auftritt, abgedeckt ist, wird in diesem Fall 2^{-k} als obere Schranke genommen.

Insgesamt erhalten wir als Repräsentation

$$\tilde{b_k} = r(b_k, k) := \begin{cases} \max\{j \leq k \,:\, 2^{-j} \geq b_k\} & : \quad b_k < 1 \\ 0 & : \quad \text{sonst} \end{cases} \tag{2.2}$$

Es gilt nun für $\tilde{b_k}$:

$$(a_n)_{n\in\mathbb{N}} \text{ konvergent} \quad \Leftrightarrow \quad \lim_{k\to\infty} \tilde{b}_k = \infty.$$

Die $\tilde{b_k}$ werden nun mit der Methode der *parallelen Berechnung* berechnet. Da diese Methode in dieser Arbeit sehr häufig zur Anwendung kommen wird, erfolgt hier exemplarisch eine etwas detailliertere Darstellung. Ziel ist, nach (abzählbar) unendlich vielen Rechenschritten über die Werte sämtlicher $\tilde{b_k}$ für alle $k \in \mathbb{N}$ zu verfügen. Die Anzahl der Rechenschritte, die benötigt wird, um die $\tilde{b_k}$ zu berechnen, ist aber unbeschränkt. Daher können die $\tilde{b_k}$ nicht nacheinander für $k = 1, 2, \ldots$ berechnet werden, sondern deren

Berechnung muß *parallel* erfolgen. Dazu wird in einer äußeren Endlosschleife der Parameter k beginnend bei 1 fortlaufend immer weiter inkrementiert. Für jedes $j < k$ wird eine aktuelle Approximation $\tilde{b}_{j,k}$ von $\tilde{b}_j$ (die "Approximation von $\tilde{b}_j$ beim k-ten Schleifendurchlauf der äußeren Schleife") geführt. $\tilde{b}_{j,k}$ wird als natürliche Zahl gespeichert, die die entsprechende Zweierpotenz repräsentiert. In jeder neuen Runde der äußeren Endlosschleife wird für das neue k der Wert $\tilde{b}_{k,k} = r(|a_k - a_{k+1}|, k)$ initialisiert. Dann durchläuft in einer inneren Schleife der Parameter j alle Werte von 1 bis k. In dieser inneren Schleife werden nun für alle l mit $j \leq l < k+1$ die Werte $|a_l - a_{k+1}|$ berechnet, und wenn $|a_l - a_{k+1}| > 2^{-\tilde{b}_{j,k}}$ ist, dann wird $\tilde{b}_{j,k}$ entsprechend verringert, außer im Falle $\tilde{b}_{j,k} = 0$, wo der Wert nicht mehr verändert wird. Nachdem die innere Schleife komplett durchlaufen wurde, gilt für $1 \leq j \leq k : \tilde{b}_{j,k} = r(\max_{j \leq n < m \leq k+1} |a_n - a_m|, k)$ mit der in Gleichung 2.2 definierten Repräsentation r.
Auf diese Weise werden die $\tilde{b}_j$ durch die $\tilde{b}_{j,k}$ parallel approximiert. Da diese durch natürliche Zahlen repräsentiert werden und im Laufe der Berechnung höchstens verringert werden, wird für jedes j die Berechnung stationär, d.h. es gibt für jedes j ein k_j mit $\tilde{b}_{k,j} = \tilde{b}_j$ für $k \leq k_j$.
Auf dem Ausgabeband gibt $\mathcal{M}_1$ die $\tilde{b}_{k,j}$ in jedem Schleifendurchlauf innerhalb einer reellen Zahl unär codiert aus, wobei die einzelnen Zahlen durch Nullen getrennt werden: Ist $u(n) \in \{0,1\}^*$ die unäre Darstellung für eine natürliche Zahl n ($u(1) = 1, u(2) = 11, u(3) = 111, \ldots$), so ist die Binärdarstellung der Zwischenausgaben der Berechnung von $\mathcal{M}_1$ nach dem k-ten Durchlauf der äußeren Schleife gleich

$$0u(\tilde{b}_{1,k})0u(\tilde{b}_{2,k})0u(\tilde{b}_{3,k})0 \ldots 0u(\tilde{b}_{k,k}).$$

Da die Berechnung für jedes $\tilde{b}_{j,k}$ stationär wird, konvergieren die Ausgaben gegen eine reelle Zahl. Es gilt also:

$$\mathcal{M}_1(\mathcal{M}) = 0u(\tilde{b}_1)0u(\tilde{b}_2)0u(\tilde{b}_3)0 \ldots$$

Es gilt nun

$$(a_n)_{n\in\mathbb{N}} \text{ konvergent} \quad \Leftrightarrow \quad \mathcal{M}_1(\mathcal{M}) \text{ irrational.}$$

Ist nämlich $(a_n)_{n\in\mathbb{N}}$ divergent, dann wird die Folge $(\tilde{b}_k)_{k\in\mathbb{N}}$ irgendwann stationär sein, denn ansonsten würde $(b_k)_{k\in\mathbb{N}}$ gegen 0 konvergieren. Dann hat aber $\mathcal{M}_1(\mathcal{M})$ ab einer hinreichend großen Stelle eine periodische Binärbruchdarstellung, ist also rational. Ist umgekehrt $(a_n)_{n\in\mathbb{N}}$ konvergent, dann wird b_k für hinreichend große k jede positive Zahl unterschreiten und somit die unäre Kodierung von $\tilde{b}_k$ immer länger werden, $\mathcal{M}_1(\mathcal{M})$ kann in diesem Fall also nicht periodisch und somit nicht rational sein.
$\mathcal{M}_2$ ist nun einfach eine Maschine, die überprüft, ob ihre Eingabe eine rationale Zahl ist oder nicht. Dies wird realisiert, indem alle rationalen Zahlen aufgezählt werden, und immer, wenn die aktuell aufgezählte Zahl ungleich der Eingabe ist, eine 1 ausgegeben wird, und sobald diese Zahl gleich der Eingabe ist, fortwährend nur noch 0 in einer Endlosschleife. □

Anmerkung.(Parallele Berechnungen) Die im obigen Beweis verwendete Technik der *parallelen Berechnung* wird in ähnlicher Form im weiteren Verlauf der Arbeit häufiger verwendet. Es können endlich viele Maschinen parallel simuliert werden, und bei geeigneten Aufzählungen können sogar abzählbar

unendlich viele Maschinen parallel simuliert werden. Dabei ist die Vorgehensweise wie im obigen Beweis immer die, daß die Berechnungstiefe jeder einzelnen simulierten Maschine immer weiter erhöht wird, und dies für alle simulierten Maschinen parallel gemacht wird.
Es können auch unterschiedliche Maschinen zu unterschiedlichen Berechnungstiefen simuliert werden. Ist etwa eine Aufzählung $\{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$ gegeben, d.h. eine Maschine $\mathcal{M}$, die bei Eingabe eines Paares (n, x) die Maschine $\mathcal{M}_n$ auf Eingabe x berechnet, so ist eine parallele Simulation aller (unendlich vieler) Maschinen dadurch denkbar, daß zunächst die erste Maschine einen Berechnungsschritt lang, dann die ersten beiden Maschinen zwei Berechnungsschritte lang, dann die ersten drei Maschinen drei Berechnungsschritte lang usw. simuliert werden.
Diese Methode, die wir hier parallele Berechnung nennen, findet auch in der klassischen Berechenbarkeitstheorie häufig in ähnlicher Form Anwendung, wobei hier natürlich keine unendlich langen Berechnungen berücksichtigt werden. Die Methode wird beispielsweise beim klassischen Beweis der Äquivalenz deterministischer und nichtdeterministischer Turing-Maschinen verwendet. In der englischsprachigen Literatur wird meist von *Dovetailing* gesprochen.

Das Konvergenzproblem für analytische Maschinen ist also durch zwei hintereinander ausgeführte analytische Maschinen entscheidbar. Da es aber nicht durch eine Maschine analytisch entscheidbar ist, folgt sofort, daß die Menge der analytisch berechenbaren Funktionen nicht unter Komposition abgeschlossen ist.

Korollar 2.2.12. *Die Menge der analytisch berechenbaren Funktionen ist nicht unter Komposition abgeschlossen.*

Eine analoge Argumentation zeigt, daß auch die Menge der δ-$\mathbb{Q}$-analytischen Funktionen nicht unter Komposition abgeschlossen ist. Das Argument kann sogar noch weiter geführt werden: Betrachtet man die Mengen der Funktionen, die durch Hintereinanderausführung von n analytischen Maschinen berechenbar sind, dann kann man zeigen, daß diese Mengen für aufsteigende n eine Hierarchie bilden:

Satz 2.2.13 (Hierarchiesatz für analytische Maschinen, [Cha98])**.** *Es bezeichne $\{\mathbb{R}^i$-analytisch$\}$ die Klasse der Funktionen, die durch Hintereinanderausführung von i analytisch $\mathbb{R}$-berechenbaren Funktionen darstellbar sind. Dann gilt für alle $i \in \mathbb{N}$:*

$$\{\mathbb{R}^i\text{-analytisch}\} \subsetneq \{\mathbb{R}^{i+1}\text{-analytisch}\}$$

Analoges gilt für δ-$\mathbb{Q}$-Maschinen:

$$\{\delta\text{-}\mathbb{Q}^i\text{-analytisch}\} \subsetneq \{\delta\text{-}\mathbb{Q}^{i+1}\text{-analytisch}\}$$

Beschränktheitsproblem; Häufungswerte

Wir haben gesehen, daß das Konvergenzproblem für analytische Maschinen nicht analytisch entscheidbar ist. Die Konvergenz bezieht sich hier auf die Folge der Ausgaben der Zielkofigurationen einer Maschine. Eine etwas einfachere Frage ist die, ob diese Folge eine beschränkte Folge bildet. Wir bezeichnen dies als *Beschränktheitsproblem für analytische Maschinen*. Auch dieses Problem ist nicht analytisch entscheidbar:

Satz 2.2.14. *Für eine Maschine $\mathcal{M}$ bezeichne $(\mathcal{M}^{(n)}(x))_{n\in\mathbb{N}}$ die Ausgabefolge ihrer Zielkonfigurationen bei Eingabe x. Dann ist die Funktion*

$$f : \mathbf{M}_{\mathbb{R}} \times \mathbb{R} \to \mathbb{R},\ f(\mathcal{M}, x) = \begin{cases} 1 & : \quad (\mathcal{M}^{(n)}(x))_{n\in\mathbb{N}} \text{ beschränkt} \\ 0 & : \quad \text{sonst} \end{cases}$$

nicht analytisch berechenbar.

Beweis. Wir nehmen das Gegenteil an, daß also eine Maschine $\mathcal{B}$ existiert, die entscheidet, ob eine gegebene Maschine $\mathcal{M}$ beschränkte Zwischenausgaben hat. Es muß also gelten (analytische Berechnung):

$$\mathcal{B}(\mathcal{M}, x) = \begin{cases} 1 & : \quad (\mathcal{M}^{(n)}(x))_{n\in\mathbb{N}} \text{ beschränkt} \\ 0 & : \quad \text{sonst} \end{cases}$$

Die Maschine $\mathcal{W}$ sei nun wie folgt definiert: $\mathcal{W}$ erhält als Eingabe x und simuliert $\mathcal{B}$ auf der Eingabe (x, x). Dabei wird ein separater Zähler N mitgeführt. Immer, wenn das Zwischenergebnis von $\mathcal{B}$ größer als $\frac{2}{3}$ ist, gibt $\mathcal{W}$ die Zahl N aus, inkrementiert diese und führt die Simulation fort. Falls während der Simulation das Zwischenergebnis von $\mathcal{B}$ kleiner als $\frac{1}{3}$ ist, gibt $\mathcal{W}$ die Zahl 0 aus.
Ist nun $\mathcal{W}(\mathcal{W})$ beschränkt? Wir unterscheiden zwei Fälle:

- $\mathcal{W}(\mathcal{W})$ ist beschränkt. Dann müssen die Zwischenergebnisse der simulierten Maschine $\mathcal{B}$ angesetzt auf $\mathcal{W}$ irgendwann stets größer als $\frac{2}{3}$ sein, da diese ja gegen 1 konvergieren. Dann wird die Ausgabe von $\mathcal{W}$ aber nach Konstruktion über alle Grenzen wachsen.

- $\mathcal{W}(\mathcal{W})$ ist nicht beschränkt. Analog müssen nun die Zwischenergebnisse von $\mathcal{B}$ irgendwann stets kleiner als $\frac{1}{3}$ sein, und $\mathcal{W}$ gibt nur noch 0 aus, ist also beschränkt.

□

Wie bei der Vorgehensweise zum Konvergenzproblem sieht man auch, daß das *Beschränktheitsproblem auf leerem Band* nicht entscheidbar ist, d.h. die Funktion

$$\mathcal{B}(\mathcal{M}) = \begin{cases} 1 & : \quad (\mathcal{M}^{(n)})_{n\in\mathbb{N}} \text{ beschränkt} \\ 0 & : \quad \text{sonst} \end{cases}$$

ist nicht berechenbar.
Durch Reduktion auf das Beschränktheitsproblem können wir nun zeigen, daß die Voraussetzung der Stetigkeit in Lemma 2.2.8 notwendig ist:

Korollar 2.2.15. *Die Komposition $g \circ f$ einer analytisch berechenbaren Funktion f mit einer endlich berechenbaren Funktion g ist im allgemeinen nicht analytisch berechenbar.*

Beweis. Wir nehmen das Gegenteil an und zeigen, wie wir mit einer solchen Komposition das Beschränktheitsproblem für analytische Maschinen lösen können. Sei $\mathcal{M}$ eine Maschine, von der die Beschränktheit entschieden werden soll. Die zuerst ausgeführte analytische Maschine überprüft nun für $n = 1, 2, \ldots$, ob die Zwischenausgaben von $\mathcal{M}$ n überschreiten. Es wird binär kodiert in einer reellen Zahl gespeichert, wenn eine Schranke n überschritten wurde. Wenn bei der Simulation die nächste Zwischenausgabe von $\mathcal{M}$ erreicht ist, wird in jedem Fall der aktuelle Stand ausgegeben. Ist $\mathcal{M}$ nun unbeschränkt, so konvergiert die Ausgabe gegen 1, andernfalls gegen eine von 1 verschiedene reelle Zahl. Dies kann nun leicht mit einer dahinter geschalteten $\mathbb{R}$-Maschine überprüft werden und 1 bzw. 0 ausgegeben werden. □

Eine weitere Anwendung der Reduktion auf das Beschränktheitsproblem zeigt, daß der größte Häufungswert einer berechenbaren Folge nicht analytisch berechenbar ist. Dazu nennen wir eine Folge $(a_n)_{n\in\mathbb{N}} \in \mathbb{R}^{\mathbb{N}}$

- $\mathbb{R}$-berechenbar, wenn die Funktion $\mathbb{N} \to \mathbb{R}, n \mapsto a_n$ $\mathbb{R}$-berechenbar ist und
- analytisch $\mathbb{R}$-berechenbar, wenn die Funktion $\mathbb{N} \to \mathbb{R}, n \mapsto a_n$ analytisch $\mathbb{R}$-berechenbar ist.

Wenn der größte Häufungswert einer $\mathbb{R}$-berechenbaren Folge nicht analytisch berechenbar ist, dann gilt dies erst recht für den größten Häufungswert einer analytisch $\mathbb{R}$-berechenbaren Folge. Da der Beweis der ersten Aussage aber eine Verfeinerung des Beweises der zweiten ist, beweisen wir beide Aussagen separat.

Korollar 2.2.16. *Der größte Häufungswert einer beschränkten analytisch berechenbaren Folge ist im allgemeinen nicht analytisch berechenbar. Genauer: Es gibt keine Maschine, die bei Eingabe einer analytischen Maschine, die eine Folge $(a_n)_{n\in\mathbb{N}}$ berechnet, den Grenzwert $\limsup_{n\to\infty} a_n$ berechnet.*

Beweis. Wir führen das Problem auf das Beschränktheitsproblem zurück. Wir nehmen an, es gebe eine Maschine $\mathcal{L}$, die den größten Häufungswert einer analytisch berechenbaren Folge berechnet. Ist nun eine Maschine $\mathcal{M}$ gegeben, konstruieren wir daraus eine Maschine $\mathcal{M}_B$, die eine Folge berechnet (d.h. $\mathcal{M}_B$ konvergiert auf $n \in \mathbb{N}$) und für die gilt:

$$\limsup_{n\to\infty} \mathcal{M}_B(n) = \begin{cases} 0 & : \quad (\mathcal{M}^{(n)})_{n\in\mathbb{N}} \text{ beschränkt} \\ 1 & : \quad \text{sonst} \end{cases}$$

Sei $\mathcal{M}$ die Eingabemaschine. Die Maschine $\mathcal{M}_B$, simuliert nun bei Eingabe n die Maschine $\mathcal{M}$ und es wird für $k = 1, 2, \ldots$ überprüft, ob $\mathcal{M}^{(k)} \geq n$ ist. Ist die Ungleichung für ein konkretes k erfüllt, wird 1 ausgegeben und in einen Terminalzustand übergegangen. Andernfalls wird 0 ausgegeben und k inkrementiert. Ist $\mathcal{M}$ beschränkt, so gibt es ein n_0, so daß die Maschine $\mathcal{M}_B$ bei Eingabe $n \geq n_0$ in ihrer unendlich andauernden Berechnung immer 0 ausgibt, da die Schranke n für kein k durch $\mathcal{M}^{(n)}$ überschritten wurde. Im anderen Fall ist für alle n das Ergebnis $\mathcal{M}_B(n) = 1$, da für jedes n ein k existiert mit $\mathcal{M}^{(k)} \geq n$. Eine Maschine, die das Beschränktheitsproblem löst, konstruiert aus ihrer Eingabemaschine $\mathcal{M}$ zunächst die Maschine $\mathcal{M}_B$ (diese Konstruktion ist offenbar $\mathbb{R}$-berechenbar) und setzt darauf die Maschine $\mathcal{L}$ an. Damit wird tatsächlich das komplementäre Beschränktheitsproblem gelöst, d.h. es wird 0 bei Beschränktheit und 1 bei Unbeschränktheit berechnet. Da die Funktion $x \mapsto 1 - x$ jedoch $\mathbb{R}$-berechenbar und stetig ist, kann sie nach Lemma 2.2.8 dahinter ausgeführt werden, so daß auch das ursprüngliche Beschränktheitsproblem gelöst wird. □

Durch ein etwas verfeinertes Vorgehen gelingt es sogar zu zeigen, daß der größte Häufungswert einer Folge sogar dann nicht analytisch berechenbar ist, wenn die Folge nicht analytisch berechenbar ist, sondern nur endlich $\mathbb{R}$-berechenar.

Satz 2.2.17. *Der größte Häufungswert einer beschränkten $\mathbb{R}$-berechenbaren Folge ist im allgemeinen nicht analytisch berechenbar. Genauer: Es gibt keine Maschine, die bei Eingabe einer $\mathbb{R}$-Maschine, die eine Folge $(a_n)_{n\in\mathbb{N}}$ berechnet, den Grenzwert $\limsup_{n\to\infty} a_n$ berechnet.*

Beweis. Das Problem wird wieder auf das Beschränktheitsproblem zurückgeführt. Sei $\mathcal{L}$ eine Maschine, die bei Eingabe einer $\mathbb{R}$-Maschine $\mathcal{F}$ (die eine Folge berechnet) den größten Häufungswert der durch $\mathcal{F}$ berechneten Folge analytisch berechnet.
Sei nun $\mathcal{M}$ eine Maschine, für die das Beschränktheitsproblem gelöst werden soll. Wir konstruieren wieder eine Maschine $\mathcal{M}_B$, auf die $\mathcal{L}$ zur Lösung des Beschränktheitsproblems für $\mathcal{M}$ angesetzt wird. Im Gegensatz zum vorherigen Beweis können wir nun nicht mehr $\mathcal{M}$ so lange simulieren, bis ein Wert k überschritten wird, da im Falle der Beschränktheit der Maschine diese Rechnung nicht enden würde. Die Eingabe für $\mathcal{L}$ muß hier aber eine *endliche* $\mathbb{R}$-Maschine $\mathcal{M}_B$ sein. Diese arbeitet wie folgt: Es sei $j \in \mathbb{N}$ die Eingabe von $\mathcal{M}_B$. Es sei weiter P eine Turing-berechenbare bijektive Aufzählung des Gitters $P : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ mit Umkehrfunktionen L und R, so daß gilt: $P(j) = (L(j), R(j))$, $L(P(n,m)) = n$ und $R(P(n,m)) = m$. In Lemma 4.1.5 wird ein Beispiel für solche Funktionen gegeben. $\mathcal{M}_B$ überprüft nun für die Eingabe j, ob $\mathcal{M}$ innerhalb der ersten $L(j)$ Schritte den Wert $R(j)$ übersteigt. Zusätzlich überprüft sie auch, ob schon vorher dieser Wert übertroffen wird, es wird also für alle $1 \leq m < L(j)$ überprüft, ob $\mathcal{M}^{(m)} > R(j)$ gilt. Falls nun gilt: $\mathcal{M}^{(m)} \leq R(j)$ für alle $1 \leq m < L(j)$, aber $\mathcal{M}^{L(j)} > R(j)$, dann wird eine 1 ausgegeben, in allen anderen Fällen eine 0. In beiden Fällen terminiert $\mathcal{M}_B$ mit ihrer Ausgabe. Auf diese Weise wird sichergestellt, daß für jede Schranke k nur ein *einziges Mal* eine 1 ausgegeben wird, wenn diese überschritten wird. Nun gilt:
Wenn $\mathcal{M}$ beschränkt ist, dann gilt $\mathcal{M}_B(j) = 1$ nur für endlich viele j, denn es werden nur endlich viele Schranken k überschritten, und für jede überschrittene Schranke wird für genau ein j (nämlich jenes mit dem kleinsten $L(j)$) eine 1 ausgegeben. Ist umgekehrt $\mathcal{M}$ unbeschränkt, so wird für unendlich viele j der berechnete Wert von $\mathcal{F}(j)$ gleich 1 sein, da für jedes k ein n existiert mit $\mathcal{M}^{(m)} > k$. Da P bijektiv ist, gibt es ein j mit $P(j) = (m,k)$. Ist m minimal mit $\mathcal{M}^{(m)} > k$, dann gilt nach Konstruktion $\mathcal{M}_B(j) = 1$. Es gilt also:

$$\limsup_{j\to\infty} \mathcal{M}_B(j) = \begin{cases} 1 & : \quad \mathcal{M} \text{ unbeschränkt} \\ 0 & : \quad \text{sonst} \end{cases}$$

Nun wird also wie oben die Maschine $\mathcal{L}$ auf $\mathcal{M}_B$ angesetzt. Schließlich können mit demselben Argument wie im letzten Beweis beim Ergebnis 0 und 1 noch vertauscht werden. □

Damit ist es auch nicht möglich, unter der Voraussetzung, daß nur endlich viele endliche Häufungswerte existieren, diese auch in irgendeiner Ordnung zu berechnen. Ansonsten könnte man mit Lemma 2.2.8 nach der (analytischen) Berechnung der Häufungswerte die stetige $\mathbb{R}$-berechenbare Funktion $\max : \mathbb{R}^* \to \mathbb{R}$ ausführen und somit den größten Häufungswert berechnen.

2.2.2 Berechnungsgraph und Berechnungsbaum

Das Ziel dieses Abschnittes ist es, den Darstellungssatz für endlich $\mathcal{R}$-berechenbare Funktionen herzuleiten. Dieser charakterisiert die $\mathcal{R}$-berechenbaren Funktionen vollständig. In [BCSS98] und [HVS95] finden sich ebenfalls Beweise dieses Satzes. Wir wollen ihn hier herleiten, da die Herleitung ein klares Bild auf die Struktur von Berechnungen wirft und wir für die Verallgemeinerung des Darstellungssatzes auf analytische Maschinen einige hier verwendete Begriffe benötigen werden.

Eine Berechnung einer Maschine ist eine Folge von Konfigurationen. Abstrahiert man von der konkreten Konfiguration und unterscheidet Konfigurationen nur durch den Inhalt des Programmzählers, so ergibt sich ein endlicher Graph, der *Berechnungsgraph* der Maschine.
In [BCSS98] werden Maschinen durch den Berechnungsgraphen definiert. Bei endlichen Berechnungen sind beide Modelle äquivalent. Wir haben die Definition über ein Maschinenmodell vorgezogen, weil es größere Allgemeinheit gestattet und insbesondere die verschiedenen Klassen berechenbarer Funktionen leichter zugänglich sind.
Der Berechnungsgraph liefert die grundsätzliche Struktur einer Maschine bzw. eines Programms. Verfolgt man einzelne Berechnungen einer Maschine im Berechnungsgraphen und "rollt dabei den Graphen zu einem Baum ab", so erhält man den *Berechnungsbaum*. Anhand des Berechnungsbaumes kann man einzelne Berechnungen anschaulich verfolgen und die Operationen der Maschine leicht erkennen.

Definition 2.2.18. *Sei $\mathcal{M}$ eine $\mathcal{R}$-Maschine mit Programm $\pi = (\pi_1, \ldots, \pi_N)$. Der* Berechnungsgraph $\mathcal{G}(\mathcal{M})$ *ist ein gerichteter Graph mit Knotenmenge $\{\pi_1, \ldots, \pi_N\}$. Die Menge der Kanten ist gegeben durch*

$$\{\pi_i \to \pi_{i+1} \,|\, 1 \leq i < N,\ \pi_i \text{ kein Verzweigungsbefehl }\}$$
$$\{\pi_i \to \pi_m, \pi_i \to \pi_n \,|\, \pi_i = \texttt{if } \alpha > 0 \texttt{ then goto } m \texttt{ else goto } n\}$$

Bei Verzweigungsbefehlen sind die Kanten mit $\alpha > 0$ bzw. $\alpha \leq 0$ beschriftet, analog mit $=, \neq$ bei nichtgeordneten Ringen.

Der Berechnungsbaum von $\mathcal{M}$ ist nun der zu $\mathcal{G}(\mathcal{M})$ gehörige Baum:

Definition 2.2.19. *Sei $\mathcal{M}$ eine $\mathcal{R}$-Maschine mit Programm $\pi = (\pi_1, \ldots, \pi_N)$. Der* Berechnungsbaum $\mathcal{B}(\mathcal{M})$ *ist ein gerichteter beschrifteter Graph mit Knotenmenge $V \subseteq \{\pi_1, \ldots, \pi_N\}^*$. $\mathcal{B}(\mathcal{M})$ ist induktiv wie folgt definiert:*

- *Die Wurzel von $\mathcal{B}(\mathcal{M})$ ist (π_1).*
- *Ist $(\pi_{i_1}, \ldots, \pi_{i_k})$ ein Knoten von $\mathcal{B}(\mathcal{M})$ mit $\pi_{i_k} = \texttt{end}$, so hat er keine Kinder.*
- *Ist $(\pi_{i_1}, \ldots, \pi_{i_k})$ ein Knoten von $\mathcal{B}(\mathcal{M})$ mit $\pi_{i_k} \neq \texttt{end}$, so hat er Kinder*

$$\begin{array}{ll} (\pi_{i_1}, \ldots \pi_{i_k}, \pi_{i_k+1}) & \text{falls } \pi_i \text{ kein Verzweigungsbefehl} \\ \left.\begin{array}{l}(\pi_{i_1}, \ldots \pi_{i_k}, \pi_n) \\ (\pi_{i_1}, \ldots \pi_{i_k}, \pi_m)\end{array}\right\} & \text{falls } \pi_i = \texttt{if } \alpha > 0 \texttt{ then goto } m \texttt{ else goto } n \end{array}$$

Der Berechnungsbaum einer Maschine ist ein nützliches Hilfsmittel, will man die Berechnungen einer Maschine untersuchen. Genauer betrachtet man verschiedene *Berechnungspfade* in diesem Baum. Ein Pfad ist eine Folge von Knoten im Berechnungsbaum, $\sigma = (\sigma_i)_{i \in \mathbb{N}}$.
Wir ordnen jeder Berechnung der Maschine $\mathcal{M}$ einen Berechnungspfad im Berechnungsbaum zu:

Definition 2.2.20. *Sei $\mathcal{M}$ eine $\mathcal{R}$-Maschine und $(k_i = (\alpha_i, \beta_i, \gamma_i, \pi, x^{(i)}, y^{(i)}, z^{(i)}))_{i\in\mathbb{N}}$ die Berechnung von $\mathcal{M}$ angesetzt auf x. Dann ist der* Berechnungspfad *$\sigma(x) = (\sigma_1(x), \sigma_2(x), \ldots) \in (\{\pi_1, \ldots, \pi_n\}^*)^{\mathbb{N}}$ von x komponentenweise definiert durch*

$$\sigma_i(x) := (\pi_{\beta_1}, \ldots, \pi_{\beta_i}),\ i \in \mathbb{N}$$

Wir definieren weiter den Berechnungspfad von x bis zum Zeitpunkt i *als*

$$\sigma_{\leq i}(x) = (\sigma_1(x), \ldots, \sigma_i(x)).$$

Anmerkung. Nach dieser Definition sind alle Berechnungspfade unendlich lang. Bei endlichen, haltenden Berechnungen gilt ab einem Berechnungszeitpunkt i: $k_{i+1} = k_i$ (sobald $\pi_{\beta_i} =$ `end`). Deshalb können wir Pfade, die haltenden Berechnungen entsprechen, auch als ab einem Index i endliche Pfade der Länge i (wenn i minimal) auffassen.

Der i-te Knoten von $\sigma(x)$ ist also definiert als der Knoten, der die Befehlsfolge von $\mathcal{M}$ bis zum Berechnungszeitpunkt i widerspiegelt.
Anhand des Berechnungspfades kann man die Berechnungen, die für eine Eingabe x von der Maschine durchgeführt werden, leicht verfolgen. An jedem Knoten werden Zuweisungen, arithmetische Berechnungen oder Verzweigungen ausgeführt. An jedem Knoten werden also elementare Funktionen (Identität, Addition, Multiplikation und jeweilige Inverse) ausgeführt, und die Inhalte der Register ergeben sich durch Hintereinanderausführung dieser elementaren Funktionen. Dies bedeutet aber, daß die Inhalte der Register rationale Funktionen der Eingabe x bilden. Präziser geht man wie folgt vor:
Für jede Speicherzelle z_i, jede Ausgabezelle y_i und die Register α und γ werden nun die an einem Knoten des Berechnungsbaums *berechneten Funktionen* auf natürliche Weise definiert:

- Sei $v = (\pi_1)$ die Wurzel von $\mathcal{B}(\mathcal{M})$. Dann ist

$$\alpha(v, x) = \gamma(v, x) = z_i(v, x) = y_i(v, x) = 0.$$

- Seien die Funktionen für $v = (\pi_1, \ldots, \pi_v)$ schon definiert und $w = (\pi_1, \ldots, \pi_v, \pi_w)$ ein Nachfolger von v. Dann setzen wir stets $\alpha(w, x) = \alpha(v, x)$, $\gamma(w, x) = \gamma(v, x)$, $z_i(w, x) = z_i(v, x)$, $y_i(w, x) = y_i(v, x)$, außer in den folgenden Fällen:

$$\begin{aligned}
\alpha(w, x) &= x_i && \text{falls} && \pi_w = \alpha := x_i \\
\alpha(w, x) &= z_i(v, x) && \text{falls} && \pi_w = \alpha := z_i \\
\alpha(w, x) &= c && \text{falls} && \pi_w = \alpha := c \\
y_i(w, x) &= \alpha(v, x) && \text{falls} && \pi_w = y_i := \alpha \\
z_i(w, x) &= \alpha(v, x) && \text{falls} && \pi_w = z_i := \alpha \\
\alpha(w, x) &= \alpha(v, x) + z_i(v, x) && \text{falls} && \pi_w = \alpha := \alpha + z_i \\
\alpha(w, x) &= \alpha(v, x) \cdot z_i(v, x) && \text{falls} && \pi_w = \alpha := \alpha \cdot z_i \\
\alpha(w, x) &= -\alpha(v, x) && \text{falls} && \pi_w = \alpha := -\alpha \\
\alpha(w, x) &= \alpha(v, x)^{-1} && \text{falls} && \pi_w = \alpha := \alpha^{-1}
\end{aligned}$$

Man sieht sofort: $\alpha(v,\cdot)$, $z_i(v,\cdot)$, $y_i(v,\cdot)$ sind Polynome bzw. im Falle, daß $\mathcal{R}$ Divisionsbereich ist, rationale Funktionen. Insbesondere sind an allen Verzweigungsstellen im Programm die Bedingungen `if` $\alpha > 0$ `then` ... semi-algebraische Bedingungen, das heißt, α ist ein Polynom bzw. eine rationale Funktion der Eingabe x. Weiter sieht man, daß die berechneten Funktionen nur vom Pfad im Berechnungsbaum abhängig sind. Haben zwei Eingaben x_1 und x_2 also den gleichen Berechnungspfad, so sind die berechneten rationalen Funktionen dieselben, aber mit verschiedenen Argumenten.

Definition 2.2.21. *Sei $\mathcal{M}$ eine $\mathcal{R}$-Maschine und $\sigma \in \mathcal{B}(\mathcal{M})$ ein Pfad im Berechnungsbaum. Der* Einzugsbereich von σ *ist*

$$D_\sigma = \{x \in D_\mathcal{M} | \sigma(x) = \sigma\}.$$

Alle $x \in D_\sigma$ haben den gleichen Berechnungspfad, und für alle diese x wird in den Registern, im Speicher und auf dem Ausgabeband das gleiche Polynom bzw. die gleiche rationale Funktion berechnet. Die Maschine verhält sich also auf all diesen Eingaben gleich. Die Zugehörigkeit einer Eingabe x zu einer Menge D_σ ist durch die Folge semi-algebraischer Bedingungen `if` $\alpha > 0$ `then` ... im Berechnungspfad bestimmt. Ist dies eine endliche Folge, so ist D_σ also eine semi-algebraische Menge. Dabei verstehen wir unter einer *semi-algebraischen Menge* eine Menge, die durch endlich viele polynomiale Gleichungen und Ungleichungen der Form $p(\xi_1,\ldots,\xi_n) = 0$, $p(\xi_1,\ldots,\xi_n) \neq 0$ und $p(\xi_1,\ldots,\xi_n) > 0$ definiert werden kann.
Betrachten wir nun endliche Berechnungen: Sei $H_\mathcal{M}$ der Haltebereich von $\mathcal{M}$. Dann ist jeder Pfad $\sigma(x)$ für $x \in H_\mathcal{M}$ von endlicher Länge. Es kann aber nur abzählbar viele Pfade endlicher Länge geben, wie man leicht einsieht, wenn man die Pfade ihrer Länge nach aufzählt und berücksichtigt, daß es höchstens 2^n Pfade der Länge n geben kann, da ein Knoten höchstens zwei Kinder haben kann. Also ist die Menge

$$\{D_{\sigma(x)} | x \in H_\mathcal{M}\}$$

eine abzählbare Menge semi-algebraischer Mengen. Damit haben wir gezeigt:

Satz 2.2.22 (Darstellungssatz für $\mathcal{R}$-Maschinen)**.** *Sei $\mathcal{M}$ eine $\mathcal{R}$-Maschine, und $\Phi : H_\mathcal{M} \to \mathcal{R}$ die von $\mathcal{M}$ berechnete Funktion. Dann ist der Haltebereich $H_\mathcal{M}$ eine abzählbare Vereinigung semialgebraischer Mengen, auf denen die Funktion Φ ein Polynom bzw. im Falle daß $\mathcal{R}$ ein Divisionsbereich ist, eine rationale Funktion ist.*

Der Darstellungssatz wurde bereits in [BCSS98] und [Vie96] bewiesen. Er klärt die Struktur von $\mathcal{R}$-berechenbaren (also endlich berechenbaren) Funktionen auf. Ein wesentliches Ziel unserer Arbeit ist es zu untersuchen, ob Verallgemeinerungen des Darstellungssatzes für analytische Maschinen gelten; insbesondere, ob auf durch Programmpfade festgelegten Teilmengen des Definitionsbereichs analytisch berechenbare Funktionen durch Potenzreihen darstellbar sind.
Wir geben hier noch eine Verschärfung des Darstellungssatzes an, die bekannt ist, in der Literatur aber selten erwähnt wird. Die Koeffizienten der Polynome bzw. rationalen Funktionen, die entlang eines Pfades σ im Akkumulator, im Speicher und auf dem Ausgabeband berechnet werden, sind sogar selbst wieder berechenbar. Wir formulieren dies exemplarisch für die Ausgabe.

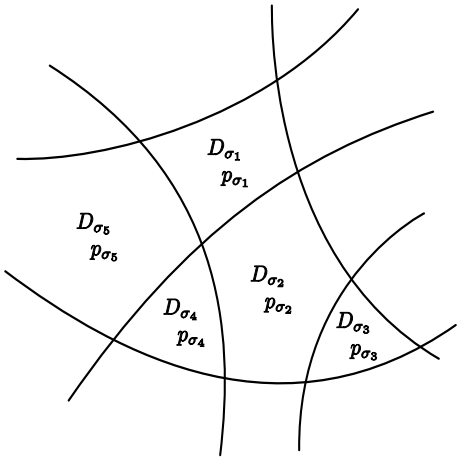


Abbildung 2.2: Darstellungssatz für $\mathcal{R}$-berechenbare Funktionen

Satz 2.2.23 (Konstruktiver Darstellungssatz)**.** *Die Darstellung der von $\mathcal{R}$-Maschinen berechneten Funktionen durch Polynome bzw. rationale Funktionen ist $\mathcal{R}$-berechenbar. Das heißt, es gibt eine $\mathcal{R}$-Maschine $\mathcal{P}$, die bei Eingabe einer $\mathcal{R}$-Maschine $\mathcal{M}$ und eines Eingabewertes $x \in H_{\mathcal{M}}$ die Koeffizienten der durch $\mathcal{M}$ berechneten rationalen Funktion entlang des Pfades $\sigma(x)$ berechnet.*

Beweis. Der Beweises von Satz 2.2.22 liefert im wesentlichen auch eine Konstruktion der entlang eines Pfades in den Registern und auf den Bändern berechneten Funktionen: Sei $\mathcal{M}$ eine Eingabemaschine und $x \in H_{\mathcal{M}}$ eine Eingabe. Sei $\sigma = \sigma(x)$ (für $\mathcal{M}$) und $r_\sigma = \frac{p_\sigma}{q_\sigma}$ die von $\mathcal{M}$ berechnete rationale Funktion entlang σ mit Zähler- und Nennerpolynom. Die Maschine $\mathcal{P}$ berechnet nun die Koeffizienten von p und q durch Simulation von $\mathcal{M}$. Dabei werden für den Akkumulator und für jede von $\mathcal{M}$ benutzte Speicher- und Ausgabebandzelle die Koeffizienten der dort berechneten rationalen Funktionen mitberechnet. □

Wir beweisen nun einen Hilfssatz, der zeigt, daß eine über den reellen oder komplexen Zahlen berechenbare Funktion auch stets auf mindestens einer nichtleeren offenen Menge durch eine rationale Funktion dargestellt wird.

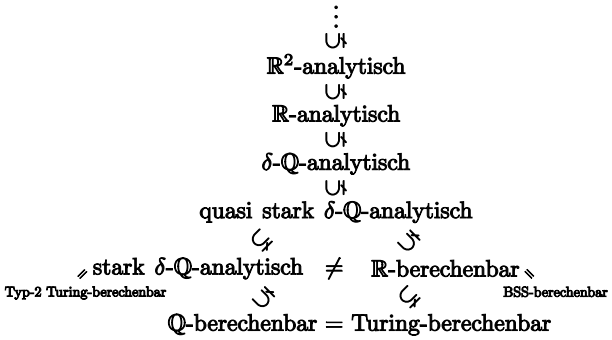
Lemma 2.2.24. *Sei eine offene Menge D gegeben, die sich als Vereinigung abzählbar vieler semi-algebraischer Mengen schreiben läßt:*

$$D = \bigcup_{i \in \mathbb{N}} D_i.$$

Dann gibt es wenigstens ein i, so daß D_i nichtleeres Inneres hat.

Beweis. Sei also D offen, D_i semi-algebraisch für $i \in \mathbb{N}$ und $D = \bigcup_{i \in \mathbb{N}} D_i$. Dann gibt es nach dem Kategoriensatz von Baire wenigstens ein i, so daß $\overline{D_i}$ nichtleeres Inneres hat. Da D_i semi-algebraisch ist, muß also auch D_i selbst nichtleeres Inneres haben. □

Tabelle 2.2 *Hierarchie der Klassen berechenbarer Funktionen*



2.2.3 Einordnung der Funktionsklassen und Beispiele

In diesem Abschnitt geben wir Beispiele für die verschiedenen Klassen berechenbarer Funktionen und gehen insbesondere auf die in der Einleitung genannten Beispiele ein. Einen Überblick über die Hierarchie der berechenbaren Funktionen gibt Tabelle 2.2. Ein Großteil der Resultate dieses Abschnittes finden sich bereits in [Vie96] und [Cha98]. Wir gehen auf sie ein, da sie einen guten Überblick über die Klassen berechenbarer Funktionen verschaffen.

$\mathbb{R}$-berechenbare Funktionen

Die einfachste betrachtete Klasse berechenbarer Funktionen bilden die $\mathbb{Q}$-berechenbaren Funktionen, die (bei geeigneter Definition rationaler Berechnungen) äquivalent zu den Turing-berechenbaren Funktionen sind. Die erste echt größere Klasse ist die Klasse der $\mathbb{R}$-berechenbaren Funktionen. Sie ist auf natürliche Weise echt größer, da der Definitionsbereich der Funktionen gleich der Menge der reellen Zahlen ist, und jede Turing-Maschine auf einer $\mathbb{R}$-Maschine simuliert werden kann.

Beispiel 2.2.25.

1. *Konstante Funktionen* $f_c : \mathbb{R} \to \mathbb{R},\ x \mapsto c$ sind genau dann $\mathbb{R}$-berechenbar, wenn $c \in \mathbb{Q}$. Natürlich ist jede konstante Funktion f_c für $c \in \mathbb{R}$ auch $\mathbb{R}[c]$-berechenbar.

2. *Polynomfunktionen* $p(x) = a_n x^n + \ldots + a_0$ sind genau dann $\mathbb{R}$-berechenbar, wenn für die Koeffizienten gilt: $a_i \in \mathbb{Q}$. Analog zu den konstanten Funktionen sind Polynome stets $\mathbb{R}[a_0, \ldots, a_n]$-berechenbar.

3. Die *Heaviside-Funktion*

$$h : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 0 & : \quad x \leq 0 \\ 1 & : \quad x > 0 \end{cases} .$$

   ist $\mathbb{R}$-berechenbar, da exakte Vergleiche mit reellen Zahlen verfügbar sind.

Die $\mathbb{R}$-berechenbaren Funktionen werden durch den Darstellungssatz 2.2.22 charakterisiert. Mit Hilfe dieses Satzes lassen sich leicht Beispiele nicht $\mathbb{R}$-berechenbarer Funktionen angeben. Jede nichtrationale analytische Funktion etwa ist nicht $\mathbb{R}$-berechenbar, etwa die Exponentialfunktion $\exp : \mathbb{R} \to \mathbb{R}, x \mapsto e^x$ oder die Wurzelfunktion $\sqrt{\cdot} : \mathbb{R}^+ \to \mathbb{R}, x \mapsto \sqrt{x}$. Denn nach dem Identitätssatz für analytische Funktionen gibt es keine nichtleere offene Menge, auf der diese Funktionen mit einer rationalen Funktion übereinstimmen. Nach dem Darstellungssatz gibt es aber für eine $\mathbb{R}$-berechenbare Funktion semi-algebraische Mengen A_i, $i \in \mathbb{N}$, auf denen die Funktion rational ist und für die gilt

$$\mathbb{R} = \bigcup_{i \in \mathbb{N}} A_i.$$

Da die semi-algebraischen Mengen über $\mathbb{R}$ gerade Vereinigungen (halboffener) Intervalle sind, muß wenigstens eines der A_i nichtleeres Inneres haben, da eine abzählbare Vereinigung einzelner Punkte nicht das Kontinuum ergeben kann.

Stark δ-$\mathbb{Q}$-analytisch berechenbare Funktionen

Die stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen sind die einzige Klasse, die nicht in der aufsteigenden Hierarchie der hier betrachteten Funktionsklassen einzuordnen sind. Das liegt daran, daß nicht alle $\mathbb{R}$-berechenbaren Funktionen auch stark δ-$\mathbb{Q}$-analytisch sind. Die Genauigkeitsschranke erzwingt zusammen mit der Rundung die Stetigkeit der Funktion:

Satz 2.2.26 ([Vie96])**.** *Die stark δ-$\mathbb{Q}$-analytischen Funktionen sind stetig.*

Beweis. Sei f stark δ-$\mathbb{Q}$-analytisch mit Maschine $\mathcal{M}_f$. Wir nehmen an, es gibt eine Folge $(x_n)_{n \in \mathbb{N}}$ mit $x_n \to x$, aber $f(x_n) \not\to f(x)$. Damit gibt es ein $\varepsilon > 0$, so daß für unendlich viele n gilt: $|f(x_n) - f(x)| > \varepsilon$. Es sei n ein Berechnungszeitpunkt, zu dem die Genauigkeit der Ausgabe kleiner als $\varepsilon/2$ ist. Wir wählen eine Rundung ρ, die bei Genauigkeit k die reelle Achse äquidistant in Intervalle der Länge 2^{-n} unterteilt. Ist δ_n nun die Rundungsgenauigkeit zum Zeitpunkt n, dann gibt es ein i, so daß $\rho(n, x_i) = \rho(n, x)$. Damit verhält sich die Maschine $\mathcal{M}_f$ auf den Eingaben x und x_i bis zum Zeitpunkt n identisch. Die Genauigkeitsangabe für die beiden Eingaben ist also ebenfalls identisch, sowie die n-te Approximation. Dann können sich die Funktionswerte aber nicht um ε unterscheiden. □

Die stark δ-$\mathbb{Q}$-analytischen Funktionen entsprechen den Typ-2 Turing-berechenbaren Funktionen [Wei00]. Sie sind in jeder der folgenden Klassen enthalten.
Im Gegensatz zu den $\mathbb{R}$-berechenbaren Funktionen umfassen die stark δ-$\mathbb{Q}$-analytischen Funktionen auch nichtrationale algebraische Funktionen und transzendente analytische Funktionen, wie die Exponentialfunktion. Zur Berechenbarkeit analytischer Funktionen verweisen wir auf Kapitel 3. Andererseits sind elementare nichtstetige Funktionen wie die Heaviside-Funktion nicht stark δ-$\mathbb{Q}$-analytisch.

Quasi-stark δ-$\mathbb{Q}$-analytisch berechenbare Funktionen

Die quasi-stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen unterscheiden sich von den stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen dadurch, daß die Genauigkeitsschranke

der Ausgabe nicht immer gilt, sondern nur *fast immer*, also von einem gewissen Berechnungszeitpunkt an. Diese relaxierte Bedingung gestattet es, auch nicht stetige Funktionen zu berechnen. Die Idee besteht darin, daß $\mathbb{R}$-Maschinen auf rationalen Zahlen mittels mit Intervallarithmetik von immer größer werdender Präzision simuliert werden. Dies funktioniert prinzipiell auch mit starken δ-$\mathbb{Q}$-Maschinen, bei Verzweigungsbefehlen kann es jedoch passieren, daß die falsche Verzweigungsrichtung gewählt wird, wenn das Vertrauensintervall noch zu groß ist. Bei hinreichender Genauigkeit jedoch kann bei endlichen Berechnungen immer sichergestellt werden, daß die richtige Auswahl getroffen wird. Wir zitieren hier ohne Beweis den *Simulationssatz* von Chadzelek, der zeigt, daß jede $\mathbb{R}$-berechenbare Funktion auch quasi-stark δ-$\mathbb{Q}$-analytisch ist.

Satz 2.2.27 (Simulationssatz für $\mathbb{R}$-Maschinen, [Cha98])**.** *Jede $\mathbb{R}$-berechenbare Funktion ist quasi-stark δ-$\mathbb{Q}$-analytisch.*

Beispiel 2.2.28. Statt des detaillierten Beweises zeigen wir kurz, wie man die Heaviside-Funktion quasi-stark δ-$\mathbb{Q}$-analytisch berechnen kann, um dem Leser ein Gefühl für quasi-starke Berechnungen zu geben. Die Berechnung geschieht in Phasen, die durch die Rundungsgenauigkeit δ bestimmt sind. Die Eingabe x erhält die Maschine in gerundeter Form $\tilde{x} = \rho(\delta, x)$. Ist nun $0 \in (\tilde{x}-\delta, \tilde{x}+\delta)$, so wird eine 0 als Funktionsergebnis und eine 0 als Ausgabegenauigkeit ausgegeben und die Rundungsgenauigkeit erhöht. Ist umgekehrt $0 \notin (\tilde{x}-\delta, \tilde{x}+\delta)$, so wird je nach dem Vorzeichen von $\tilde{x}$ 0 oder 1 als Ergebnis und 0 als Ausgabegenauigkeit ausgegeben und in einen Endzustand übergegangen. Ist nun $x \neq 0$, so wird bei hinreichend hoher Rundungsgenauigkeit gelten $0 \notin (\tilde{x}-\delta, \tilde{x}+\delta)$, und die Berechnung wird mit dem richtigen Ergebnis halten. Die Ausgabegenauigkeit ist nur falsch, solange die 0 vom Vertrauensintervall überlappt wird, also nur endlich oft. Im Falle $x = 0$ wird immer gelten $0 \in (\tilde{x}-\delta, \tilde{x}+\delta)$ und somit die Ausgabe gegen das richtige Ergebnis 0 "konvergieren". Die Ausgabegenauigkeit ist in diesem Fall immer korrekt. In beiden Fällen konvergiert sie gegen 0, da sie stets 0 ist.

Die quasi-stark δ-$\mathbb{Q}$-analytischen Funktionen bilden die größte und letzte Klasse in dieser Hierarchie, die noch unter Komposition abgeschlossen ist.

Robust δ-$\mathbb{Q}$-analytisch berechenbare Funktionen

Bei den robust δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen muß keine Genauigkeitsschranke mit ausgegeben werden. Analog wie bei $\mathbb{R}$-analytisch berechenbaren Funktionen zeigt man (vgl. auch [Cha98]), daß diese nicht unter Komposition abgeschlossen sind. Damit ist schon gezeigt, daß sie eine echte Oberklasse der quasi-stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen bilden.

Beispiel 2.2.29. Ein Beispiel einer Funktion, die nicht robust δ-$\mathbb{Q}$-analytisch berechenbar ist, ist die Funktion

$$\chi_{\mathbb{Q}} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 0 & : \quad x \in \mathbb{R} - \mathbb{Q} \\ 1 & : \quad x \in \mathbb{Q} \end{cases}$$

Ein detaillierter Beweis dieser Tatsache findet sich in [Vie96]. Die Idee besteht darin, die feste Rundung $\rho(x, \delta) = \frac{[x \cdot 2^\delta]}{2^\delta}$ zu verwenden, und eine Folge rationaler Zahlen x_n zu konstruieren, bei der die Dualbruchdarstellung von x_n ein Präfix der Darstellung von

x_{n+1} ist. Für jede der Zahlen x_n muß die Ausgabe einer die Funktion berechnenden Maschine gegen 1 konvergieren, für den Grenzwert $\lim_{n\to\infty} x_n =: x$ jedoch gegen 0, woraus ein Widerspruch entsteht, da die Berechnung der Maschine wegen der Rundung für sukzessive x_n identisch bis zur letzten Stelle von x_n ist.

Analytisch $\mathbb{R}$-berechenbare Funktionen

Die analytisch $\mathbb{R}$-berechenbaren Funktionen bilden die größte Klasse in der Hierarchie, die noch durch eine einzige Maschine berechnet werden können. Diese Klasse umfaßt alle vorangegangenen Klassen echt.

Beispiel 2.2.30.

1. Wir werden sehen, daß jede Funktion, die durch eine Potenzreihe definiert wird und deren Koeffizientenenfolge Turing-berechenbar oder auch $\mathbb{C}$-berechenbar ist, analytisch $\mathbb{C}$-berechenbar ist.

2. Als Beispiel einer analytisch $\mathbb{R}$-berechenbaren Funktion, die nicht robust δ-$\mathbb{Q}$-analytisch berechenbar ist, dient $\chi_{\mathbb{Q}}$. Diese Funktion wird analytisch berechnet, indem die rationalen Zahlen aufgezählt und mit der Eingabe verglichen werden. Bei Übereinstimmung wird eine 1 ausgegeben und in einen Endzustand übergegangen, andernfalls 0 ausgegeben und die Aufzählung fortgesetzt.

Weitere Beispiele folgen mit den analytisch berechenbaren analytischen Funktionen in Kapitel 3. Dort wird auch gezeigt, inwiefern zusätzliche Bedingungen wie die Holomorphie einer Funktion zum Zusammenbruch der Hierarchie führen.

Typ-2 Turing-Maschinen

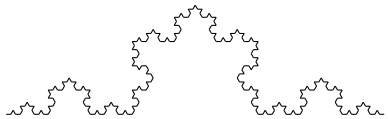
Wir schließen den Abschnitt mit einem Vergleich der analytischen Maschinen mit dem Maschinenmodell, das in der Typ-2 Theorie der Berechenbarkeit verwendet wird. Dabei skizzieren wir nur ein vereinfaches Modell, auf Repräsentationen, mit deren Hilfe die Expressivität der Theorie vergrößert wird, gehen hier nicht ein. Für eine ausführliche Darstellung vgl. etwa [Wei95].
Das grundlegende Modell der Theorie ist ein Turing-Maschinenmodell mit einseitig unendlichen Bändern über $\{0,1\}$. Es gibt mehrere Eingabebänder, mehrere Arbeitsbänder und ein Ausgabeband. In der Typ-2 Theorie wird gefordert, daß Ein- und Ausgabebänder *Einwegbänder* sind, d.h. der Kopf der Turing-Maschine kann sich nur von links nach rechts bewegen. Mit Funktionen in : $\mathbb{R}^k \to \{0,1\}^{\mathbb{N}}$ bzw. out : $\{0,1\}^{\mathbb{N}} \to \mathbb{R}$ werden Ein- und Ausgabebänder interpretiert und dadurch Funktionen $\mathbb{R}^k \to \mathbb{R}$ definiert.
Die Einschränkung, daß das Ausgabeband ein Einwegband ist, führt dazu, daß die durch Typ-2 Turing-Maschinen berechenbaren Funktionen automatisch stetig sind. Ferner kann die Maschine in endlicher Zeit nur ein endliches Anfangsstück des Eingabebandes lesen, und der Eingabewert, mit der die Maschine auf ihren Arbeitsbändern rechnet, ist stets gerundet. Daher gilt:

Lemma 2.2.31 ([Vie96, Cha98])**.** *Die Menge der Typ-2 Turing-berechenbaren Funktionen ist gleich der Menge der (robust) stark δ-$\mathbb{Q}$-berechenbaren Funktionen.*

2.3 Darstellungssätze für analytische Maschinen

Abgesehen von der allgemeinen Diskussion zu Beginn des Abschnitts wenden wir uns wieder den Körpern $\mathbb{R}$ und $\mathbb{C}$ zu. Der Darstellungssatz für $\mathcal{R}$-berechenbare Funktionen 2.2.22 charakterisiert vollständig diese Klasse berechenbarer Funktionen. Die Erweiterung der $\mathcal{R}$-Maschinen durch unendliche konvergierende Berechnungen zu analytischen Maschinen wirft die Frage nach einer Verallgemeinerung des Darstellungssatzes auf, werden doch bei endlichen Teilberechnungen stets $\mathcal{R}$-berechenbare Funktionen erzeugt. Es stellt sich also die Frage, inwieweit die analytisch berechenbaren Funktionen durch Potenzreihen darstellbar sind. Dabei ist der Konvergenzbegriff der analytischen Maschinen wesentlich. Die Konvergenz der durch eine analytische Maschine berechneten Zwischenergebnisse gegen die von der Maschine berechnete Funktion ist eine *punktweise* Konvergenz. Würde man hier voraussetzen, daß die durch die n-te Approximation definierten Funktionen *gleichmäßig* gegen die Grenzfunktion konvergieren, so würde bei vorausgesetzter Analytizität der n-ten Approximationen sofort folgen, daß die Grenzfunktion ebenfalls analytisch, also durch Potenzreihen darstellbar, ist.
Bei $\mathcal{R}$-berechenbaren Funktionen ist jede einzelne haltende Berechnung von endlicher Länge; daher können nur endlich viele Vergleiche während einer solchen Berechnung stattfinden und der Definitionsbereich wird somit in abzählbar viele semi-algebraische Mengen unterteilt. Vergleichbares gilt bei unendlichen Berechnungen nicht mehr. Es kann sogar passieren, daß jede Eingabe einen eigenen Berechnungspfad hat, daß also alle Mengen D_σ einpunktig sind. In diesem Fall macht eine Darstellung durch Potenzreihen natürlich keinen Sinn, da eine nichttriviale Konvergenz einer Reihe immer nur auf einer offenen Menge stattfinden kann.
Auch die wesentlich schwächere Eigenschaft der Stetigkeit gilt im allgemeinen nicht für analytisch berechenbare Funktionen. Das einfache Beispiel $\chi_{\mathbb{Q}}$ zeigt sofort, daß eine $\mathbb{R}$-analytisch berechenbare Funktion nirgends stetig sein muß.

Beispiel 2.3.1. Die Unterteilung der Definitionsbereiche der analytisch berechenbaren Funktionen durch die Berechnungspfade ist im allgemeinen überabzählbar: Dazu sei $K : [0, 1] \to \mathbb{R}$ die Funktion, deren Graph die Kochsche Kurve darstellt:



K ist $\mathbb{R}$-analytisch berechenbar. Tatsächlich ist K sogar stark δ-$\mathbb{Q}$-analytisch berechenbar. Die Genauigkeitsschranke bestimmt man, indem man durch Triangulierung den Bereich eingrenzt, in dem $K(x)$ liegen kann. Eine genauere Darstellung findet sich in [Cha98].
K ist ein Beispiel für eine überall stetige, aber nirgends differenzierbare Funktion. Es ist also nicht möglich, daß K auf irgendeiner offenen Teilmenge ihres Definitionsbereiches durch eine Potenzreihe dargestellt werden kann.

Dieses Beispiel zeigt, daß es sogar stark δ-$\mathbb{Q}$ analytisch berechenbare Funktionen gibt, die nirgends glatt sein müssen, also sicher auch nicht durch Potenzreihen darstellbar

sind. Man muß sich also bei der Untersuchung auf die Fälle beschränken, bei denen offene nichtleere D_σ existieren. Das ist beispielsweise der Fall, wenn bei jeder Berechnung nur endlich viele Vergleiche auftreten.
Auch in diesem Fall gibt es im allgemeinen keine Darstellung durch Potenzreihen auf dem gesamten Innern der D_σ-Mengen. Es macht hier einen großen Unterschied, ob man auf dem Körper der reellen Zahlen $\mathbb{R}$ oder dem der komplexen Zahlen $\mathbb{C}$ arbeitet. Während auf $\mathbb{R}$ selbst auf offenen D_σ noch große Unregelmäßigkeiten entstehen können, sind die arithmetischen Operationen auf $\mathbb{C}$ stark genug, um zumindest in großen Teilen Darstellung durch Potenzreihen im Innern von D_σ-Mengen zu erzwingen.

2.3.1 Darstellungen über $\mathbb{R}$

In diesem Abschnitt betrachten wir Maschinen über dem Körper der reellen Zahlen $\mathbb{R}$. Wir werden zunächst zeigen, daß für allgemeine $\mathbb{R}$-analytische Maschinen selbst im Innern offener D_σ-Mengen eine Darstellung durch Potenzreihen nicht gegeben ist. Tatsächlich werden wir eine Maschine angeben, die ohne Vergleiche arbeitet und dennoch eine nicht stetige Funktion berechnet.
Durch Einschränkung des Maschinenmodells läßt sich mehr erreichen; bekannt ist bereits, daß stark δ-$\mathbb{Q}$-analytische Maschinen auf ihrem gesamten Definitionsbereich stetig sind. Für quasi-stark δ-$\mathbb{Q}$-analytische Maschinen formulieren wir ein abgeschwächtes Resultat.

$\mathbb{R}$-analytische Maschinen

Sei $\mathcal{M}$ eine $\mathbb{R}$-analytische Maschine, und $\Phi_\mathcal{M}$ die von $\mathcal{M}$ berechnete Funktion. Wir betrachten die Unterteilung des Definitionsbereiches D_Φ von $\Phi_\mathcal{M}$ in Mengen

$$D_\Phi = \bigcup_\sigma D_\sigma,$$

wobei σ die Pfade des Berechnungsbaums von $\mathcal{M}$ durchläuft.
Wir betrachten nun die durch folgendes Programm gegebene Maschine $\mathcal{M}$:

```
1: z_1 = 1
2: α := 1/(1 + (z_1 · (x_1)^2))
3: y_1 := α
4: z_1 := z_1 + 1
5: print
6: goto 2
```

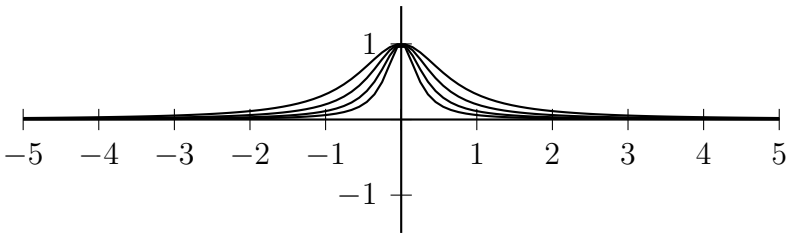
Die Maschine $\mathcal{M}$ berechnet die Funktion

$$\Phi_\mathcal{M} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} 1 & : \quad x = 0 \\ 0 & : \quad x \neq 0 \end{cases}$$

Die $n-$te Approximation $\mathcal{M}^{(n)}(x)$ ist gegeben durch

$$f_n(x) := \mathcal{M}^{(n)}(x) = \frac{1}{1 + nx^2}.$$

Mit f_n ist also eine Folge rationaler Funktionen gegeben, die punktweise, aber notwendigerweise nicht gleichmäßig gegen $\Phi_{\mathcal{M}}$ konvergiert.



Da im Programm von $\mathcal{M}$ keine bedingten Verzweigungen vorkommen, ist $D_{\sigma(x)} = \mathbb{R}$ für alle $x \in \mathbb{R}$, daß heißt, der Berechnungsbaum besteht aus einem einzigen Pfad σ. $\Phi_{\mathcal{M}}$ ist nicht auf ganz D_σ stetig, also auch nicht analytisch und nicht durch eine Potenzreihe darstellbar.
Nimmt man aus dem obigen Beispiel den Nullpunkt aus dem Definitionsbereich heraus, so ist die berechnete Funktion analytisch. Man könnte also darauf hoffen, daß analytisch berechenbare Funktionen bis auf verschwindende Ausnahmepunkte im Innern der D_σ analytisch sind. Das ist aber auch nicht der Fall.
Der folgende Satz ist auch ohne den Zusammenhang der Darstellung analytisch berechenbarer Funktionen interessant, zeigt er doch die Macht der Konstanten. Wir zeigen, daß unter Zuhilfenahme beliebiger reeller Konstanten *jede* stetige Funktion, die die rationalen Zahlen in sich abbildet, auf einem abgeschlossenen Intervall analytisch $\mathbb{R}$-berechenbar ist:

Satz 2.3.2. *Sei $f : [0,1] \to \mathbb{R}$ eine stetige Funktion mit $f(\mathbb{Q} \cap [0,1]) \subseteq \mathbb{Q}$. Dann gibt es eine Konstante c, so daß f $\mathbb{R}[c]$-analytisch berechenbar (vgl. Anmerkung zur Definition 2.2.1) ist und es gibt eine Maschine, die f so berechnet, daß alle $x \in [0,1]$ denselben Berechnungspfad haben.*

Beweis. Sei f eine Funktion, die die Voraussetzung des Satzes erfüllt. Zur Berechnung von f wird der Weierstraßsche Approximationssatz verwendet. Dazu seien die Bernsteinpolynome zu f definiert als

$$B_n(x) := \sum_{m=0}^{n} f\left(\frac{m}{n}\right) \binom{n}{m} x^m (1-x)^{n-m}.$$

Die B_n konvergieren auf $[0,1]$ gleichmäßig gegen die stetige Funktion f, siehe etwa [SW96]. Werden die Werte von $f(\frac{m}{n})$ vorausgesetzt, so sind die $B_n(x)$ $\mathbb{R}$-berechenbar. Bei der naiven Berechnung der Summe durch eine einfache Schleife werden zwar Vergleiche benötigt. Man beachte aber, daß diese Vergleiche nur *innere Variablen* betreffen, nämlich etwaige Zähler für n und m. Diese sind aber in keiner Weise von der Eingabe x abhängig. Daher führen diese Vergleiche auch nicht zu unterschiedlichen Berechnungspfaden σ für verschiedene x.
Die Bereitstellung der Werte $f(\frac{m}{n})$ erfolgt nun mit Hilfe einer Konstanten. Die Folge $\left(f(\frac{m}{n})\right)_{m \le n}$ ist eine abzählbare Folge rationaler Zahlen. Es sei $P : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ die

bijektive berechenbare Aufzählung aller Paare natürlicher Zahlen und $L, R : \mathbb{N} \to \mathbb{N}$ die Umkehrabbildungen mit $P(L(n), R(n)) = n$ und $L(P(j,k)) = j$, $R(P(j,k)) = k$ für $j, k, n \in \mathbb{N}$. Mit Hilfe von P und einer berechenbaren präfixfreien Kodierung C aller natürlichen Zahlen können wir $\left(f(\frac{m}{n})\right)_{m \leq n}$ als Folge so kodieren, daß die Information in einer einzigen reellen Zahl c gespeichert ist.
Die Konstante c ist nun im Programm der Maschine gespeichert, und immer, wenn die Berechnung eines Bernsteinpolynoms einen Wert $f(\frac{m}{n})$ verlangt, wird ein $\mathbb{R}$-berechenbares Unterprogramm aufgerufen, daß aus der Konstante c diesen Wert heraus liest. Dieses Unterprogramm kann ebenfalls Verzweigungen verwenden; erneut sind nur innere Variablen für eine Verzweigung ausschlaggebend. □

Anmerkung. In diesem Beweis zwischen *inneren Variablen* und *äußeren Variablen* unterschieden. Dabei werden Vergleiche, die nur innere Variablen betreffen, nicht als relevant für den Berechnungsbaum angesehen; wir unter inneren Variablen verstehen wir solche, die ohne Bezug auf die Eingabe im Programm definiert werden. Formal kann man die inneren Variablen bzw. Register und Speicherzellen, die nur von inneren Variablen abhängen, induktiv über der Menge der Programmbefehle definieren. Bei der Definition des Berechnungsbaumes haben wir eine solche Unterscheidung nicht gemacht, und formal entstehen durch solche Vergleiche auch Verzweigungen im Berechnungsbaum. Da die Verzweigungsbedingung aber nicht von der Eingabe abhängt, wird für alle Eingaben genau ein Pfad ausgewählt. Für die Einzugsbereiche der Berechnungspfade sind innere Variablen also nicht relevant.

Ist nun eine Funktion wie in Satz 2.3.2 gegeben, die zudem noch nirgends differenzierbar ist, so folgt, daß es selbst in offenen, nichtleeren D_σ bei analytischen Maschinen über $\mathbb{R}$ keine offene Teilmenge geben muß, auf der die berechnete Funktion durch Potenzreihen dargestellt werden kann.

Beispiel 2.3.3. Wir konstruieren nun eine Funktion $F : [0,1] \to \mathbb{R}$, die die Eigenschaften aus Satz 2.3.2 erfüllt. Tatsächlich wird F derart gestaltet sein, daß $f(\frac{m}{n})$ $\mathbb{R}$-*berechenbar* ist, das heißt, es wird nicht mehr die Verwendung einer nichtrationalen Konstante c benötigt, in der die rationalen Funktionswerte der Funktion kodiert werden.
Es sei $g : \mathbb{R} \to \mathbb{R}$ die periodische Fortsetzung der Funktion

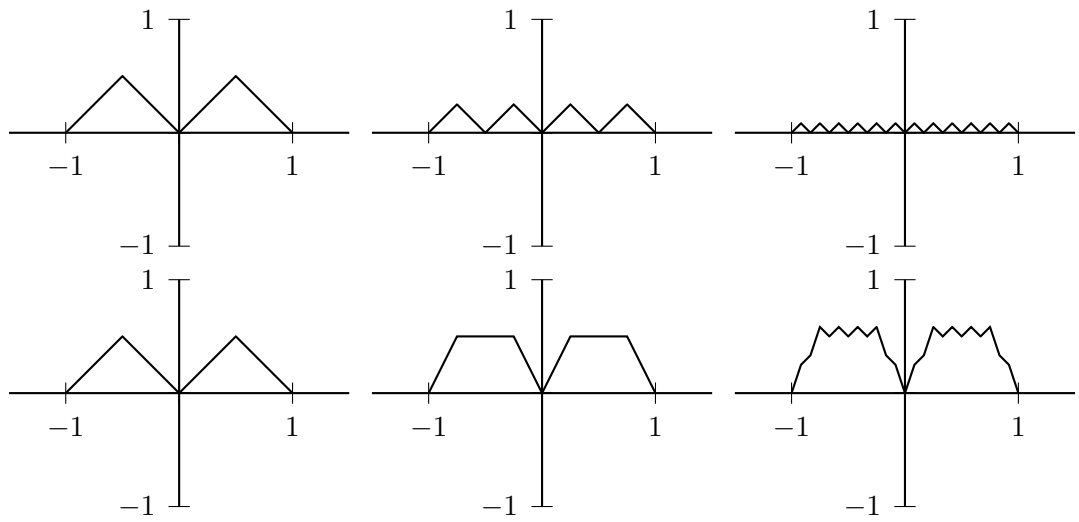
$$x \mapsto \begin{cases} x & : \ 0 \leq x \leq \frac{1}{2} \\ 1 - x & : \ \frac{1}{2} \leq x \leq 1 \end{cases}$$

Der Graph von g ist also einfach ein kleines Dreieck, unendlich oft nebeneinandergereiht. Wir skalieren nun g auf halbe Größe und addieren die skalierte Funktion zu g hinzu, und iterieren den Vorgang, und zwar jeweils mit dem Skalierungsfaktor $\frac{1}{n!}$. Wir erhalten

$$G(x) := \sum_{k=1}^{\infty} \frac{1}{k!} g(k!x)$$

Diese Summe ist an allen rationalen Stellen $\frac{p}{q}$ endlich, denn g hat an den ganzen Zahlen jeweils Nullstellen, und spätestens wenn $k \geq q$ gilt, ist somit $g(k!\frac{p}{q}) = 0$. Dies ist eine für unsere Zwecke ganz wesentliche Eigenschaft, denn durch die Endlichkeit der Summe wird die Funktion an den rationalen Stellen *endlich* $\mathbb{R}$-berechenbar. Es gilt: G ist eine stetige, nirgends differenzierbare Funktion. Dies zeigen wir in Lemma 2.3.4 am Ende des Absatzes. Mit Hilfe von Satz 2.3.2 können wir G mit einer analytischen Maschine berechnen, und zwar so, daß die einzigen verwendeten Vergleiche innere Variablen (in diesem Fall die Zählveriablen n und m) betreffen. Damit ist ein konkretes Beispiel für

Tabelle 2.3 *Die Bildung von G (oben: Summanden, unten: Summe)*



eine analytische Maschine gefunden, die für alle Eingaben den gleichen Berechnungspfad durchläuft, und deren berechnete Funktion an keiner Stelle durch eine Potenzreihe darstellbar ist.

Lemma 2.3.4. *Die oben definierte Funktion G ist auf ganz* $\mathbb{R}$ *stetig und nirgends differenzierbar.*

Beweis. Man sieht leicht, daß die Summe in der Definition von G gleichmäßig konvergiert: Es gilt mit $g(x) \leq 1$ und $k! \geq 2^k$ $(k > 3)$:

$$\left| \sum_{k=1}^{N} \frac{1}{k!} g(k!x) - G(x) \right| \leq \sum_{k=N+1}^{\infty} \frac{1}{2^k} = 2^{-N}.$$

Folglich ist G eine stetige Funktion, da die Funktion g stetig ist.
Wir zeigen nur, daß G an keiner rationalen Stelle differenzierbar ist. Mit einem etwas aufwendigeren Beweis kann man auch zeigen, daß sie nirgends differenzierbar ist, aber für unsere Zwecke genügen die rationalen Stellen, da daraus bereits folgt, daß die Funktion nirgends durch eine Potenzreihe darstellbar ist.
Sei $x = \frac{p}{q} \in \mathbb{Q}$. Wir betrachten die durch $x_j := x + \frac{1}{j!}$ definierte Folge. Wir machen nacheinander folgende Beobachtungen:

1. $g(k!x) = 0$ für $k \geq q$ und $g(k!(x + \frac{1}{j!})) = 0$ für $k \geq j > q$.

2.

$$G(x) = G\left(\frac{p}{q}\right) = \sum_{k=1}^{q-1} \frac{1}{k!} g(k!x) \text{ und } G\left(x + \frac{1}{j!}\right) = \sum_{k=1}^{j-1} \frac{1}{k!} g\left(k!\left(x + \frac{1}{j!}\right)\right)$$

3. Für hinreichend große j und für $k = 1, \ldots, q$ liegen $k!x$ und $k!x + \frac{k!}{j!}$ jeweils im gleichen Definitionsabschnitt von g (bzw. $k!x$ liegt auf der Grenze falls ganzzahlig). Für diese j und k gilt: $g(k!x + \frac{k!}{j!}) = g(k!x) \pm \frac{k!}{j!}$, das Vorzeichen ist vom Definitionsabschnitt abhängig.

4. Für $k \geq q$ ist der Wert $k!x$ ganzzahlig, und es gilt somit ebenfalls $g(k!x + \frac{k!}{j!}) = g(k!x) + \frac{k!}{j!} (= \frac{k!}{j!})$.

Insgesamt ergibt sich

$$\begin{aligned}
\left| G(x) - G\left(x + \frac{1}{j!}\right) \right| &= \left| \sum_{k=1}^{q-1} \frac{1}{k!} g(k!x) - \sum_{k=1}^{j-1} \frac{1}{k!} g\left(k!x + \frac{k!}{j!}\right) \right| \\
&= \left| \sum_{k=1}^{q-1} \frac{1}{k!} g(k!x) - \sum_{k=1}^{j-1} \frac{1}{k!} g(k!x) \pm \frac{k!}{j!} \right| \\
&= \sum_{k=1}^{j-1} \frac{k!}{j!}
\end{aligned}$$

Für den Differenzenquotienten erhalten wir

$$\frac{|G(x) - G(x + \frac{1}{j!})|}{\frac{1}{j!}} = j! \sum_{k=1}^{j-1} \frac{k!}{j!} = \sum_{k=1}^{j-1} k!$$

Damit kann der Differenzenquotient nicht konvergieren und die Funktion an der Stelle x somit nicht differenzierbar sein. □

2.3.2 δ-$\mathbb{Q}$-analytische Funktionen

Wir wissen bereits, daß stark δ-$\mathbb{Q}$-analytische Funktionen stetig sind (Satz 2.2.26). Durch die Rundung und die Genauigkeitsschranke werden also stärkere Voraussetzungen für die berechneten Funktionen geschaffen. Aber auch diese Voraussetzungen reichen nicht aus, damit solche Funktionen analytisch sind, selbst wenn ihr Definitionsbereich nicht durch äußere Variablen betreffende Vergleiche aufgespalten wird. Für die Funktion aus Beispiel 2.3.3 ist es wegen der abschnittsweisen Definition der Basisfunktion jedoch nicht einfach, eine Genauigkeitsschranke ohne Verwendung von Vergleichen zu berechnen. Wir ändern das Beispiel daher etwas ab.

Beispiel 2.3.5. Wir definieren die Funktion

$$H(x) := \sum_{k=1}^{\infty} \frac{1}{k!} \sin(k!x).$$

Auf ähnliche Weise, wie für die Funktion G aus dem letzten Beispiel, zeigt man, daß H auf $[0, 1]$ stetig und nirgends differenzierbar ist.
Für diese Funktion können wir nicht Satz 2.3.2 verwenden, da die Funktionen, aus denen die Summe gebildet wird, nicht $\mathbb{R}$-berechenbar sind. Wir approximieren die Funktion daher direkt. Prinzipiell ist klar, daß die Summe mit einer δ-$\mathbb{Q}$-Maschine berechnet werden

kann, und auch wie man sie approximiert. Wir geben hier aber genaue Abschätzungen an, damit deutlich wird, daß die Funktion und auch die Genauigkeitsschranke ohne Vergleichsoperationen (die die Eingabe betreffen) berechnet werden können.
Die Berechnung von H mit einer starken δ-$\mathbb{Q}$-Maschine erfolgt durch eine diagonale Approximation: Für wachsende n wird die Summe

$$H_n(x) := \sum_{k=1}^{n} \frac{1}{k!} s_n(k!x)$$

berechnet, wobei die Funktionen s_n wiederum Approximationen des Sinus sind. Wir definieren

$$s_n(x) := \sum_{k=1}^{(n!)^2} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

Dabei wählt man als obere Summationsgrenze $(n!)^2$, um die Berechnung der Genauigkeitsschranke zu vereinfachen. Es gilt nämlich für $|x| \leq 1$ und $k \leq n$:

$$\begin{aligned}
|\sin(k!x) - s_n(k!x)| &= \left| \sum_{j=(n!)^2+1}^{\infty} (-1)^j \frac{(k!x)^{2j+1}}{(2j+1)!} \right| \\
&\leq \sum_{j=(n!)^2+1}^{\infty} \frac{(k!)^{2j+1}}{(2j+1)!} \\
&\leq \sum_{j=(n!)^2+1}^{\infty} \frac{(k!)^{2j+1}}{j^{j+1} j!} \\
&\leq \sum_{j=(n!)^2+1}^{\infty} \frac{(k!)^{2j+1}}{((n!)^2)^{j+1} j!} \qquad \text{da } (j \geq (n!)^2) \\
&\leq \sum_{j=(n!)^2+1}^{\infty} \frac{1}{j!} \qquad \text{da } (n \geq k) \\
&\leq \sum_{j=(n!)^2+1}^{\infty} \frac{1}{2^j} \leq 2^{-(n!)^2}
\end{aligned}$$

Für $x, \tilde{x} \in [0,1]$ sieht man weiter

$$|s_n(k!x) - s_n(k!\tilde{x})| \leq \sum_{j=1}^{(n!)^2} \frac{(k!)^{2j+1} |x - \tilde{x}|^{2j+1}}{(2j+1)!}$$

Durch Anwendung des Mittelwertsatzes der Differentialrechnung und wegen $x, \tilde{x} \in [0,1]$ gilt $|x - \tilde{x}|^{2j+1} \leq (2j+1)|x - \tilde{x}|$, und man erhält durch sehr grobe Abschätzung

$$\begin{aligned}
|s_n(k!x) - s_n(k!\tilde{x})| &\leq \sum_{j=1}^{(n!)^2} \frac{(n!)^{2j+1} (2j+1) |x - \tilde{x}|}{(2j+1)!} \\
&\leq e^{(n!)^2} |x - \tilde{x}| < 3^{(n!)^2} |x - \tilde{x}|
\end{aligned}$$

Die letzte Abschätzung machen wir, damit wir die irrationale Konstante e nicht benötigen. Wir erhalten nun für einen gerundeten Wert $\tilde{x} = \rho(x, \delta)$

$$\begin{aligned}
|H_n(\tilde{x}) - H(x)| &= \left| \sum_{k=1}^{n} \frac{1}{k!} s_n(k!\tilde{x}) - \sum_{k=1}^{\infty} \frac{1}{k!} \sin(k!x) \right| \\
&= \left| \sum_{k=1}^{n} \frac{1}{k!} (s_n(k!\tilde{x}) - s_n(k!x)) + \sum_{k=1}^{n} \frac{1}{k!} (s_n(k!x) - \sin(k!x)) + \right. \\
&\qquad \left. + \sum_{k=n+1}^{\infty} \frac{1}{k!} \sin(k!x) \right| \\
&\leq \sum_{k=1}^{n} \frac{1}{k!} 3^{(n!)^2} |x - \tilde{x}| + \sum_{k=1}^{n} \frac{1}{k!} 2^{-(n!)^2} + \sum_{k=n+1}^{\infty} \frac{1}{2^k} \\
&\leq n 3^{(n!)^2} |x - \tilde{x}| + 2^{-(n!)^2+1} + 2^{-n} \leq n 3^{(n!)^2} |x - \tilde{x}| + 2^{-n+1}
\end{aligned}$$

Die Berechnung der Funktion H erfolgt nun durch Berechnung von H_n für wachsende n. Dabei wird n in einer äußeren Schleife erhöht, und dann in einer inneren Schleife die s_n mit der gerundeten Eingabe $\tilde{x}$ berechnet und die Summe gebildet. Dabei wird bei Erreichen des nächsten n die Rundungsgenauigkeit δ so lange erhöht, bis $|n3^{(n!)^2}\delta + 2^{-n+1}| < 2^{-n+2}$ gilt. Dies wird dann als Genauigkeitsschranke mit ausgegeben, und die Berechnung hängt nicht von der Eingabe x ab. Damit ist eine stark δ-$\mathbb{Q}$-analytische Maschine angegeben, die die Funktion H ohne Vergleiche, die von der Eingabe abhängen, berechnet.

Quasi-stark δ-$\mathbb{Q}$-analytische Funktionen

Wir haben bereits gesehen, daß selbst für stark δ-$\mathbb{Q}$-analytische Funktionen keine stärkeren Regularitätseigenschaften gelten. Daher gilt dies erst recht nicht für quasi-stark δ-$\mathbb{Q}$-analytische Funktionen. Diese müssen im Gegensatz zu den stark δ-$\mathbb{Q}$-analytischen Funktionen nicht einmal stetig sein, was daraus folgt, daß die Genauigkeitsschranke nicht immer, sondern nur fast immer gelten muß. Es gilt jedoch

Satz 2.3.6. *Sei $f : D \to \mathbb{R}^m$ quasi-stark δ-$\mathbb{Q}$ analytisch. Dann teilt sich der Definitionsbereich von f in abzählbar viele Mengen*

$$D = \bigcup_{n \in \mathbb{N}} G_n$$

auf, so daß die Einschränkung von f auf G_n stetig ist.

Beweis. Sei $\mathcal{M}$ eine f auf D berechnende Maschine, und es sei G_n die Menge aller Punkte aus D, für die die Genauigkeitsschranke bei der Berechnung ab der n-ten Zielkonfiguration korrekt ist. Da für jede Eingabe ein Berechnungszeitpunkt n existiert, ab dem die Genauigkeitschranke korrekt ist, gilt die Beziehung $D = \bigcup_{n \in \mathbb{N}} G_n$. Betrachten wir die Einschränkung von f auf G_n, so ist diese Funktion auf G_n sogar stark δ-$\mathbb{Q}$-analytisch. Denn eine Maschine zur starken Berechnung von f auf G_n simuliert $\mathcal{M}$ und beginnt erst mit der Ausgabe, nachdem in der simulierten Maschine die n-te Zielkonfiguration erreicht wurde. Der Eintrittszeitpunkt der Genauigkeitsschranke ist als Konstante in der Maschine gespeichert. Damit ist die Einschränkung von f auf G_n aber stetig. □

2.3.3 Darstellungen über $\mathbb{C}$

Im Gegensatz zu den reellen Zahlen folgt aus der Berechenbarkeit mit analytischen Maschinen über den komplexen Zahlen oftmals schon auf großen Bereichen eine Darstellung durch Potenzreihen. Der Grund dafür ist in den starken Konvergenzeigenschaften analytischer Funktionen zu finden; eine punktweise Konvergenz holomorpher Funktionen impliziert zwar im allgemeinen keine kompakte Konvergenz, aber in bestimmten Fällen ist dies der Fall, und selbst im allgemeinen Fall liegt doch auf dichten Teilmengen des Definitionsbereichs eine solche Konvergenz vor. Dabei ist die *kompakte Konvergenz* der relevante Konvergenzbegriff für holomorphe Funktionen, also die gleichmäßige Konvergenz auf kompakten Teilmengen.

Anmerkung. In diesem Abschnitt werden Maschinen über den komplexen Zahlen $\mathbb{C}$ betrachtet. Die Maschinen speichern also komplexe Zahlen und führen arithmetische Operationen mit ihnen aus. Dabei sollte stets beachtet werden, daß *ausschließlich* komplexe Operationen durchgeführt werden können und insbesondere die Maschine auf Real- und Imaginärteil keinen Zugriff hat. Für endliche $\mathbb{C}$ Maschinen gilt daher insbesondere, daß die nach dem Darstellungssatz entlang eines Berechnungspfades σ gegebenen rationalen Funktionen im Inneren von D_σ holomorph sind.

$\mathbb{C}$-analytische Maschinen

Zunächst untersuchen wir allgemeine $\mathbb{C}$-analytische Maschinen. Wie im reellen Fall auch können durch bedingte Verzweigungsbefehle leicht Unstetigkeitsstellen erzeugt werden; das ist auch sinnvoll, will man etwa zusammengesetzte analytische Funktionen berechnen. Wir betrachten also auch im komplexen Fall nichtleere, offene Teilmengen von Einzugsbereichen D_σ von Berechnungspfaden und untersuchen die dort von der Maschine berechneten Funktionen.

Beispiel 2.3.7. Dieses Beispiel soll verdeutlichen, daß es wichtig ist, stets *offene* Teilmengen des Definitionsbereiches bzw. von Einzugsbereichen D_σ zu betrachten.
Dazu sei $\mathcal{M}$ die analytische Maschine aus Abschnitt 2.3.1. Nun untersuchen wir, was passiert, wenn wir auch komplexe Eingaben für die Maschine zulassen. Die Maschine $\mathcal{M}$ berechnet die Funktion

$$\Phi_{\mathcal{M}} : \mathbb{C} - \left\{ \pm \frac{i}{\sqrt{n}} \middle| n \in \mathbb{N} \right\} \to \mathbb{C}, z \mapsto \begin{cases} 1 & : \quad z = 0 \\ 0 & : \quad z \neq 0,\ z \neq \pm\frac{i}{\sqrt{n}} \end{cases}$$

Das heißt, die Funktion $\Phi_{\mathcal{M}}$ setzt die Funktion aus Abschnitt 2.3.1 auf $\mathbb{C} - \{\pm\frac{i}{\sqrt{n}} | n \in \mathbb{N}\}$ fort. Die $n-$te Approximation $\Phi_{\mathcal{M}}^{(n)}$ ist offensichtlich gegeben durch

$$f_n(x) := \Phi_{\mathcal{M}}^{(n)}(z) = \frac{1}{1 + nz^2}.$$

Wieder hat der Berechnungsbaum nur einen Programmpfad σ, und es ist $D_\sigma = \mathbb{C} - \{\pm\frac{i}{\sqrt{n}} | n \in \mathbb{N}\}$. Also konvergiert $\mathcal{M}$ auf D_σ gegen eine nicht stetige Funktion.

Im Gegensatz zum reellen Fall ist die punktweise Konvergenz differenzierbarer Funktionen jedoch bereits ein hinreichendes Kriterium, daß zumindest teilweise die Differenzierbarkeit erhalten bleibt. Mit Hilfe von Konvergenzsätzen aus der Funktionentheorie, meist aus dem Zusammenhang des Satzes von Montel, erhält man folgendes Resultat.

Satz 2.3.8. *Sei $f : \mathbb{C} \supseteq D \to \mathbb{C}$ mit einer $\mathbb{C}$-analytischen Maschine $\mathcal{M}$ berechenbar. Sei weiterhin D_σ der Einzugsbereich eines Berechnungspfades σ für $\mathcal{M}$ mit nichtleerem Inneren. Dann gibt es eine in D_σ dichte und offene Teilmenge D'_σ, auf der f eine analytische Funktion darstellt.*

Beweis. Der Beweis stützt sich auf den Satz von Osgood [Osg02]. Dieser besagt, daß eine auf einer offenen Menge in $\mathbb{C}$ punktweise konvergente Folge holomorpher Funktionen auf einer dichten offenen Teilmenge bereits kompakt konvergiert.
Seien also $\mathcal{M}$, f und σ wie im Satz. Wir betrachten die Folge $(f_n)_n = (\mathcal{M}^{(n)})_n$ der n-ten Approximationen der Maschine $\mathcal{M}$. Auf D_σ sind die f_n nach dem Darstellungssatz für $\mathbb{C}$-berechenbare Funktionen alles rationale Funktionen, also insbesondere analytische Funktionen. Diese konvergieren nach Voraussetzung auf D_σ punktweise gegen die Grenzfunktion f. Ist D_σ offen, so folgt unmittelbar aus dem Satz von Osgood, daß ein in D_σ dichter Teilbereich D'_σ existiert, auf dem (f_n) kompakt konvergiert und der Grenzwert f somit eine analytische Funktion ist. Falls D_σ nicht offen ist, so wenden wir den Satz von Osgood auf das nach Voraussetzung nichtleere Innere von D_σ an. Die so erhaltene Menge D'_σ ist dicht im Innern von D_σ, also auch dicht in D_σ. □

Jede analytisch berechenbare Funktion, deren berechnende Maschine Berechnungspfade mit Einzugsbereich mit nichtleerem Inneren hat, ist also in einer offenen Teilmenge ihres Definitionsbereichs holomorph! Der Beweis dieses Satzes verwendet nur den Darstellungssatz für $\mathbb{C}$-berechenbare Funktionen und darüber hinaus lediglich Sätze aus der Funktionentheorie über konvergente Folgen analytischer Funktionen. Es wird nur verwendet, daß die n-ten Approximationen auf D_σ rationale Funktionen sind, ihre spezielle Form dagegen geht nicht in den Beweis ein.
Die Voraussetzung, daß ein Einzugsbereich D_σ mit nichtleerem Inneren existiert, ist beispielsweise gegeben, wenn auf jedem Berechnungspfad der Maschine nur endlich viele Vergleiche mit äußeren Variablen vorkommen. Denn dann gibt es auf jedem Pfad nur endlich viele Verzweigungen, und die Voraussetzungen von Lemma 2.2.24 sind gegeben.

Korollar 2.3.9. *Sei $f : D \to \mathbb{C}$ mit einer Maschine $\mathcal{M}_f$ analytisch berechenbar, so daß bei jeder Berechnung nur endlich viele von der Eingabe abhängige Vergleiche vorkommen. Dann gibt es eine offene Menge, auf der die Funktion durch eine Potenzreihe darstellbar ist.*

Im allgemeinen sind Folgen holomorpher Funktionen, die punktweise konvergieren, nicht notwendigerweise auch lokal gleichmäßig konvergent. Mit Hilfe des Satzes von Runge kann man zeigen, daß Folgen holomorpher Polynome existieren, die auf $\mathbb{C}$ gegen eine nicht stetige Grenzfunktion konvergieren. Die Folgen werden aber nicht auf eine im Sinne der Berechenbarkeit konstruktive Weise gewonnen, es könnte daher sein, daß sie nicht als Zwischenergebnisse der Berechnung einer analytischen Maschine auftreten können. Wir vermuten jedoch, daß die Einschränkung in Satz 2.3.8, daß die n-ten Approximationen nur auf einer offenen dichten Teilmenge gleichmäßig konvergieren, notwendig ist:

Vermutung 2.3.10. *Es gibt $\mathbb{C}$-analytische Maschinen, die ohne Vergleiche mit äußeren Variablen eine nichtstetige Funktion berechnen.*

2.4 Nichtdeterministische Maschinen

In diesem Abschnitt gehen wir auf Nichtdeterminismus bei analytischen Maschinen ein. Da der Schwerpunkt dieser Arbeit auf deterministischen Maschinen liegt, ist die Darstellung der Definition der nichtdeterministischen Berechnung und des nichtdeterministischen Entscheidungsproblems knapp gefaßt. Schließlich wird noch die Berechnungsmächtigkeit des Nichtdeterminismus mit der des Determinismus verglichen.
Während deterministische Maschinen bei einer Eingabe stets die gleiche Folge von Konfigurationen durchlaufen, können nichtdeterministische Maschinen bei gleicher Eingabe unterschiedliche Folgen von Konfigurationen durchlaufen. Formal wird dies bei Turing-Maschinen dadurch realisiert, daß die Konfigurationsübergänge nicht mehr durch eine *Funktion* definiert werden, sondern durch eine *Relation*. Auf diese Weise ist einer Eingabe nicht mehr eindeutig eine Berechnung zuzuordnen, sondern vielmehr eine Menge von Berechnungen. Daher werden im Zusammenhang mit Nichtdeterminismus Turing-Maschinen als Akzeptoren aufgefaßt, die Entscheidungsprobleme lösen. Eine Eingabe wird dann als akzeptiert angesehen, wenn es *wenigstens eine* akzeptierende Berechnung gibt. In diesem Zusammenhang sieht man eine nichtdeterministische Maschine häufig als Maschine an, die für ein gegebenes Problem eine Lösung rät und dann verifiziert, ob der geratene Wert tatsächlich eine Lösung des Problems darstellt.
Wir führen Nichtdeterminismus nun jedoch nicht nur für Entscheidungsprobleme ein, sondern zur Berechnung von Funktionen. Dies ist allgemeiner, da die Entscheidbarkeit einer Menge $A \subseteq \mathcal{R}^n$ äquivalent mit der Berechenbarkeit der charakteristischen Funktion χ_A ist. Wenn eine nichtdeterministische Maschine gegeben ist, die zu einer Eingabe mehrere gültige Berechnungen hat, so müssen diese gültigen Berechnungen dasselbe Resultat haben. Andernfalls ist es nicht möglich, für eine Maschine eindeutig eine von ihr berechnete Funktion zu definieren. Mit gültigen Berechnungen sind bei klassischen Turing-Maschinen haltende und bei analytischen Maschinen haltende und analytische Berechnungen gemeint.
Bei $\mathcal{R}$-Maschinen betrachten wir nun zwei verschiedene Möglichkeiten, Nichtdeterminismus zu definieren. Überträgt man das Konzept des Nichtdeterminismus für Turing-Maschinen auf $\mathcal{R}$-Maschinen, dann muß es die Möglichkeit geben, ausgehend von bestimmten Konfigurationen in mehrere unterschiedliche Folgekonfigurationen zu wechseln. Das kann dadurch realisiert werden, daß es bei einem Sprungbefehl mehrere Sprungziele geben kann, zu denen die Maschine nichtdeterministisch verzweigen kann. Diese Form des Nichtdeterminismus, die wir *schwachen Nichtdeterminismus* nennen, entspricht insofern dem Nichtdeterminismus bei Turing-Maschinen, als es bei einer gegebenen Konfiguration mehrere mögliche Folge-Operationen der Maschine geben kann.
Die zweite Möglichkeit, Nichtdeterminismus für $\mathcal{R}$-Maschinen zu definieren, besteht in der Umsetzung der intuitiven Vorstellung von Nichtdeterminismus: Es wird ermöglicht, eine Zahl (hier ein Element des Rings $\mathcal{R}$) zu raten. Im Falle der reellen Zahlen wird hier also das Raten von reellen Zahlen als Maschinenoperation eingeführt.
In beiden Fällen wird die Berechenbarkeit einer Funktion dadurch definiert, daß es eine nichtdeterministische Maschine gibt, die zu jeder Eingabe aus dem Definitionsbereich wenigstens eine haltende Berechnung hat, und bei der *alle* haltenden Berechnungen das gleiche Resultat haben. Bei unendlichen konvergierenden Berechnungen wird dies

natürlich entsprechend für analytische Berechnungen gefordert. Die Entscheidbarkeit einer Menge wird in beiden Fällen über die Berechenbarkeit der charakteristischen Funktion definiert.
Betrachtet man die unterschiedlichen Konzepte des Nichtdeterminismus (nichtdeterministische Verzweigungen bzw. Raten natürlicher Zahlen) für klassische Turing-Maschinen, so sieht man schnell ein, daß beide Konzepte bezüglich der Berechenbarkeit äquivalent sind und zum gleichen Begriff berechenbarer Funktionen führen. Das Raten einer natürlichen Zahl entspricht dem Raten endlich vieler Bits. Es gilt sogar, daß der Nichtdeterminismus nicht aus der Klasse der deterministisch berechenbaren Funktionen heraus führt. Dies sieht man leicht durch Simulation. Analog sind alle klassisch nichtdeterministisch entscheidbaren Mengen auch deterministisch entscheidbar. Wir werden in diesem Abschnitt untersuchen, inwieweit diese Konzepte auch für $\mathbb{R}$-Maschinen übereinstimmen.
Wir wollen nun die Definition der nichtdeterministischen Maschinen formal angeben. Dabei halten wir die grundlegende Definition allgemein für einen Ring $\mathcal{R}$, werden dann aber konkret den Körper $\mathbb{R}$ der reellen Zahlen betrachten.
Zunächst geben wir die Definition der mathematischen nichtdeterministischen Maschine an. Dabei gibt es noch nicht die Unterscheidung der beiden erwähnten unterschiedlichen Konzepte des Nichtdeterminismus. Hier kann zu großen Teilen die Definition der (deterministischen) mathematischen Maschine

$$\mathcal{M} = (K, K_s, K_e, K_t, \Delta, A, \text{in}, \text{out})$$

aus Abschnitt 2.1.1 übernommen werden. Um nichtdeterministische Berechnungen zu ermöglichen, ist Δ nun aber nicht mehr die Übergangsfunktion, sondern die *Übergangsrelation*, die wir auffassen als Funktion $\Delta : K \to \wp(K)$ von der Menge der Konfigurationen in ihre Potenzmenge. Für eine Berechnung $b = (k_1, k_2, \ldots)$ muß die relaxierte Bedingung $k_{i+1} \in \Delta(k_i)$ gelten.
Bei einer deterministischen Maschine $\mathcal{M}$ kann einer Eingabe x eindeutig eine Berechnung $b = (k_1, k_2, \ldots)$ zugeordnet werden mit $k_1 = \text{in}(x)$ und $k_{i+1} = \Delta(k_i)$. Ist $\mathcal{M}$ nichtdeterministisch, so ist dies nicht mehr möglich. Es ergibt sich vielmehr zu jeder Eingabe x eine Menge nichtdeterministischer Berechnungen $\mathcal{B}_x$. Eine Berechnung aus $\mathcal{B}_x$ kann haltend, nicht haltend oder auch analytisch sein. Gibt es mehrere endliche oder analytische Berechnungen in $\mathcal{B}_x$ mit unterschiedlichen Resultaten, so kann der Maschine $\mathcal{M}$ nicht eindeutig eine berechnete Funktion zugeordnet werden. Damit eine eindeutige Funktion definiert werden kann, muß gefordert werden, daß all jene Berechnungen zur Eingabe x, die endlich bzw. analytisch sind, auch das gleiche Resultat haben. Dies motiviert die folgende Definition des nichtdeterministischen Halte- bzw. Definitionsbereichs.

$$NH_{\mathcal{M}} = \left\{ x \in A^* \;\middle|\; \begin{array}{l}\text{ex. endliche Berechnung von } \mathcal{M} \text{ angesetzt auf } x \text{ und}\\ \text{alle endl. oder anal. Berechnungen auf } x \text{ haben das-}\\ \text{selbe Ergebnis}\end{array} \right\}$$

$$ND_{\mathcal{M}} = \left\{ x \in A^* \;\middle|\; \begin{array}{l}\text{ex. analytische Berechnung von } \mathcal{M} \text{ angesetzt auf } x\\ \text{und alle analyt. oder endl. Berechnungen auf } x \text{ haben}\\ \text{dasselbe Ergebnis}\end{array} \right\}$$

Bei der Definition der nichtdeterministischen Registermaschinen wird nun die Unterscheidung zwischen starkem und schwachem Nichtdeterminismus gemacht. Dazu führen

wir zwei neue Befehle für die Programmiersprache Prog ein. Für den schwachen Nichtdeterminismus, bei dem das Raten sich auf die Wahl des Programmpfades beschränkt, wird ein Sprungbefehl mit mehreren möglichen Zieladressen eingeführt. Beim starken Nichtdeterminismus kann ein Element des Ringes $\mathcal{R}$ geraten werden. Dies wird durch einen entsprechenden Programmbefehl realisiert.

Definition 2.4.1. *Eine* schwach nichtdetermistische $\mathcal{R}$*-Maschine ist eine* $\mathcal{R}$*-Maschine mit* $\mathsf{Prog}_w = \mathsf{Prog} \cup \{\mathtt{ngoto}\,\mathtt{n_1}, \ldots, \mathtt{n_k}\}$.
Eine stark nichtdetermistische $\mathcal{R}$*-Maschine ist eine* $\mathcal{R}$*-Maschine mit* $\mathsf{Prog}_s = \mathsf{Prog} \cup \{\mathtt{guess}\,\alpha\}$.
Die Semantik der Befehle ergibt sich wie folgt: Ist $k = (\alpha, \beta, \gamma, \pi, x, y, z)$ *eine Konfiguration, so gelte:*

$$\Delta(k) = \begin{cases} \{(\alpha, n_i, \gamma, \pi, x, y, z) \mid 1 \leq i \leq k\} & \textit{falls } \pi_\beta = \mathtt{ngoto}\,\mathtt{n_1}, \ldots, \mathtt{n_k} \\ \{(r, \beta+1, \gamma, \pi, x, y, z) \mid r \in \mathcal{R}\} & \textit{falls } \pi_\beta = \mathtt{guess}\,\alpha \end{cases}$$

Anmerkung.

1. Eine alternative Möglichkeit der Definition des starken Nichtdeterminismus besteht darin, daß die geratenen Zahlen als zusätzliche Eingabe auf dem Eingabeband stehen. So wird Nichtdeterminismus für endliche Maschinen von Blum, Shub und Smale definiert [BCSS98]. Diese Definition ist mit unserer offenbar äquivalent.
2. Trifft eine Maschine auf eine Operation $\mathtt{guess}\,\alpha$ bzw. $\mathtt{ngoto}\,\mathtt{n_1}, \ldots, \mathtt{n_k}$, so gibt es zur aktuellen Konfiguration mehrere Folgekonfigurationen. Wir sprechen hier von einer *nichtdeterministischen Verzweigung*. Unter Berücksichtigung aller nichtdeterministischen Verzweigungen kann man den Berechnungsbaum zu *einer Eingabe* definieren. Die Pfade dieses Baums bezeichnen wir als *nichtdeterministische Berechnungspfade*. Diese müssen wohl unterschieden werden von den Berechnungspfaden, die sich bei einer deterministischen Maschine durch unterschiedliche Eingaben ergeben. Sind die Ausgaben der Maschine entlang eines Berechnungspfades konvergent, so sprechen wir von einem *(gegen den Wert x) konvergenten Berechnungspfad*, andernfalls von einem *divergenten Berechnungspfad*.
3. Für die von einer Maschine berechnete Funktion sind nur die haltenden bzw. analytischen Berechnungspfade relevant. Ist ein Berechnungspfad (d.h. die zugehörige Berechnung) divergent, dann hat das Verhalten der Maschine entlang dieses Pfades keinen Einfluß auf den von der Maschine berechneten Wert. Diese Tatsache wird im Sinne einer Beweistechnik auf folgende Weise genutzt: Wird im Laufe einer Berechnung festgestellt, daß "falsch geraten" wurde, dann darf dieser nichtdeterministische Berechnungspfad nicht zu einem Ergebnis führen. Das wird dadurch realisiert, daß die Maschine das `exception`-Kommando aufruft. Wenn im folgenden in Beweisen vom *Beenden eines nichtdeterministischen Berechnungspfades* gesprochen wird, dann ist dieser Zusammenhang gemeint.
4. Schwach nichtdeterministische Maschinen können keine reellen Zahlen raten. Mit Hilfe nichtdeterministischer Verzweigungen lassen sich aber leicht natürliche oder sogar rationale Zahlen raten: Ein Unterprogramm zum Raten einer natürlichen Zahl inkrementiert in einer Schleife einen Zähler, und ein nichtdeterministischer Sprungbefehl ermöglicht es, aus der Schleife herauszuspringen.

Bei klassischen Turing-Maschinen sind Determinismus und Nichtdeterminismus bezüglich ihrer Berechnungsmächtigkeit äquivalent. Wir werden sehen, daß dies bei $\mathbb{R}$-Maschinen bezüglich der Entscheidbarkeit bei endlichen Berechnungen ebenfalls gilt, und insbesondere daß sowohl starke als auch schwache nichtdeterministisch entscheidbare Mengen auch deterministisch entscheidbar sind. Die Menge der stark nichtdeterministisch

berechenbaren Funktionen ist jedoch echt größer als die der deterministisch und schwach nichtdeterministisch berechenbaren.

Satz 2.4.2. *Die Menge der schwach nichtdeterministisch $\mathbb{R}$-berechenbaren Funktionen und die der deterministisch $\mathbb{R}$-berechenbaren Funktionen sind gleich.*

Beweis. Die Äquivalenz von schwachem Nichtdeterminismus und Determinismus sieht man wie im klassischen Fall bei Turing-Maschinen durch Simulation. Die (endlich) vielen nichtdeterministischen Verzweigungen einer Maschine bei fester Eingabe werden durch eine deterministische Maschine parallel simuliert, bis eine der nichtdeterministischen Berechnungen hält. Das Ergebnis dieser Berechnung wird dann ausgegeben. Da nach Voraussetzung die Eingabe aus dem Definitionsbereich der Maschine ist und somit wenigstens eine der nichtdeterministischen Berechnungen hält, wird dieses Verfahren nach endlicher Zeit ein Ergebnis liefern. Das Ergebnis ist auch korrekt, da alle haltenden Berechnungen der nichtdeterministischen Maschine das gleiche Resultat haben. □

Man sieht sehr schnell ein, daß die Äquivalenz zwischen starkem und schwachem Nichtdeterminismus bezüglich Berechenbarkeit nicht mehr gilt:

Satz 2.4.3. *Die Funktion $\sqrt{\cdot} : \mathbb{R}^+ \to \mathbb{R}$, $x \mapsto \sqrt{x}$ ist stark nichtdeterministisch $\mathbb{R}$-berechenbar, aber nicht $\mathbb{R}$-berechenbar.*

Beweis. Daß die Funktion nicht $\mathbb{R}$-berechenbar ist, folgt aus dem Darstellungssatz: Die Wurzel ist auf keiner nichtleeren offenen semi-algebraischen Menge durch eine rationale Funktion darstellbar.
Eine stark nichtdeterministische Maschine $\mathcal{M}$, die $\sqrt{x}$ berechnet, rät bei einer Eingabe x zunächst den Wert w der Wurzel von x. Nun vergleicht sie w^2 mit x, gibt bei Gleichheit w aus und geht andernfalls in eine Endlosschleife. Offenbar gibt es für jede Eingabe x genau einen haltenden nichtdeterministischen Berechnungspfad. □

Bezüglich der Entscheidbarkeit von Mengen sind starker und schwacher Nichtdeterminismus jedoch äquivalent:

Satz 2.4.4. *Die Mengen der stark nichtdeterministisch $\mathbb{R}$-entscheidbaren Mengen, die der schwach nichtdeterministisch $\mathbb{R}$-entscheidbaren Mengen und die der deterministisch $\mathbb{R}$-entscheidbaren Mengen sind gleich.*

Beweis. Daß jede schwach nichtdeterministisch $\mathbb{R}$-entscheidbare Menge auch deterministisch $\mathbb{R}$-entscheidbar ist, folgt durch Übergang zur charakteristischen Funktion aus Satz 2.4.2.
Sei A nun eine stark nichtdeterministisch $\mathbb{R}$-entscheidbare Menge und $\mathcal{M}$ eine stark nichtdeterministische Maschine, die die charakteristische Funktion χ_A berechnet. Wir beschreiben eine deterministische Maschine $\mathcal{M}_D$, die χ_A deterministisch berechnet. Dazu wird bei Eingabe x die Maschine $\mathcal{M}$ simuliert. Dabei sind die im Laufe einer Berechnung geratenen Zahlen bei einer deterministischen Berechnung nicht bekannt, und werden als Parameter geführt. Die Inhalte der Register, des Speichers und des Ausgabebandes bilden nun rationale Funktionen in diesen geratenen Zahlen. Es werden nun analog der Vorgehensweise im konstruktiven Darstellungssatz die Koeffizienten dieser rationalen Funktionen berechnet. Immer, wenn eine neue Zahl geraten wird, wird auch

eine neue symbolische Variable eingeführt. Bei Verzweigungsbefehlen der Form `if then else` muß die Simulation durch parallele Berechnungen in zwei neue Simulationen aufgespalten werden. Die Bedingung $\alpha > 0$ bzw. $\alpha = 0$ wird als semi-algebraische Bedingung an die geratenen Zahlen für jede Verzweigung und jeden Pfad σ in eine Liste L_σ semi-algebraischer Bedingungen aufgenommen. Auf diese Weise wird wie im Darstellungssatz ein Berechnungsbaum erzeugt, der jedoch hier nicht von der Eingabe (diese ist gegeben) abhängt, sondern von den geratenen Zahlen.
Trifft die Simulation für einen der parallel simulierten Pfade auf eine haltende Berechnung, so muß festgestellt werden, ob diese Berechnung tatsächlich auftreten kann, d.h. ob die semi-algebraischen Bedingungen in der Liste L_σ lösbar sind. Sind sie nicht lösbar, wird dieser Pfad verworfen und die Simulation entlang der anderen Pfade fortgesetzt. Sind die semi-algebraischen Bedingungen jedoch lösbar, gibt es also ein Tupel geratener Zahlen $(\zeta_1, \ldots, \zeta_n)$, das diesen Berechnungspfad erzeugt. Die Ausgabe der nichtdeterministischen Maschine bei der zugehörigen haltenden Konfiguration ist also gleich Ergebnis $\chi_A(x)$, da nach Definition des Nichtdeterminismus alle haltenden Berechnungen dasselbe Ergebnis liefern. Diese Ausgabe wird bestimmt, indem für die rationale Funktion der Ausgabe $y(\zeta_1, \ldots, \zeta_n)$ die Gleichungen $y(\zeta_1, \ldots, \zeta_n) = 0$ bzw. $y(\zeta_1, \ldots, \zeta_n) = 1$ zur Liste L_σ hinzugefügt werden. Da die nichtdeterministische Maschine mit Ausgabe 0 oder 1 hält, ist genau eines der beiden Gleichungssysteme lösbar. Diese Gleichungssysteme bestehen aus Gleichungen der Form $r(x) = 0$, $r(x) \geq 0$ oder $r(x) \neq 0$. Die Lösbarkeit solcher semi-algebraischer Gleichungssysteme ist mit Hilfe von Tarskis Quantorenelimination stets entscheidbar. Ein Beweis dieser Tatsache würde jedoch den Rahmen der Arbeit sprengen, wir verweisen auf die Fachliteratur [BPR06]. Man sieht jedoch leicht, daß die Lösbarkeit semi-algebraischer Systeme auf die Lösbarkeit algebraischer Gleichungssysteme reduziert werden kann, siehe dazu Lemma 2.4.5.
Nun muß noch sichergestellt werden, daß die Maschine $\mathcal{M}_D$ in ihrer Simulation tatsächlich auf einen haltenden Berechnungspfad von $\mathcal{M}$ trifft. Nach Voraussetzung gibt es aber für jede Eingabe $x \in \mathbb{R}$ ein Tupel geratener Zahlen $(\zeta_1, \ldots, \zeta_n)$, für die die Berechnung hält, da $\mathcal{M}$ die Menge A entscheidet. Da alle möglichen Berechnungen simuliert werden, wird auch diese Berechnung simuliert. □

Bei endlichen Berechnungen verhält es sich mit dem Nichtdeterminismus also zumindest die Entscheidbarkeit betreffend wie bei Turing-Maschinen: Die Berechnungsmächtigkeit wird durch Nichtdeterminismus nicht verändert, bei Entscheidbarkeit sind die verschiedenen Varianten des Nichtdeterminismus auch gleichmächtig.
Wir zeigen noch das im Beweis zu Satz 2.4.4 erwähnte

Lemma 2.4.5. *Seien r_i, $1 \leq i \leq n$, rationale Funktionen. Sei $S = \{r_i \diamond 0 \,|\, 1 \leq i \leq n\}$ ein System von Gleichungen und Ungleichungen, wobei $\diamond \in \{=, \neq, >, <, \geq, \leq\}$. Dann läßt sich dieses System auf ein algebraisches Gleichungssystem zurückführen.*

Beweis. Zunächst wird aus dem System S ein bezüglich der Lösbarkeit äquivalentes System S' erzeugt, indem jede Ungleichung aus S unter Hinzufügen einer neuen Variable durch eine oder zwei Gleichungen ersetzt wird. Dazu führen wir für jedes i die Variablen

z_{i_1}, z_{i_2} ein und nehmen Ersetzungen der folgenden Art vor:

$$\begin{aligned}
r_i(x_1,\ldots,x_n) \geq 0 &\rightarrow r_i(x_1,\ldots,x_n) - z_{i_1}^2 = 0\\
r_i(x_1,\ldots,x_n) \leq 0 &\rightarrow r_i(x_1,\ldots,x_n) + z_{i_1}^2 = 0\\
r_i(x_1,\ldots,x_n) \neq 0 &\rightarrow r_i(x_1,\ldots,x_n) \cdot z_{i_i} - 1 = 0\\
r_i(x_1,\ldots,x_n) > 0 &\rightarrow r_i(x_1,\ldots,x_n) \cdot z_{i_1} - 1 = 0 \text{ und } r_i(x_1,\ldots,x_n) - z_{i_2}^2 = 0\\
r_i(x_1,\ldots,x_n) < 0 &\rightarrow r_i(x_1,\ldots,x_n) \cdot z_{i_1} - 1 = 0 \text{ und } r_i(x_1,\ldots,x_n) + z_{i_2}^2 = 0
\end{aligned}$$

Schließlich erhält man ein algebraisches Gleichungssystem S'' aus S', indem die Gleichungen, in denen rationale Funktionen $r_i = \frac{p_i}{q_i}$ vorkommen, mit dem Nenner q_i multipliziert werden. Dabei muß jeweils die Bedingung $q_i \neq 0$ hinzugefügt werden, was wie oben durch Hinzunahme einer Gleichung mit einer neuen Variablen erreicht wird. □

Wir wenden uns nun dem Nichtdeterminismus bei unendlichen Berechnungen zu. Zunächst zeigen wir, daß analog zum deterministischen Fall das Konvergenzproblem für nichtdeterministische Maschinen nicht mit Hilfe einer nichtdeterministischen Maschine entschieden werden kann.

Satz 2.4.6. *Das Konvergenzproblem für schwach nichtdeterministische Maschinen ist nicht schwach deterministisch analytisch entscheidbar:*
Gegeben eine schwach nichtdeterministische Maschine $\mathcal{M}$ und Eingabe x, ist dann die Maschine $\mathcal{M}$ auf der Eingabe x im schwach nichtdeterministischen Sinne konvergent?

Beweis. Der Beweis erfolgt durch Diagonalisierung. Wir nehmen an, es gäbe eine schwach nichtdeterministische Maschine $\mathcal{K}$, die das Konvergenzproblem löst, d.h. die bei Eingabe von $(\mathcal{M}, x)$ gegen 1 konvergiert, falls $\mathcal{M}$ auf x konvergiert und ansonsten gegen 0. Dabei ist mit Konvergenz die nichtdeterministische Konvergenz gemeint, d.h. es muß einen nichtdeterministischen Pfad geben, dessen Ausgaben gegen 0 bzw. 1 konvergieren, und die Ausgaben aller konvergenten Pfade konvergieren gegen dieses Ergebnis. Wir konstruieren damit eine schwach nichtdeterministische Maschine $\mathcal{W}$ und erhalten einen Widerspruch. Die Maschine $\mathcal{W}$ simuliert bei Eingabe x die Maschine $\mathcal{K}$ auf der Eingabe (x, x). Dabei wird bei einer nichtdeterministischen Verzweigung von $\mathcal{B}$ die Simulation ebenfalls nichtdeterministisch verzweigt. Immer, wenn die Ausgabe der simulierten Maschine $\mathcal{B}$ nun größer als $\frac{2}{3}$ ist, wird nacheinander 0 und 1 ausgegeben, also eine divergente Berechnung erzeugt. Im anderen Fall wird die Ausgabe der simulierten Maschine $\mathcal{B}$ ausgegeben.
Die Frage, ob die Maschine $\mathcal{W}$ nun mit Eingabe $\mathcal{W}$ im schwach nichtdeterministischen Sinne konvergiert, führt zu einem Widerspruch: Ist $\mathcal{W}(\mathcal{W})$ konvergent, so konvergieren die Ausgaben jedes konvergenten Berechnungspfades der Simulation von $\mathcal{B}$ gegen 1 und sind somit fast immer größer als $\frac{2}{3}$. Dann erzeugt $\mathcal{W}$ nach Konstruktion aber für alle diese Pfade die divergente Ausgabe von alternierenden Nullen und Einsen. Für alle nicht konvergenten Pfade der Simulation wird ebenfalls eine divergente Ausgabe erzeugt, da die Ausgabe des Pfades durch die Simulation einfach weitergereicht wird, außer in dem Fall, daß Ausgaben größer als $\frac{2}{3}$ sind, aber in diesem Fall kann ebenfalls keine Konvergenz vorliegen. Im Widerspruch zur Annahme ist $\mathcal{W}(\mathcal{W})$ also nicht konvergent, da kein konvergenter nichtdeterministischer Pfad existiert.
Ist umgekehrt $\mathcal{W}(\mathcal{W})$ divergent, so muß wenigstens ein Pfad der Simulation von $\mathcal{B}$ gegen

0 konvergieren, und alle konvergierenden Pfade müssen zum gleichen Grenzwert konvergieren. Für all diese Pfade wird die Ausgabe fast immer kleiner als $\frac{2}{3}$ sein, und somit wird die Ausgabe von $\mathcal{W}$ für diese Pfade ebenfalls gegen 0 konvergieren. Da $\mathcal{W}$ bei ihrer Simulation aus divergenten Pfaden keine konvergenten erzeugt, sind diese für die Betrachtung irrelevant. Damit existieren konvergente Berechnungspfade von $\mathcal{W}$, und alle konvergenten Berechnungspfade konvergieren gegen 0. Die Maschine ist also im schwach nichtdeterministischen Sinne konvergent im Widerspruch zur Annahme. □

Während im endlichen Fall Determinismus und schwacher Nichtdeterminismus die gleichen berechenbaren Funktionen erzeugen, ist dies im analytischen Fall nicht mehr so. Die Möglichkeit der nichtdeterministischen Verzweigung gestattet es, das Konvergenzproblem für deterministische analytische Maschinen schwach nichtdeterministisch zu entscheiden. Die wesentliche Idee hierbei ist, daß für eine gegebene Maschine nichtdeterministisch geraten wird, ob diese konvergiert oder nicht und dann überprüft wird, ob richtig geraten wurde.

Satz 2.4.7. *Die Klasse der (schwach) nichtdeterministisch analytisch $\mathbb{R}$-berechenbaren Funktionen ist echt größer als die der deterministisch analytisch $\mathbb{R}$-berechenbaren.*

Beweis. Wir führen den Beweis für schwach nichtdeterministische Maschinen. Wir zeigen später, daß die Menge der schwach nichtdeterministisch berechenbaren Funktionen eine Teilmenge der stark nichtdeterministisch berechenbaren Funktionen ist, und damit folgt die Aussage auch für stark nichtdeterministische Maschinen.
Es wird nun gezeigt, daß das Konvergenzproblem für analytische Maschinen mit einer schwach nichtdeterministischen analytischen Maschine entscheidbar ist. Dazu konstruieren wir eine schwach nichtdeterministische Maschine $\mathcal{N}$, die die Funktion

$$f : \mathbf{M}_{\mathbb{R}} \to \mathbb{R},\ f(\mathcal{M}) = \begin{cases} 1 & : \quad \mathcal{M} \text{ konvergiert auf leerem Band} \\ 0 & : \quad \text{sonst} \end{cases}$$

berechnet. Wir wissen bereits, daß diese Funktion nicht deterministisch analytisch $\mathbb{R}$ berechenbar ist.
Die Maschine $\mathcal{N}$ erhält als Eingabe eine Maschine $\mathcal{M}$. Um die Konvergenz von $\mathcal{M}$ zu entscheiden, "rät" $\mathcal{N}$ zunächst, ob $\mathcal{M}$ konvergiert oder nicht. Dazu wird einfach ein `ngoto`-Befehl verwendet, der in zwei verschiedene Unterprogramme springt. In dem Fall, daß $\mathcal{M}$ konvergiert, obwohl geraten wurde, sie divergiere, oder im umgekehrten Fall, daß $\mathcal{M}$ divergiert obwohl Konvergenz geraten wurde, darf die nichtdeterministische Berechnung entlang dieses falschen Berechnungspfades nicht konvergieren. Es ist hier wichtig, sich die Definition der Konvergenz einer nichtdeterministischen Maschine vor Augen zu führen: Sie liegt vor, wenn es bei einer Eingabe wenigstens eine konvergente Berechnung gibt, und wenn zusätzlich alle Berechnungen, die konvergieren, denselben Grenzwert haben. Dabei entstehen verschiedene nichtdeterministische Berechnungen dadurch, daß durch einen `ngoto`-Befehl mit mehreren Sprungzielen verzweigt wird. Im folgenden wird auch ausgenutzt, daß mit Hilfe von schwach nichtdeterministischen Verzweigungen natürliche Zahlen geraten werden können.

1. Wir betrachten zunächst den Fall, daß $\mathcal{M}$ als konvergent angenommen wird. $\mathcal{N}$ wird schließlich als Ausgabe 1 auf dem Ausgabeband stehen haben, falls richtig

geraten wurde. Ansonsten wird diese nichtdeterministische Teilberechnung nicht konvergieren und somit für die berechnete Funktion irrelevant sein. $\mathcal{N}$ testet, ob die Folge der Zwischenausgaben $\mathcal{M}^{(k)}$ eine Cauchyfolge bildet. Dazu wird für $n = 1, 2, \ldots$ parallel (hier noch deterministisch) überprüft, ob $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| < 2^{-n}$ für alle hinreichend großen k, l gilt. Immer, wenn ein neues n in die Berechnung aufgenommen wird, rät $\mathcal{N}$ ein $n_0(n)$. Für dieses $n_0(n)$ soll für alle $k, l > n_0(n)$ die Beziehung $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| < 2^{-n}$ gelten. Dies wird durch parallele Berechnungen für alle n und immer größere k, l überprüft, und nach jedem erfolgreichen Test produziert $\mathcal{N}$ eine 1 auf dem Ausgabeband. Wenn ein Paar $k, l > n_0(n)$ gefunden wird mit $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| \geq 2^{-n}$, dann ist das geratene $n_0(n)$ falsch und dieser nichtdeterministische Berechnungspfad wird durch ein `exception`-Kommando zur Nichtkonvergenz gezwungen.
Wenn die Eingabemaschine $\mathcal{M}$ nun tatsächlich konvergiert, so gibt es für jedes n ein $n_0(n)$, so daß für alle $k, l > n_0(n)$ die Beziehung $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| < 2^{-n}$ gilt. Das heißt, daß es in diesem Fall für jedes n einen nichtdeterministischen Berechnungspfad gibt, der unendlich oft eine 1 auf das Ausgabeband schreibt, nämlich jenen, bei dem ein korrektes $n_0(n)$ geraten wurde. Das gilt natürlich für alle nichtdeterministischen Pfade, bei denen die Tests nie scheitern. Also konvergiert das Ergebnis in diesem Fall tatsächlich gegen 1.
Falls dagegen nicht richtig geraten wurde und $\mathcal{M}$ in Wahrheit divergiert, so gibt es ein n, so daß es für alle $n_0(n)$ immer $k, l > n_0(n)$ gibt mit $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| \geq 2^{-n}$. Dann wird aber der entsprechende nichtdeterministische Berechnungspfad, bei dem dieses $n_0(n)$ geraten wurde, abgebrochen; das heißt, kein nichtdeterministischer Berechnungspfad kann in diesem Fall konvergieren und somit wegen paralleler Berechnung für alle n die gesamte Berechnung nicht. In diesem Fall wird nämlich die Maschine bei diesem n für jedes geratene $n_0(n)$ die Berechnung abbrechen, sobald das Gegenbeispielpaar k, l gefunden wurde.

2. Nun betrachten wir den Fall, daß $\mathcal{N}$ rät, $\mathcal{M}$ sei nicht konvergent. Falls die Annahme stimmt, dann gibt es ein $n \in \mathbb{N}$, so daß für alle $n_0 \in \mathbb{N}$ es stets $k, l \geq n_0$ gibt mit $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| \geq 2^{-n}$. $\mathcal{N}$ rät also zunächst dieses n. Dann werden für immer größere n_0 Paare $k(n_0), l(n_0)$ geraten und es wird überprüft, ob $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| \geq 2^{-n}$ gilt. Ist dies der Fall, so wird eine 0 auf dem Ausgabeband erzeugt. Andernfalls wird dieser nichtdeterministische Berechnungspfad abgebrochen.
Falls die Annahme richtig war und $\mathcal{M}$ wirklich divergiert, dann existiert ein n, so daß $\mathcal{N}$ für alle n_0 richtige Werte für $k(n_0), l(n_0)$ rät und es gibt eine gegen 0 konvergierende Ausgabe.
Wenn aber $\mathcal{M}$ tatsächlich entgegen der Annahme konvergiert, so gibt es für jedes geratene n ein n_0, für das kein Paar $k(n_0), l(n_0)$ mit $|\mathcal{M}^{(k)} - \mathcal{M}^{(l)}| \geq 2^{-n}$ geraten werden kann; jeder nichtdeterministische Berechnungspfad wird also in diesem Fall abgebrochen und die Maschine kommt über dieses n_0 nicht hinaus.

□

Die Möglichkeit, durch Raten verschiedene Berechnungspfade "auszuprobieren", vergrößert also die Berechnungsmächtigkeit einer Maschine. Daß das Raten von reellen Zahlen zumindest das nichtdeterministische Verzweigen umfasst, ist leicht einzusehen:

Lemma 2.4.8. *Jede schwach nichtdeterministisch analytisch $\mathbb{R}$-berechenbare Funktion ist auch stark nichtdeterministisch analytisch $\mathbb{R}$-berechenbar.*

Beweis. Das Raten einer reellen Zahl hat das Aufspalten einer Berechnung in unendlich viele weitere Berechnungspfade zur Folge. Ein Befehl einer schwach nichtdeterministischen Maschine der Form $\texttt{ngoto}\,\texttt{n}_1, \ldots, \texttt{n}_k$ kann beispielsweise dadurch simuliert werden, daß eine reelle Zahl geraten wird, die reelle Zahlengerade in k Bereiche unterteilt wird, und das Sprungziel in Abhängigkeit von dem der geratenen Zahl zugeordneten Bereich ausgewählt wird. Alternativ bringt man zunächst das Programm in eine Normalform, bei der jeder `ngoto`-Befehl genau zwei Sprungziele hat. Dann rät die Maschine am Anfang des Programms eine reelle Zahl, und bei jeder simulierten nichtdeterministischen Verzweigung wird diese Zahl als Kodierung eines unendlichen binären Entscheidungsbaums aufgefaßt. □

Der Abschnitt wird nun mit der Vermutung abgeschlossen, daß starker und schwacher Nichtdeterminismus im analytischen Fall separiert sind.

Vermutung 2.4.9. *Die Menge der schwach nichtdeterministischen analytisch berechenbaren Funktionen ist eine* echte *Teilmenge der stark nichtdeterministisch analytisch berechenbaren Funktionen.*

2.5 Fazit

In diesem Kapitel wurden die analytischen Maschinen auf ähnliche Weise wie in [Cha98] eingeführt. Dazu wurde zunächst das abstrakte Maschinenmodell gegeben, das dann durch Registermaschinen spezialisiert wurde, die über exakte Arithmetik über einem Ring $\mathcal{R}$ verfügen. Das Modell ermöglicht es, berechenbare Funktionen und entscheidbare Mengen über den reellen und komplexen Zahlen zu definieren. Dabei können endliche und unendliche konvergierende Berechnungen unterschieden werden. Mit Hilfe von δ-$\mathbb{Q}$-Maschinen können reellwertige Funktionen durch rationale Berechnungen approximiert werden.
Aus Kardinalitätsgründen können nicht alle rellen Funktionen berechenbar sein. Ein konkretes Beispiel für ein Problem, das nicht analytisch entscheidbar ist, ist das Konvergenzproblem für analytische Maschinen, das Analogon zum Halteproblem für Turing-Maschinen. Wir haben im Gegensatz zu [Cha98] auch das Beschränktheitsproblem für analytische Maschinen betrachtet und sind so zum Beweis der Tatsache gelangt, daß auch die Menge der Häufungswerte einer beschränkten Folge und damit insbesondere ihr größter und kleinster Häufungswert nicht analytisch berechenbar sind.
Im Gegensatz zum endlichen Fall kann aber das Konvergenzproblem für analytische Maschinen durch mehrere hintereinander ausgeführte analytische Maschinen entschieden werden. Vierke [Vie96] konnte zeigen, daß drei hintereinander ausgeführte analytische Maschinen hierfür genügen. Wir haben dieses Resultat verschärft und gezeigt, daß bereits die Komposition zweier analytischer Maschinen genügt, um das Konvergenzproblem zu entscheiden und damit ein diesbezüglich offenes Problem aus [Cha98] gelöst.
Die Struktur der Berechnung auf $\mathcal{R}$-Maschinen wird mit Hilfe des Berechnungsbaumes der Maschine ersichtlich. Die Betrachtung dieses Baumes und seiner Berechnungspfade ermöglicht es, die endlich berechenbaren Funktionen zu charakterisieren und den Darstellungssatz (vgl. [BCSS98]) zu formulieren. Durch die arithmetische Natur der Berechnungen von $\mathcal{R}$-Maschinen ergibt sich, daß die $\mathcal{R}$-berechenbaren Funktionen auf Teilbereichen ihrer Definitionsmenge rationale Funktionen darstellen. Die Teilbereiche ergeben sich durch die Verzweigungsbefehle und können damit als semi-algebraische Mengen charakterisiert werden. Betrachtet man analytische Berechnungen auf $\mathbb{R}$ und $\mathbb{C}$, so stellen die Zwischenergebnisse dieser Berechnungen rationale Funktionen dar, die gegen das Ergebnis der Berechnung konvergieren. Wir haben gezeigt, daß über $\mathbb{R}$ aus dieser Beobachtung nicht geschlossen werden kann, daß die Funktionen auf Teilbereichen durch Potenzreihen dargestellt werden können. Dies gilt auch für Funktionen, die durch δ-$\mathbb{Q}$-Maschinen approximiert werden können, sogar für solche, bei denen für jedes ausgegebene Zwischenergebnis die Genauigkeit stets bekannt ist. Über $\mathbb{C}$ konnten wir den Darstellungssatz für $\mathbb{C}$-berechenbare Funktionen auf analytisch $\mathbb{C}$-berechenbare Funktionen dahingehend verallgemeinern, daß analytisch berechenbare Funktionen, die für jede Eingabe mit endlich vielen Vergleichsoperationen auskommen, zumindest auf Teilbereichen durch Potenzreihen dargestellt werden können.
Das Kapitel schließt mit einer Untersuchung des Nichtdeterminismus bei analytischen Maschinen ab. Für endliche Maschinen wurde Nichtdeterminismus in [BCSS98] eingeführt durch die Möglichkeit, reelle Zahlen raten zu können. Wir haben hier nun gesehen, daß die Verallgemeinerung des Nichtdeterminismus von Turing-Maschinen zu zwei

verschiedenen Möglichkeiten führt, Nichtdeterminismus für $\mathbb{R}$-Maschinen zu definieren. Die erste besteht im Raten von reellen Zahlen, und die zweite in nichtdeterministischen Verzweigungen bei Sprungbefehlen. Es stellte sich heraus, daß die beiden Formen des Nichtdeterminismus äquivalent bei Entscheidbarkeitsfragen sind, daß sie jedoch zu unterschiedlichen Mengen berechenbarer Funktionen führen. Während bei endlichen Maschinen Nichtdeterminismus und Determinismus auch über $\mathbb{R}$ bei Entscheidbarkeitsfragen äquivalent sind, haben nichtdeterministische analytische Maschinen eine größere Berechnungsmächtigkeit als deterministische analytische Maschinen.

Kapitel 3
Berechenbare analytische Funktionen

3.1 Berechenbarkeit bei analytischen Funktionen

Die klassischen analytischen Funktionen haben eine große Bedeutung in der Mathematik. Die komplexe Funktionentheorie ist eines der schönsten Gebäude innerhalb der Mathematik, und die Eigenschaften analytischer Funktionen sind reichhaltig. Bereits bei der Entwicklung der Theorie von Riemann, Weierstraß und vielen anderen haben auch Aspekte der Berechenbarkeit eine Rolle gespielt, wenn auch nicht auf eine maschinentheoretische Art und Weise. Allein die Darstellung durch Potenzreihen stellt eine enge Verbindung zwischen berechenbaren Funktionen und analytischen Funktionen her. Denn in gewisser Hinsicht stellt eine Potenzreihe ja schon eine Berechnungsvorschrift dar. Man kann sogar noch weiter gehen und eine Potenzreihe als Maschine auffassen, wie in [Hot94] geschehen.
Gegenstand dieses Kapitels ist es, den Begriff der *berechenbaren analytischen Funktion* zu diskutieren und die Berechenbarkeit analytischer Funktionen mit Hilfe der zuvor untersuchten Maschinenmodelle zu definieren. Nicht jede analytische Funktion wird man in einem intuitiven Sinne als berechenbar ansehen, kann man doch sehr viel Information in der Koeffizientenfolge einer Potenzreihe kodieren. Setzt man etwa den k-ten Koeffizienten einer Potenzreihe auf 1, wenn die k-te analytische Maschine einer Aufzählung konvergiert und ansonsten auf 0, dann liegt eine Potenzreihe vor, die im Innern des Einheitskreises konvergiert und die im intuitiven Sinne sicherlich nicht als berechenbar gilt.
Dieses Beispiel zeigt auch gleich zwei Möglichkeiten auf, die Berechenbarkeit einer analytischen Funktion zu charakterisieren. Auf der einen Seite kann man die Berechenbarkeit der Funktion voraussetzen, auf der anderen die Berechenbarkeit der Koeffizientenfolge. Eine zentrale Frage dieses Kapitels ist, inwieweit diese beiden unterschiedlichen Möglichkeiten zum selben Berechenbarkeitsbegriff führen.
Unter dem Aspekt der bereits betrachteten Maschinenmodelle können wir analytische Funktionen dann als berechenbar betrachten, wenn sie mit Hilfe einer Maschine berechnet werden können. Hier muß natürlich das Maschinenmodell genauer angegeben werden. Analytische Funktionen haben Teilmengen der komplexen Zahlenebene als Definitionsbereich und sind (abgesehen von reellen Konstantenfunktionen) immer allgemein komplexwertig. Daher kann man diese Funktionen nicht mit einer klassischen Turing-Maschine berechnen. Mit Hilfe analytischer Maschinen kann man solche Funktionen aber definieren. Zunächst kann man mit (endlichen) $\mathbb{C}$-Maschinen komplexwertige Funktionen definieren. Wir werden aber sehen, daß die Menge der analytischen Funktionen,

die gleichzeitig endlich $\mathbb{C}$-berechenbar sind, sehr überschaubar ist. Deshalb werden wir allgemeinere analytische Maschinenmodelle für die Definition der Berechenbarkeit analytischer Funktionen zugrunde legen. Für jedes der untersuchten Maschinenmodelle läßt sich die zugehörige Klasse analytischer Funktionen definieren.
Vom Standpunkt der klassischen Berechenbarkeitstheorie aus bietet es sich im Zusammenhang mit den erwähnten Potenzreihen an, solche analytischen Funktionen als berechenbar zu bezeichnen, deren Potenzreihenentwicklung eine Turing-berechenbare Folge ist. Dieser Ansatz führt jedoch nicht sehr weit, gelten doch auf diese Weise schon Funktionen mit nichtrationalen Koeffizienten als nicht mehr berechenbar. Diese Funktionen sind auch nicht unter Komposition abgeschlossen, wie das folgende einfache Beispiel zeigt: Die Funktion e^z ist in obigem Sinne berechenbar, denn die Koeffizientenfolge $(\frac{1}{k!})_{k\in\mathbb{N}}$ ist Turing-berechenbar, wenn man den Begriff der Turing-Berechenbarkeit auf natürliche Weise auf die rationalen Zahlen erweitert. Die Komposition der Funktion mit sich selbst, e^{e^z}, hat an der Stelle 0 den Wert e, hat also keinen rationalen nullten Koeffizienten in der Potenzreihenentwicklung.
Eine weitere Möglichkeit der Definition berechenbarer analytischer Funktion besteht darin, auf den natürlichen Zahlen gegebene klassisch berechenbare Funktionen zu analytischen Funktionen auf einem möglichst großen Teilbereich der komplexen Zahlenebene fortzusetzen. Ein Beispiel hierfür ist die bekannte Γ-Funktion, die die primitiv-rekursive Funktion der Fakultät, $n!$, fortsetzt. Auf diese Möglichkeit der Definition wird in Kapitel 4 kurz eingegangen.

Anmerkung. Zum Sprachgebrauch: Für die komplex-analytischen Funktionen gibt es mehrere unterschiedliche Bezeichnungen wie etwa holomorphe Funktionen, komplex differenzierbare Funktionen und analytische Funktionen. Diese Begriffe kommen von den unterschiedlichen äquivalenten Möglichkeiten, diese Funktionen zu definieren. Wir verwenden hier zumeist den Begriff analytische Funktion. Dabei wird das Wort "analytisch" doppelt verwendet, da es ja auch Maschinen und Berechnungen charakterisiert. Dennoch verwenden wir den Begriff und nicht etwa holomorph, weil er sofort an die Potenzreihenentwicklung erinnert, die im Folgenden im Vordergrund steht.
Weiter wollen wir anmerken, daß wir in diesem Kapitel stets Funktionen betrachten, die auf ihrem (zusammenhängenden) gesamten Definitionsbereich eine analytische Funktion darstellen. Mit Hilfe von $\mathbb{C}$-Maschinen lassen sich durch Verzweigungsbefehle natürlich auch *zusammengesetzte analytische Funktionen* definieren, also Funktionen, deren Definitionsbereich sich in semi-algebraische Mengen aufspaltet, auf denen die Funktion dann eine jeweils eigene analytische Funktion darstellt. Diese zusammengesetzten Funktionen sind aber nicht Gegenstand dieses Kapitels.

Wir betrachten im folgenden Funktionen, die auf *Gebieten*, also offenen zusammenhängenden Teilmengen von $\mathbb{C}$, definiert sind. Für analytische Funktionen bieten sich auf natürliche Weise auch andere Definitionsmengen an, namentlich die *Riemannschen Flächen*. Auf diese allgemeinere Auffassung der analytischen Funktionen werden wir in dieser Arbeit nur am Rande eingehen, und uns auf Teilmengen der komplexen Ebene beschränken. Wenn wir von einer analytischen Funktion auf einem Gebiet D und ihrer analytischen Fortsetzung auf ein Gebiet $G \supseteq D$ sprechen, meinen wir damit immer einen konkreten Zweig dieser analytischen Funktion.
Im Verlauf dieses Kapitels werden zunächst die berechenbaren analytischen Funktionen definiert. Im ersten Kapitel wurden verschiedene unterschiedlich mächtige Maschinenmodelle definiert. Man kann nun für jedes dieser Maschinenmodelle eine entsprechende Klasse berechenbarer analytischer Funktionen definieren, und diese analog zum Hierarchiesatz miteinander in Beziehung setzen. Dies werden wir am Ende des Kapitels

auch tun, das Hauptaugenmerk liegt jedoch nicht darin, jede Klasse einzeln zu untersuchen, sondern die Konzentration auf die durch allgemeine analytische Maschinen berechenbaren bzw. koeffizientenberechenbaren analytischen Funktionen zu lenken. Bei der Betrachtung der allgemeinen berechenbaren Funktionen ergibt sich für diese Klasse mit der Nichtabgeschlossenheit unter Komposition eine nicht sehr befriedigende Eigenschaft, die unter anderem erst zur Untersuchung der anderen Klassen geführt hat. Wir werden sehen, daß dies bei analytischen Funktionen im wesentlichen nicht mehr der Fall ist, sondern daß die koeffizientenberechenbaren analytischen Funktionen unter Komposition abgeschlossen sind. Der Hierarchiesatz, der für generelle analytisch berechenbare Funktionen gilt, bricht also unter der Zusatzvoraussetzung der komplexen Differenzierbarkeit zusammen.
Wir werden dann weitere Eigenschaften berechenbarer analytischer Funktionen untersuchen und sehen, daß diese neben der Komposition weitere Abschlußeigenschaften besitzen. Wenn eine analytische Funktion auf einem Gebiet definiert ist, und die Potenzreihe dieser Funktion in einem Punkte über das Gebiet hinaus konvergiert, läßt sich diese Funktion über das Gebiet hinaus analytisch fortsetzen. Wir werden sehen, daß dies auch für die Berechenbarkeit gilt. Wenn also eine analytische Funktion in einem Gebiet D berechenbar ist, und für diese eine Fortsetzung auf dem Gebiet $G \supseteq D$ existiert, dann gibt es für jeden Punkt aus G eine Umgebung, in der die Funktion berechenbar ist. Weiter werden wir sehen, daß auch die lokale Umkehrfunktion einer berechenbaren analytischen Funktion ebenfalls lokal berechenbar ist.

3.1.1 Endlich $\mathbb{C}$-berechenbare Funktionen

Zunächst betrachten wir jene Funktionen, welche durch endliche $\mathbb{C}$-Maschinen berechnet werden können. Die endliche Berechenbarkeit ist jedoch nicht der natürliche Berechenbarkeitsbegriff für analytische Funktionen. Intuitiv erwartet man, daß es nicht möglich ist, selbst einfache transzendente analytische Funktionen oder sogar nichtrationale algebraische Funktionen mit nur endlich vielen Rechenschritten berechnen zu können. Diese Intuition wird auch bestätigt:

Satz 3.1.1. *Sei $f : D \to \mathbb{C}$ eine auf dem Gebiet D analytische Funktion und sei f auf D $\mathbb{C}$-berechenbar. Dann stellt f eine auf D rationale Funktion dar.*

Beweis. Nach dem Darstellungssatz 2.2.22 für $\mathbb{C}$-berechenbare Funktionen ist

$$D = \bigcup_{\sigma} D_\sigma,$$

wobei die abzählbar vielen D_σ die semi-algebraischen Einzugsbereiche der Berechnungspfade sind. Nach Lemma 2.2.24 muß wenigstens ein an dieser Vereinigung beteiligtes D_{σ_0} ein nichtleeres Inneres haben. Nach dem Darstellungssatz ist f auf D_{σ_0} eine rationale Funktion. Es ist also $f|_{D_{\sigma_0}} = r|_{D_{\sigma_0}}$ für eine rationale Funktion r. Da D_{σ_0} nichtleeres Inneres hat, folgt aus dem Identitätssatz für analytische Funktionen bereits $f = r$ auf ganz D. □

Die endlich berechenbaren analytischen Funktionen werden also durch die rationalen Funktionen abgedeckt. Ist eine rationale Funktion kein Polynom, so hat sie in der Regel dennoch eine unendliche Potenzreihenentwicklung. Wir zeigen, daß für eine $\mathbb{C}$-berechenbare analytische Funktion die Folge der Koeffizienten der Potenzreihenentwicklung stets eine $\mathbb{C}$-berechenbare Folge bildet.

Satz 3.1.2.

1. *Sei f auf dem Gebiet D $\mathbb{C}$-berechenbar und analytisch. Dann ist für $z_0 \in D$ die Potenzreihenentwicklung von f um z_0 eine $\mathbb{C}[C]$-berechenbare Folge. Dabei ist die Konstantenmenge C endlich.*

2. *Die Berechenbarkeit unter 1. ist konstruktiv, wenn auf Real- und Imaginärteil der Register zugegriffen werden kann, d.h. es gibt eine $\mathbb{C}$-Maschine mit der zusätzlichen Operation der Konjugation $\alpha := \overline{\alpha}$, die bei Eingabe einer Maschine für f, eines Punktes $z_0 \in D$ und eines Indexes n den n-ten Koeffizienten der Potenzreihenentwicklung von f um z_0 berechnet.*

Beweis.

1. Es sei $f(z) = \sum_{k=0}^{\infty} a_k (z-z_0)^k$. Nach dem Darstellungssatz 2.2.22 unterteilt sich D in abzählbar viele semi-algebraische Mengen, auf denen f rational ist. Nach Lemma 2.2.24 muß wenigstens eine dieser Mengen ein nichtleeres Inneres haben, f ist also auf einer offenen Teilmenge von D rational, und somit wegen des Zusammenhangs von D nach dem Identitätssatz für analytische Funktionen auf ganz D rational. Sei $f(z) = \frac{p(z)}{q(z)}$ mit Polynomen p und q. Durch Entwicklung von p und q um z_0 erhalten wir Polynome $\tilde{p}$ und $\tilde{q}$ mit $f(z) = \frac{\tilde{p}(z-z_0)}{\tilde{q}(z-z_0)}$. Die Maschine $\mathcal{M}_a$, die die Folge $(a_k)_{k\in\mathbb{N}}$ berechnet, hat als Konstanten die Koeffizienten von $\tilde{p}$ und $\tilde{q}$ gespeichert. Bei Eingabe von n berechnet sie nun symbolisch die n-te Ableitung von $f(z) = \frac{\tilde{p}(z-z_0)}{\tilde{q}(z-z_0)}$ und wertet diese an der Stelle z_0 aus. Als Ergebnis wird dann $a_n = \frac{f^{(n)}(z_0)}{n!}$ ausgegeben.

2. Die Maschine, die bei Eingabe einer Maschine $\mathcal{M}_f$ für f, eines Punktes z_0 und eines Indexes n den n-ten Koeffizienten von f bei z_0 berechnet, im folgenden $\mathcal{K}$, bestimmt zunächst symbolisch die Koeffizienten der rationalen Funktion, die f darstellt. Dazu muß $\mathcal{K}$ die Koeffizienten der von $\mathcal{M}_f$ berechneten rationalen Funktion entlang eines Pfades σ berechnen, für den D_σ eine offene nichtleere Menge ist. Um einen solchen Pfad zu finden, werden alle Berechnungspfade der Maschine $\mathcal{M}$ parallel simuliert. Dabei werden in der Simulation stets die Pfade verworfen, die kein offenes nichtleeres Inneres haben können, wie Pfade, die durch $= 0$ Verzweigungen entstehen. In solchen Fällen wird immer der $\neq$-Pfad ausgewählt. Bei Verzweigungen der Form $|\alpha| \geq |z_i|$ wird mit Methoden von Lemma 2.4.5 überprüft, ob einer der beiden Fälle $\alpha\overline{\alpha} - z_i\overline{z_i} > 0$ bzw. $\alpha\overline{\alpha} - z_i\overline{z_i} < 0$ eine nichtleere Lösungsmenge hat. Die Pfade mit nichtleerer Lösungsmenge werden weiter simuliert. Findet $\mathcal{K}$ auf diese Weise einen haltenden Pfad σ, ist die entsprechende Menge D_σ eine offene Menge, da sie durch den endlichen Durchschnitt nichtleerer offener Mengen definiert werden kann. Die symbolisch berechneten Koeffizienten der rationalen

Funktion sind also die der Funktion $f(z) = \frac{p(z)}{q(z)}$. $\mathcal{K}$ bestimmt durch Entwicklung die Koeffizienten der Polynome $\tilde{p}$ und $\tilde{q}$ mit $f(z) = \frac{\tilde{p}(z-z_0)}{\tilde{q}(z-z_0)}$ und berechnet wie im ersten Fall den Koeffizienten a_n der Reihenentwicklung von f um z_0.

□

Anmerkung.

1. Die Idee der Berechnung der Koeffizienten besteht hier also darin, die Koeffizienten der rationalen Funktion zu berechnen, die durch die Maschine an einem Berechnungspfad σ berechnet wird. Ist ein Berechnungspfad bekannt, der einen offenen, nichtleeren Einzugsbereich hat, so sind die Koeffizienten der Potenzreihenentwicklung stets berechenbar.
2. Die Einschränkung, daß die Maschine $\mathcal{K}$ im zweiten Teil des Satzes über die Operation der Konjugation verfügt, ist wesentlich, da diese Maschine somit nicht auf holomorphe arithmetische Operationen beschränkt ist.

3.1.2 Berechenbarkeit und Koeffizientenberechenbarkeit

Im folgenden werden wir als berechenbare analytische Funktionen solche ansehen, die mit einer analytischen Maschine berechenbar sind. Wir werden zwar auch solche analytische Funktionen betrachten, die mit Hilfe (quasi-)stark δ-$\mathbb{Q}$-analytischer Maschinen berechenbar sind, aber das Hauptaugenmerk des Abschnitts liegt auf den durch allgemeine analytische Maschinen berechenbaren Funktionen.
Wir unterscheiden *berechenbare* analytische Funktionen und *koeffizientenberechenbare* analytische Funktionen, also einmal jene Funktionen, die als Funktionen mittels analytischer Maschinen berechenbar sind, und zum anderen solche, deren Koeffizientenfolge analytisch berechenbar sind.
Zunächst gehen wir von analytischen Funktionen auf einem Gebiet $D \subseteq \mathbb{C}$ aus. Das heißt, wir setzen voraus, daß die betrachteten Funktionen auf ganz D ohne Singularitäten analytisch sind.

Definition 3.1.3. *Sei $D \subseteq \mathbb{C}$ ein Gebiet, und sei $f : D \to \mathbb{C}$ eine analytische Funktion. f heißt*

- *eine* auf D (analytisch) berechenbare analytische Funktion*, wenn f auf D mit einer analytischen Maschine berechenbar ist,*
- *in $z_0 \in D$* (analytisch) koeffizientenberechenbar*, wenn f um z_0 die Entwicklung $f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k$ hat und die Folge $(a_k)_{k\in\mathbb{N}}$ analytisch berechenbar ist,*
- *(auf D)* (analytisch) koeffizientenberechenbar*, wenn f in jedem $z_0 \in D$ analytisch koeffizientenberechenbar ist.*

Die Klasse der auf D berechenbaren analytischen Funktionen bezeichnen wir mit $\mathcal{O}_{\mathcal{A}}(D)$, und die Klasse der auf D (analytisch) koeffizientenberechenbaren Funktionen bezeichnen wir mit $\mathcal{O}_{\mathcal{K}}(D)$.
Analog werden auch die entsprechenden Klassen berechenbarer analytischer Funktionen für andere Maschinentypen definiert, wie etwa ((quasi-)stark) δ-$\mathbb{Q}$-analytisch berechenbare und koeffizientenberechenbare Funktionen, die wir entsprechend mit $\mathcal{O}_{\mathcal{A}}^{\delta\text{-}\mathbb{Q}}(D)$, bzw. $\mathcal{O}_{\mathcal{K}}^{\delta\text{-}\mathbb{Q}}(D)$ bezeichnen.

Anmerkung.

1. Da wir im folgenden hauptsächlich Maschinen mit unendlichen Berechnungen betrachten, werden wir den Zusatz "analytisch" bei der Berechenbarkeit meist weglassen. Wenn wir doch endliche Berechenbarkeit meinen, drücken wir das mit dem Zusatz "endlich" explizit aus.

2. Die Koeffizientenberechenbarkeit ist ein lokaler Begriff, d.h. die Koeffizientenberechenbarkeit auf einer Menge wird über jene der einzelnen Punkte der Menge definiert. Die einzelnen Maschinen, die die Koeffizienten in den jeweiligen Punkten berechnen, müssen a priori nichts miteinander zu tun haben. Eine stärkere Voraussetzung ist, daß eine Maschine die Koeffizienten eines Punktes bei Eingabe des Punktes berechnet. In diesem Fall sprechen wir von *uniformer* Berechenbarkeit.

3. Wie bei der Definition der Berechenbarkeit von Funktionen spielt auch die Menge der Konstanten, über die eine Maschine verfügen darf, eine Rolle. Wenn eine Menge C nichtrationaler Konstanten verwendet wird, sagen wir entsprechend $[C]$-(analytisch) berechenbar/koeffizientenberechenbar.

Welche analytischen Funktionen sind berechenbar bzw. koeffizientenberechenbar? Für die Beantwortung dieser Frage ist relevant, ob komplexe nichtrationale Konstanten in den Programmen zugelassen werden oder nicht.

Beispiel 3.1.4.

1. Jedes Polynom $p(z) = \sum_{k=0}^{n} a_k z^k$ ist $\mathbb{C}[a_1, \ldots, a_n]$-analytisch berechenbar.

2. Ist für jedes k die Zahl a_k analytisch berechenbar, so ist das Polynom $p(z) = \sum_{k=0}^{n} a_k z^k$ analytisch berechenbar (ohne Konstanten).

   Beweis. Eine Maschine zur Berechnung von $p(z)$ simuliert parallel die (endlich vielen) Maschinen zur Berechnung der Koeffizienten jeweils für N Schritte, berechnet die Approximation, gibt diese auf dem Ausgabeband aus und erhöht die Schrittanzahl N.

3. Für jede $\mathbb{C}$-berechenbare (und damit auch Turing-berechenbare) Folge $(a_k)_{k\in\mathbb{N}}$ ist die Potenzreihe $f(z) = \sum_{k=0}^{\infty} a_k z^k$ innerhalb ihres Konvergenzkreises berechenbar.

   Beweis. Eine Maschine zur Berechnung von $f(z)$ berechnet naiv für aufsteigende n die Werte a_n und gibt die Partialsumme $\sum_{k=0}^{n} a_k z^k$ auf dem Ausgabeband aus.

4. Es sei eine Aufzählung der Menge der analytischen Maschinen festgelegt, und sei $\mathcal{M}_n$ die n-te Maschine in dieser Aufzählung. Wir definieren

   $$a_n = \begin{cases} 1 & : \quad \mathcal{M}_n \text{ konvergiert auf Eingabe } 0 \\ 0 & : \quad \text{sonst} \end{cases}.$$

   Die *Konvergenzfunktion für analytische Maschinen* sei definiert als

   $$k_A(z) = \sum_{k=0}^{\infty} a_k z^k.$$

   Dann konvergiert die Reihe im Innern des Einheitskreises, aber k_A ist in 0 nicht koeffizientenberechenbar und in keiner Umgebung der 0 analytisch berechenbar.

Beweis. Aus der Unlösbarkeit des Konvergenzproblems für analytische Maschinen folgt, daß k_A nicht analytisch koeffizientenberechenbar ist. Um zu sehen, daß k_A auch nicht analytisch berechenbar ist, betrachten wir für festes n

$$k_A\left(\frac{1}{2^n}\right) = \sum_{k=0}^{\infty} a_k 2^{-nk}.$$

Wäre für ein n dieser Wert berechenbar, könnte das Konvergenzproblem gelöst werden: Soll für $\mathcal{M}$ die Konvergenz entschieden werden, so wird zunächst j bestimmt mit $\mathcal{M}_j = \mathcal{M}$. Dann wird die Maschine zur Berechnung von k_A auf der Eingabe $\frac{1}{2^n}$ simuliert. Bei jeder Ausgabe dieser simulierten Maschine wird der Koeffizient für 2^{-nj} der Binärdarstellung dieser Ausgabe bestimmt und ausgegeben. Dieser Koeffizient wird bei hinreichender Genauigkeit der Berechnung von $k_A(\frac{1}{2^n})$ nicht mehr verändert, da die Folge keiner periodischen Dualentwicklung entsprechen kann, und er entspricht der Lösung des Konvergenzproblems für $\mathcal{M}_j$.
Damit ist also $k_A(2^{-n})$ für kein $n \in \mathbb{N}$ analytisch berechenbar, und somit kann die Funktion k_A in keiner Umgebung der 0 analytisch berechenbar sein.

5. Wir definieren analog die *Konvergenzfunktion für Turing-Maschinen*, indem wir $t_k = 1$ bzw. $t_k = 0$ setzen, je nachdem, ob die k-te Turing-Maschine in einer effektiven Aufzählung hält oder nicht. Wir definieren

$$k_T(z) = \sum_{k=0}^{\infty} t_k z^k.$$

Dann ist k_t im Innern des Einheitskreises analytisch koeffizientenberechenbar und analytisch berechenbar.

Beweis. Die Koeffizientenberechenbarkeit folgt unmittelbar aus der Lösbarkeit des Halteproblems für Turing-Maschinen durch analytische Maschinen.
Die Berechnung von $k_T(z)$ für $|z| < 1$ mittels einer analytischen Maschine geschieht durch parallele Approximation aller t_k, indem für wachsende n jeweils n Schritte der Turing-Maschinen $t_1, \ldots, t_n$ simuliert werden. Sobald eine der simulierten Maschinen hält, wird der jeweilige Wert für t_k mit 1 festgelegt, andernfalls als 0 angenommen. Für jedes n wird dann die Partialsumme $\sum_{k=0}^{n} t_{k,n} z^n$ ausgegeben, wobei $t_{k,n}$ die Approximation von t_k bei Schritt n ist.

3.2 Eigenschaften berechenbarer analytischer Funktionen

Wir haben zwei unterschiedliche Berechenbarkeitsbegriffe für analytische Funktionen eingeführt: Den der Berechenbarkeit, der unmittelbar an der Berechenbarkeit der Funktion orientiert ist, und den der Koeffizientenberechenbarkeit. Im Laufe des Abschnitts untersuchen wir die Eigenschaften der beiden Funktionsklassen und die Beziehungen der Klassen zueinander.

3.2.1 Koeffizientenberechenbarkeit analytischer Funktionen

Zunächst untersuchen wir die koeffizientenberechenbaren analytischen Funktionen, denn wie sich zeigt, ist dies der stärkere der beiden Begriffe. Aus der Funktionentheorie sind folgende Beziehungen bekannt:

Lemma 3.2.1 (Cauchysche Ungleichungen)**.** *Sei f auf D holomorph, $z_0 \in D$ und sei der abgeschlossene Kreis $\overline{U}_r(z_0) \subseteq D$. Dann gilt*

$$|f^{(n)}(z_0)| \leq \frac{n!}{r^n} \|f(\zeta)\|_{|\zeta - z_0| = r}$$

Beweis. Wohlbekannt, siehe etwa [FL94]. □

Korollar 3.2.2. *Sei f auf D holomorph, $z_0 \in D$ und sei der abgeschlossene Kreis $\overline{U}_r(z_0) \subseteq D$. Weiter sei $(a_k)_{k\in\mathbb{N}}$ die Folge der Koeffizienten der Potenzreihenentwicklung von f in z_0. Dann gibt es eine Konstante M, so daß für die Koeffizienten a_k gilt:*

$$|a_k| \leq \frac{M}{r^k}$$

Beweis. Dies folgt unmittelbar aus den Cauchyschen Ungleichungen zusammen mit der Beobachtung, daß

$$f^{(k)}(z_0) = k! a_k$$

□

Wir zeigen nun, daß die Berechenbarkeit der Koeffizientenfolge in einem Punkt die Berechenbarkeit der Funktion in einer Umgebung dieses Punktes zur Folge hat.

Satz 3.2.3 (Berechenbarkeit der Summe einer Potenzreihe)**.** *Die Funktion f sei auf dem Gebiet D analytisch, es sei $z_0 \in D$ und es gelte die Potenzreihenentwicklung $f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k$. Die Folge $(a_k)_{k\in\mathbb{N}}$ sei analytisch berechenbar. Es sei nun $R > 0$ so, daß der abgeschlossene Kreis mit Radius R um z_0 noch ganz in D liegt, also $\overline{U}_R(z_0) \subseteq D$. Dann ist die Funktion f in $U_R(z_0) \subseteq D$ analytisch berechenbar.*

Beweis. Wir geben eine Maschine $\mathcal{M}$ an, die f in $U_R(z_0)$ berechnet. Nach Voraussetzung ist die Koeffizientenfolge $(a_k)_{k\in\mathbb{N}}$ analytisch berechenbar; sei $\mathcal{K}$ eine analytische Maschine, die diese berechnet. Wir bezeichnen die n-te Approximation der Berechnung von a_k durch $\mathcal{K}$ mit $a_k^{(n)}$. Die Maschine $\mathcal{M}$ berechnet für ihre Eingabe z nun schrittweise immer genauere Approximationen des Funktionswertes $f(z)$. Dazu simuliert sie in

Schritt n die Maschine $\mathcal{K}$ für jedes $k = 1, \ldots, n$ für n Schritte und berechnet also für die ersten n Koeffizienten die Approximationen $a_k^{(n)}$. Die Summe

$$\sum_{k=0}^{n} a_k^{(n)}(z - z_0)^k$$

ist nun eine Approximation von $f(z)$. Da über die Güte der Approximation von $a_k^{(n)}$ jedoch nichts bekannt ist, reicht es nicht aus, diese Summe als Zwischenergebnis auszugeben. Durch die Cauchyschen Ungleichungen ist aber eine Abschätzung für die a_k gegeben. Sei M die Konstante aus den Cauchyschen Ungleichungen 3.2.1, für die $|a_k| \leq \frac{M}{R^k}$ gilt. Die Maschine $\mathcal{M}$ hat $\frac{M}{R^k}$ als Konstante gespeichert. Falls nun die n-te Approximation $a_k^{(n)}$ nicht diese Ungleichung erfüllt, so ersetzt $\mathcal{M}$ diese einfach durch $\frac{M}{R^k}$. Die n-te Ausgabe von $\mathcal{M}$ ist also

$$\sum_{k=0}^{n} b_k^{(n)}(z - z_0)^k \quad \text{mit} \quad b_k^{(n)} = \begin{cases} a_k^{(n)} & : \quad |a_k^{(n)}| \leq \frac{M}{R^k} \\ \frac{M}{R^k} & : \quad \text{sonst} \end{cases}$$

Wir behaupten, daß $\sum_{k=0}^{n} b_k^{(n)}(z - z_0)^k \to f(z)$ für $n \to \infty$ gilt.
Dazu sei $\varepsilon > 0$. Für $n_0 > 0$ gilt (der Einfachheit halber sei im folgenden $q := \frac{z - z_0}{R}$):

$$\sum_{k=n_0}^{\infty} |a_k(z - z_0)^k| \leq \sum_{k=n_0}^{\infty} \frac{M}{R^k}|z - z_0|^k = M \left| \frac{q^{n_0}}{1 - q} \right|.$$

Es sei n_0 so groß fixiert, daß $M|\frac{q^{n_0}}{1-q}| < \varepsilon$ (dies existiert, da $|z - z_0| < R$ und somit $q < 1$). Da die Maschine $\mathcal{K}$ die Werte $a_k^{(n)}$ analytisch berechnet, gibt es ein N_0, so daß für alle $k \leq n_0$ gilt: $|a_k - a_k^{(n)}| < \frac{\varepsilon}{|z - z_0|^k(n_0+1)}$ für alle $n \geq N_0$. Nach Wahl von $b_k^{(n)}$ gilt stets $b_k^{(n)} \leq \frac{M}{R^k}$. Da diese Beziehung auch für a_k gilt, ergibt sich für alle $n \geq N_0$

$$|a_k - b_k^{(n)}| \leq \begin{cases} \frac{\varepsilon}{|z - z_0|^k(n_0+1)} & : \quad 0 \leq k \leq n_0 \\ 2\frac{M}{R^k} & : \quad \text{in jedem Fall} \end{cases}$$

Für die n-te Approximation von $\mathcal{M}$ auf Eingabe z ergibt sich nun für $n \geq N_0$

$$\begin{aligned} |f(z) - \mathcal{M}^{(n)}(z)| &= \left| f(z) - \sum_{k=0}^{n} b_k^{(n)}(z - z_0)^k \right| \\ &\leq \sum_{k=0}^{n_0} |(a_k - b_k^{(n)})(z - z_0)^k| + \sum_{k=n_0}^{n} |(a_k - b_k^{(n)})(z - z_0)^k| + \\ &\qquad + \sum_{k=n}^{\infty} |a_k(z - z_0)^k| \end{aligned}$$

Hierbei wurde die Summe aufgespalten in drei Teile: Im ersten Teil bis n_0 ist die Approximationsgüte der $a_k^{(n)}$ bekannt. Im zweiten Teil bis n werden all jene Koeffizienten berücksichtigt, für die $\mathcal{M}$ bereits Approximationen berechnet, aber diese gegebenenfalls mit Hilfe der Cauchyschen Ungleichungen abschätzt. Im dritten Teil schließlich werden

die verbleibenden Summanden geführt, die $\mathcal{M}$ noch gar nicht in der Summe berücksichtigt.
Es ergibt sich nun, wenn wir zusätzlich N_0 so groß wählen, daß $N_0 > n_0$ und somit neben $n > N_0$ auch $n > n_0$ gilt,

$$\begin{aligned} |f(z) - \mathcal{M}^{(n)}(z)| &\leq \sum_{k=0}^{n_0} \frac{\varepsilon}{n_0+1} + \sum_{k=n_0}^{n} 2M\frac{|z-z_0|^k}{R^k} + \sum_{k=n}^{\infty} M\frac{|z-z_0|^k}{R^k} \\ &\leq \varepsilon + 3\varepsilon = 4\varepsilon \end{aligned}$$

□

Anmerkung. Dieser Satz ergibt die Berechenbarkeit einer Funktion, indem die Existenz einer sie berechnenden Maschine gezeigt wird. Diese Berechenbarkeit ist jedoch nicht *konstruktiv*. Dazu müßte eine Maschine angegeben werden, die bei Eingabe eines Punktes z und einer Maschine, die eine Koeffizientenfolge berechnet, den Wert der Funktion an der Stelle z berechnet.
Der einzige nichtkonstruktive Aspekt des Beweises ist im übrigen, daß die Maschine über die Konstante $\frac{M}{R^k}$ verfügt. Diese kann zudem durch Vergrößerung stets rational gewählt werden, es werden also keine nichtrationalen Konstanten verwendet.

Korollar 3.2.4. *Für eine analytische Funktion $f : D \to \mathbb{C}$ gilt: "analytisch koeffizientenberechenbar $\implies$ analytisch berechenbar", d.h. $\mathcal{O}_{\mathcal{A}}(D) \subseteq \mathcal{O}_{\mathcal{K}}(D)$.*

Satz 3.2.3 läßt sich sogar noch verschärfen. Wir werden zeigen, daß bei analytischer Berechenbarkeit der Koeffizienten sogar nicht nur die Berechenbarkeit der Funktion in einem Konvergenzkreis gilt, sondern sogar die Berechenbarkeit aller Ableitungen der Funktion. Tatsächlich ist dann nicht nur jede einzelne Ableitung analytisch berechenbar, sondern die Gesamtheit der Ableitungen der Funktion, die Ableitungen sind also *uniform*, d.h. mit einer einzigen Maschine, berechenbar.
Zunächst sieht man leicht ein, daß mit der Berechenbarkeit der Koeffizientenfolge einer Funktion in einem Punkt auch die Koeffizientenfolgen der Ableitungen in diesem Punkt uniform analytisch berechenbar sind.
Wir erinnern für das Folgende noch einmal kurz an die Schreibweise $(k)_d := \frac{k!}{(k-d)!} = k \cdot (k-1) \cdots (k-d+1)$. Wenn die Fakultät durch die Gammafunktion ersetzt wird, kann k auch eine beliebige komplexe Zahl sein (solange k bzw. $k - d \notin -\mathbb{N}$).

Lemma 3.2.5. *Sei f auf D analytisch, $z_0 \in D$ und sei $(a_k)_{k\in\mathbb{N}}$, die Koeffizientenfolge der Potenzreihenentwicklung von f, um z_0 analytisch berechenbar. Die d-te Ableitung von f habe die Potenzreihendarstellung $f^{(d)}(z) = \sum_{k=0}^{\infty} a_{(d,k)}(z-z_0)^k$. Dann ist die Abbildung $(d,k) \mapsto a_{(d,k)}$ analytisch berechenbar.*

Beweis. Durch sukzessives Ableiten sieht man, daß $a_{(d,k)} = (k+d)_d a_{k+d}$ gilt. Mit der analytischen Berechenbarkeit von $(a_k)_{k\in\mathbb{N}}$ folgt sofort auch die Behauptung. □

Satz 3.2.6. *Die Funktion f sei auf dem Gebiet D analytisch, $z_0 \in D$, und es gelte die Potenzreihenentwicklung $f(z) = \sum_{k=0}^{\infty} a_k(z-z_0)^k$. Die Folge $(a_k)_{k\in\mathbb{N}}$ sei analytisch berechenbar. Es sei nun $R > 0$ so, daß der abgeschlossene Kreis mit Radius R um z_0 noch ganz in D liegt, also $\overline{U}_R(z_0) \subseteq D$. Dann sind die Ableitungen der Funktion f in $U_R(z_0) \subseteq D$ uniform analytisch berechenbar, es ist also die Funktion $(d,z) \mapsto f^{(d)}(z)$ auf $\mathbb{N} \times U_R(z_0)$ analytisch berechenbar.*

Beweis. Der Beweis verläuft sehr ähnlich zum Beweis von Satz 3.2.3. Sei $\mathcal{K}$ eine Maschine, die $(a_k)_k$ berechnet. Die Maschine $\mathcal{M}$ erhält als Eingabe d und z und berechnet analytisch $f^{(n)}(z)$. Im Unterschied zu diesem Satz muß nicht für alle k der k-te Koeffizient a_k von f approximiert werden, sondern für alle k der k-te Koeffizient der d-ten Ableitung $a_{d,k}$. Für diesen erhalten wir mit Hilfe der Cauchyschen Ungleichung die Abschätzung

$$|a_{d,k}| = (k+d)_d a_{k+d} \leq \frac{M}{R^{k+d}}(k+d)_d.$$

Sei $a^{(n)}_{d,k} = (k+d)_d a^{(n)}_{k+d}$ die n-te Approximation des k-ten Koeffizienten der d-ten Ableitung von f durch $\mathcal{K}$. Analog zum Beweis von Satz 3.2.3 approximiert die Maschine $\mathcal{M}$ die Potenzreihe von $f^{(d)}$ für wachsende n:

$$\sum_{k=0}^{n} a^{(n)}_{(d,k)}(z-z_0)^k.$$

Dabei wird $a^{(n)}_{(d,k)}$ durch $\frac{M}{R^{k+d}}(k+d)_d$ ersetzt, falls dieser Wert bei der Approximation überschritten wird, es wird also

$$\sum_{k=0}^{n} b^{(n)}_{d,k}(z-z_0)^k \quad \text{mit} \quad b^{(n)}_{k,d} = \begin{cases} a^{(n)}_{k,d} & : \ |a^{(n)}_{k,d}| \leq \frac{M}{R^k} \\ \frac{M}{R^{k+d}}(k+d)_d & : \ \text{sonst} \end{cases}$$

berechnet. Diese Berechnung approximiert nun für wachsende n den Wert $f^{(d)}(z)$. Wir betrachten nun für große n_0 die folgende Abschätzung, wobei wieder $q := \frac{z-z_0}{R}$:

$$\begin{aligned} \sum_{k=n_0}^{\infty} \frac{M}{R^{k+d}}(k+d)_d(z-z_0)^k &= \frac{M}{R^d} q^{n_0} \sum_{k=0}^{\infty} (k+n_0+d)_d q^k \\ &= \frac{M}{R^d} q^{n_0} \sum_{k=0}^{\infty} n_0^d \left(\frac{k+d}{n_0}+1\right)_d q^k \end{aligned}$$

Nun gilt $\frac{k+d}{n_0} + 1 < k+d$, sobald nur $k \geq 2$ und $n_0 > 2$. Addieren wir für die Fälle $k = 0, 1$ die Konstante c, und benutzen schließlich die d-te Ableitung der Identität $\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$, so erhalten wir

$$\begin{aligned} \sum_{k=n_0}^{\infty} \frac{M}{R^{k+d}}(k+d)_d(z-z_0)^k &\leq \frac{M}{R^d} q^{n_0} \left(c + \sum_{k=0}^{\infty} n_0^d (k+d)_d q^k \right) \\ &= \frac{M}{R^d} q^{n_0} n_0^d \left(\frac{d!}{(1-q)^{d+1}} + c n_0^{-d} \right) \end{aligned}$$

Offenbar wird für hinreichend großes n_0 der letzte Term bei festem d und $q < 1$ beliebig klein. Mit Hilfe dieser Abschätzung sieht man nun analog zum Beweis von Satz 3.2.3:

$$\sum_{k=0}^{n} b^{(n)}_{(d,k)}(z-z_0)^k \underset{n\to\infty}{\longrightarrow} f^{(d)}(z).$$

□

Ist eine Funktion in einem Punkt koeffizientenberechenbar, dann ist also die Gesamtheit der Ableitungen in einem Konvergenzkreis um diesen Punkt berechenbar. Da die Koeffizienten einer Potenzreihe um einen Punkt sich leicht mit Hilfe der Ableitungen der Funktion in diesem Punkt ausdrücken lassen, erhalten wir, daß die Funktion schon in *jedem* Punkt des Konvergenzkreises koeffizientenberechenbar ist.

Satz 3.2.7. *Die Funktion f sei auf dem Gebiet D analytisch, und es gebe ein $z_0 \in D$, so daß die Koeffizientenfolge der Potenzreihenentwicklung von f in diesem Punkt $(a_k)_{k\in\mathbb{N}}$ analytisch berechenbar ist. Sei weiter $\overline{U}_R(z_0) \subseteq D$. Dann ist die Folge der Koeffizienten der Potenzreihenentwicklung in jedem Punkt $z_1 \in U_R(z_0)$ analytisch berechenbar.*

Beweis. Es sei $z_1 \in U_R(z_0)$. Nach Satz 3.2.6 gibt es eine Maschine $\mathcal{M}$, die die Ableitungen von f in $U_R(z_0)$ berechnet. Für $z_1 \in U_R(z_0)$ wird nun bei Eingabe k die Maschine $\mathcal{M}$ auf z_1 und k angesetzt, diese berechnet analytisch $f^{(k)}(z_1)$. Bezeichnen wir mit $K_f(z,k)$ den k-ten Koeffizienten der Potenzreihenentwicklung von f um z, so gilt:

$$k!K_f(z,k) = f^{(k)}(z_1).$$

Die Maschine $\mathcal{M}$ angesetzt auf k und z_1 wird also simuliert, und bei jedem Ausgabeschritt wird die Ausgabe durch $k!$ dividiert. □

In diesem Beweis wird darüber hinaus ersichtlich, daß die Berechnung für alle z_1 von *einer einzigen* Maschine erfolgen kann, daß also analytisch koeffizientenberechenbare Funktionen immer schon uniform berechenbar sind:

Korollar 3.2.8. *In der Situation von Satz 3.2.7 gilt sogar:*
Die Funktion $U_R(z_0) \times \mathbb{N} \to \mathbb{C}$, $(z,k) \mapsto K_f(z,k)$ ist analytisch berechenbar. Dabei bezeichne $K_f(z,k)$ den k-ten Koeffizienten der Potenzreihenentwicklung von f.

In Satz 3.2.7 haben wir gezeigt, daß eine Funktion, die auf D analytisch ist und in einem Punkt $z_0 \in D$ koeffizientenberechenbar ist, bereits in allen Punkten in einem Kreis um z_0 koeffizientenberechenbar ist. Daraus folgt nun auch, daß f in *jedem* Punkt von D koeffizientenberechenbar ist:

Korollar 3.2.9. *Die Funktion f sei auf dem Gebiet D analytisch, und es gebe ein $z_0 \in D$, so daß f in z_0 koeffizientenberechenbar ist. Dann ist f auf ganz D koeffizientenberechenbar.*

Beweis. Sei $w \in D$ beliebig. Dann gibt es wegen des Zusammenhangs von D einen stetigen Pfad $\gamma : [0,1] \to D$ mit $\gamma(0) = z_0$ und $\gamma(1) = w$. Für jedes $t \in [0,1]$ gibt es ein $r_t > 0$, so daß $\overline{U}_{r_t}(\gamma(t)) \subseteq D$. Da $\gamma[0,1] \subseteq D$ kompakt ist, gibt es endlich viele $t_1, \ldots, t_n$, so daß

$$\gamma[0,1] \subseteq \bigcup_{1\le i\le n} U_{r_{t_i}}(t_i) \qquad \text{und } U_{r_{t_i}}(t_i) \cap U_{r_{t_{i+1}}}(t_{i+1}) \neq \emptyset \text{ für alle } i < n.$$

Durch Induktion folgt nun mit Satz 3.2.7, daß f in allen $U_{r_{t_i}}(t_i)$ koeffizientenberechenbar ist, insbesondere also in w. □

Wir haben also gezeigt, daß die Koeffizientenberechenbarkeit einer Funktion in einem Punkt grundlegend für die Berechenbarkeit analytischer Funktionen ist: Daraus folgt

bereits die Koeffizientenberechenbarkeit in jedem Punkte des betrachteten Definitionsgebietes und in Konvergenzkreisen die (uniforme) Berechenbarkeit aller Ableitungen der Funktion. Das wichtige Konzept der analytischen Funktionen, daß diese allein durch ihre Entwicklung in einem Punkt (Funktionskeim) vollständig festgelegt werden, überträgt sich also auch auf die Koeffizientenberechenbarkeit!

3.2.2 Abschlußeigenschaften koeffizientenberechenbarer Funktionen

Es hat sich gezeigt, daß koeffizientenberechenbare analytische Funktionen stets auch analytisch berechenbar sind, und daß Koeffizientenberechenbarkeit weitere Berechenbarkeitseigenschaften impliziert. Die allgemeinen analytisch berechenbaren Funktionen sind nicht unter Komposition abgeschlossen. Dagegen weisen die koeffizientenberechenbaren analytischen Funktionen, die ja über analytische Maschinen definiert sind, erstaunlich starke Abschlußeigenschaften auf.

Komposition

Mit Hilfe der Ergebnisse aus 3.2.1 können wir zeigen, daß die koeffizientenberechenbaren analytischen Funktionen unter Komposition abgeschlossen sind. Die Idee des Beweises liegt darin, die uniforme Berechenbarkeit der Ableitungen in jedem Punkt auszunutzen, um die Koeffizienten der Komposition zweier berechenbarer Potenzreihen zu berechnen.

Satz 3.2.10. *Sei $D \subseteq \mathbb{C}$ ein Gebiet und $g : D \to \mathbb{C}$ und $f : g(D) \to \mathbb{C}$ koeffizientenberechenbare Funktionen. Dann ist auch $f \circ g : D \to \mathbb{C}$ koeffizientenberechenbar.*

Beweis. Wir wählen ein $z_0 \in D$ und zeigen, daß $f \circ g$ in z_0 koeffizientenberechenbar ist. Es sei $z_1 := g(z_0)$, die Potenzreihendarstellungen von f und g in z_1 bzw. z_0 seien $f(z) = \sum_{k=0}^{\infty} a_k (z - z_1)^k$ und $g(z) = \sum_{k=0}^{\infty} b_k (z - z_0)^k$. Nach Voraussetzung gibt es Maschinen $\mathcal{M}_f$ und $\mathcal{M}_g$, welche die Folgen $(a_n)_n$ und $(b_n)_n$ analytisch berechnen. Damit kann mit diesen Maschinen auch jede Ableitung von f an der Stelle z_1 und jede Ableitung von g an der Stelle z_0 berechnet werden.
Betrachtet man die Ableitungen von $f \circ g$, so erhält man durch sukzessive Anwendungen von Ketten- und Produktregel einen Ausdruck für $(f \circ g)^{(n)}(z_0)$, in dem Ableitungen von f nur an der Stelle z_1 und Ableitungen von g nur an der Stelle z_0 berechnet werden. Ist $P_n = \{(k_1, \ldots, k_n) \,|\, k_1 + 2k_2 + \cdots + nk_n = n\}$ die Menge der Tupel, die die Partitionen von n beschreiben, dann gilt

$$(f \circ g)^{(n)}(z_0) = \sum_{(k_1,\ldots,k_n) \in P_n} \frac{n!}{k_1! \cdot \ldots \cdot k_n!} \left(f^{(k_1+\cdots+k_n)}\right)(g(z_0)) \prod_{m=1}^{n} \left(\frac{g^{(m)}(z_0)}{m!}\right)^{k_m}$$

Diese Formel kann man mit etwas Rechenaufwand durch Induktion zeigen. Sie ist als *Formel von Faà di Bruno* bekannt.
Um nun die Koeffizienten von $(f \circ g)(z) =: \sum_{k=0}^{\infty} c_k (z - z_0)^k$ zu berechnen, beachte man, daß $\left(f^{(k_1+\cdots+k_n)}\right)(g(z_0)) = (k_1 + \cdots + k_n)! a_{(k_1+\cdots+k_n)}$ und $g^{(m)}(z_0) = m! b_m$, und daß somit die Formel von Faà di Bruno ein endliches Polynom in den a_j und b_i ist. Somit ist die

n-te Ableitung von $f \circ g$ an der Stelle z_0 und damit auch die Potenzreihenentwicklung von $f \circ g$ an der Stelle z_0 analytisch berechenbar.
Damit ist gezeigt, daß $f \circ g$ in z_0 koeffizientenberechenbar ist, und da die Wahl für z_0 beliebig aus D war, ist f auf ganz D koeffizientenberechenbar. □

Lokale Umkehr

Hat eine analytische Funktion f in einem Punkt z_0 eine nichtverschwindende Ableitung, also $f'(z_0) \neq 0$, dann ist f in einer Umgebung von z_0 lokal umkehrbar und die Umkehrfunktion ist ebenfalls analytisch. Auf ähnliche Weise wie bei der Komposition zeigen wir, daß bei koeffizientenberechenbarer Funktion auch die lokale Umkehrfunktion koeffizientenberechenbar ist.

Satz 3.2.11. *Sei $D \subseteq \mathbb{C}$, $z_0 \in D$ und $f : D \to \mathbb{C}$ eine koeffizientenberechenbare analytische Funktionen mit $f'(z_0) \neq 0$. Dann ist die lokale Umkehrfunktion f^{-1} von f in $f(z_0)$ und somit auch in einer Umgebung von $f(z_0)$ koeffizientenberechenbar.*

Beweis. Sei $f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k$ mit der um $z_1 = f(z_0)$ lokalen Umkehrfunktion $g(z) = \sum_{k=0}^{\infty} b_k (z - z_1)^k$. Mit Hilfe der Kettenregel ergibt sich

$$\begin{aligned} g'(z_1) &= \frac{1}{f'(g(z_1))} = \frac{1}{f'(z_0)} \\ g''(z_1) &= \frac{f''(g(z_1))g'(z_1)}{f'(g(z_1))} = \frac{f''(z_0)g'(z_1)}{f'(z_0)} \\ &\vdots \end{aligned}$$

Durch Induktion stellt man fest: Die n-te Ableitung $g^{(n)}(z_1)$ berechnet sich durch einen endlichen arithmetischen Ausdruck durch die Ableitungen von f in z_0 der Ordnung 1 bis n und die Ableitungen von g in z_1 der Ordnung 1 bis $n-1$. Dieser Ausdruck kann symbolisch bestimmt werden, und damit können die Approximationen der Koeffizienten b_k aus den Approximationen der Koeffizienten a_k und b_j für $j < k$ bestimmt werden. □

Analytische Fortsetzung

Sei $f : D \to \mathbb{C}$ eine analytische Funktion auf dem Gebiet D. Sei $z_0 \in D$ und die Potenzreihenentwicklung von f um z_0 gegeben durch $f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k$. Dann ist es möglich, daß der Konvergenzkreis U dieser Potenzreihe sich über das Gebiet D hinaus erstreckt. Wenn man zusätzlich annimmt, daß $D \cap U$ zusammenhängend ist, kann man eine Funktion g auf $D \cup U$ wie folgt definieren:

$$g(z) = \begin{cases} f(z) & : \quad x \in D \\ \sum_{k=0}^{\infty} a_k (z - z_0)^k & : \quad x \in U \end{cases}$$

Dann ist g eine wohlefinierte analytische Funktion $g : D \cup U \to \mathbb{C}$ mit $g|_D = f$. Wir nennen g eine *unmittelbare analytische Fortsetzung* von f.
Läßt man die Annahme des Zusammenhangs von $D \cap U$ weg, so kann es passieren, daß die so definierte Funktion mehrdeutig wird. Führt man diesen Gedanken weiter, so

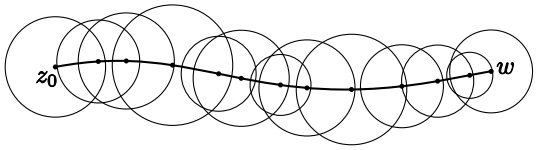


Abbildung 3.1: Analytische Fortsetzung entlang eines Pfades

kommt man zur Theorie der *Riemannschen Flächen*. Wir wollen uns hier aber darauf beschränken, nur eindeutige Fortsetzungen zu betrachten.
Ist nun ein stetiger Weg $\gamma : [0,1] \to \mathbb{C}$ gegeben mit $z_0 \in D$, $w \in C$, dann hat f eine *analytische Fortsetzung längs* γ, wenn es eine Folge von Punkten $z_0 = \gamma(0), \ldots, z_n = \gamma(1) = w$ auf γ und offene Kreise $D_0(z_0), \ldots, D_n(z_n)$ mit $z_{i+1} \in D_i(z_i)$, so daß $D_i \cap \bigcup_{j<i} D_j$ zusammenhängend ist und es analytische Funktionen $f = f_1, \ldots, f_n$ mit $f_i = f_{i+1}$ auf $D_i(z_i) \cap D_{i+1}(z_i + 1)$ gibt.
Wir nennen nun eine Funktion $g : G \to \mathbb{C}$ eine *analytische Fortsetzung* der Funktion $f : D \to \mathbb{C}$, wenn $G \supseteq D$ und $g|_D = f$. In diesem Fall existiert für jedes $w \in G$ dann stets ein stetiger Weg γ wie oben beschrieben, so daß f längs γ analytisch fortgesetzt werden kann und $g|_{D_n} = f_n$ mit den obigen Bezeichnungen gilt.
Mit Hilfe der Resultate aus Abschnitt 3.2.1 sieht man nun leicht: Ist eine analytische Funktion f auf einer Menge D koeffizientenberechenbar und ist g eine analytische Fortsetzung von f (hiermit ist immer ein konkreter Zweig gemeint), dann ist g ebenfalls koeffizientenberechenbar. Wir erhalten folgende Verallgemeinerung von Folgerung 3.2.9:

Satz 3.2.12. *Sei die analytische Funktion f auf D koeffizientenberechenbar und $G \supseteq D$ und $g : G \to \mathbb{C}$ eine analytische Fortsetzung von f. Dann ist g auf G koeffizientenberechenbar.*

Beweis. Es sei $z_0 \in D$, $w \in G$ und $\gamma : [0,1] \to \mathbb{C}$ ein stetiger Weg mit $\gamma(0) = z_0$, $\gamma(1) = w$. Dann gibt es eine Folge von Punkten $z_0 = \gamma(0), \ldots, z_n = \gamma(1) = w$ auf γ und offene Kreise $D_0(z_0), \ldots, D_n(z_n)$ mit $z_{i+1} \in D_i(z_i)$ und analytische Funktionen $f = f_1, \ldots, f_n$ mit $f_i = f_{i+1}$ auf $D_i(z_i) \cap D_{i+1}(z_i + 1)$ und $g|_{D_i} = f_i$ für alle i. Durch Induktion und Satz 3.2.7 folgt, daß jedes f_i in D_i koeffizientenberechenbar ist. Damit ist auch g in einer Umgebung von w koeffizientenberechenbar. □

Es stellt sich die Frage, ob diese Berechenbarkeit auch *konstruktiv* ist, d.h. ob eine Maschine bei Eingabe einer Maschine, die eine Funktion berechnet, und einer Maschine, die einen stetigen Weg berechnet, die analytische Fortsetzung der Funktion entlang des Weges berechnet. Im allgemeinen wird man auf diese Frage aber keine positive Antwort erwarten, da der Konvergenzradius einer Potenzreihe nicht berechenbar ist. Dies folgt daraus, daß der limes superior einer analytisch berechenbaren Folge im allgemeinen nicht berechenbar ist.

3.2.3 Berechenbare analytische Funktionen

Wir kehren nun zu den berechenbaren analytischen Funktionen zurück, bei denen nur die analytische Berechenbarkeit der Funktion, nicht aber der Koeffizienten der Potenzreihenentwicklung vorausgesetzt wird. Dabei untersuchen wir, ob eine Umkehrung von

Satz 3.2.3 gilt, ob also eine berechenbare Funktion bereits koeffizientenberechenbar ist? In diesem Fall würden auch alle Abschlußeigenschaften, die wir für koeffizientenberechenbare Funktionen gezeigt haben, für berechenbare analytische Funktionen gelten.
Der naive Ansatz, die Koeffizienten der Potenzreihenentwicklung zu berechnen, besteht in der sukzessiven Approximation der Differenzenquotienten zur Berechnung der Ableitungen der Funktion. Dieser Ansatz scheitert jedoch daran, daß etwa für die Ableitung

$$\lim_{\zeta \to 0} \frac{f(z+\zeta) - f(z)}{\zeta}$$

die Funktion f an unendlich vielen Stellen approximiert werden muß, die immer näher an z heranrücken. Dabei ist die Genauigkeit der Approximationen aber im allgemeinen nicht bekannt, und es kann somit die nötige Berechnungstiefe der einzelnen Approximationen von $f(z+\zeta)$ nicht bestimmt werden.
Auch der Ansatz, die Potenzreihe induktiv nach dem gesuchten Koeffizienten aufzulösen und diesen zu berechnen, scheitert an einem ähnlichen Problem: Will man in dem Ausdruck

$$a_n = \frac{1}{\delta^n} \left(f(\delta) - \sum_{k=0}^{n-1} a_k \delta^k \right) - \sum_{k=n+1}^{\infty} a_k \delta^{k-n}$$

a_n durch immer kleinere δ approximieren, ist wieder die nötige Berechnungstiefe von $f(\delta)$ nicht bekannt.
Da wir analytische Maschinen betrachten, bietet sich ein anderes Vorgehen an: Die Berechnung einer analytischen Maschine ergibt sich durch arithmetische Operationen, und die Zwischenergebnisse stellen rationale Funktionen dar, die gegen die berechnete Funktion konvergieren. Analog zu Satz 3.1.1 werden die rationalen Funktionen, die entlang der Berechnungspfade berechnet werden, symbolisch berechnet. Für analytische Maschinen, die nicht notwendig endliche Berechnungen haben, tritt nun das Problem auf, daß nicht mehr notwendigerweise ein Berechnungspfad σ mit einem offenen nichtleeren Einzugsbereich existieren muß. Tatsächlich kann es sogar sein, daß jeder Punkt zu einem eigenen Berechnungspfad führt. Allerdings ist offen, ob bei analytischen Funktionen nicht immer auch eine Maschine existiert, die keine solche Aufteilung des Definitionsbereichs erzeugt.
Wenn jedoch ein Pfad mit nichtleerem offenen Einzugsbereich existiert (zum Beispiel bei einer Maschine, bei der auf jedem Berechnungspfad nur endlich viele Verzweigungsbefehle mit äußeren Variablen), dann kann die Berechenbarkeit der Koeffizienten einer berechenbaren analytischen Funktion nachgewiesen werden:

Satz 3.2.13. *Sei f auf D analytisch und dort durch die Maschine $\mathcal{M}_f$ berechenbar. Es gebe einen Berechnungspfad σ von $\mathcal{M}_f$, so daß $D_\sigma \subseteq D$ nichtleeres Inneres hat. Dann ist f auf D koeffizientenberechenbar.*

Beweis. Sei $(f_n)_{n\in\mathbb{N}}$ die Folge der von $\mathcal{M}_f$ berechneten rationalen Funktionen entlang des Pfades σ. Diese konvergiert auf D_σ punktweise gegen f, also nach dem Satz von Osgood auf einem offenen, dichten Teilbereich D' von D_σ kompakt gleichmäßig gegen f. Es sei $z_0 \in D'$. Wir behaupten, daß f in z_0 koeffizientenberechenbar ist, und damit nach

den Sätzen über koeffizientenberechenbare Funktionen in ganz D. In einer Umgebung U von z_0 konvergieren die f_n gleichmäßig gegen f, und damit konvergieren nach dem Satz von Weierstraß auch die k-ten Ableitungen der Funktionen in dieser Umgebung gleichmäßig gegen die k-te Ableitung von f für alle $k \in \mathbb{N}$.
Die f_n sind als Zwischenergebnisse einer analytischen Berechnung rationale Funktionen. Durch Verfolgen des Berechnungspfades σ von $\mathcal{M}_f$ können die Koeffizienten der Zähler- und Nennerpolynome der rationalen Funktionen f_n (in endlicher Zeit) berechnet werden. Aus diesen Koeffizienten können mittels endlicher Berechnung wie in Satz 3.1.1 die Koeffizienten der Potenzreihen der f_n bei Entwicklung um z_0 berechnet werden. Bei gesuchtem k-ten Koeffizienten von f um z_0 muß nun die Folge der k-ten Koeffizienten der Funktionenfolge f_n um z_0 auf dem Ausgabeband ausgegeben werden. Wegen der gleichmäßigen Konvergenz der Ableitungen konvergieren diese Koeffizienten dann schon gegen den k-ten Koeffizienten von f. □

Vermutung 3.2.14. *Die Voraussetzungen von Satz 3.2.13 liegen bei jeder berechenbaren analytischen Funktion vor.*

Anmerkung. Im Falle, daß Vermutung 3.2.14 korrekt ist, bedeutet das, daß alle berechenbaren analytischen Funktionen auch koeffizientenberechenbar sind. Jede berechenbare analytische Funktion hat dann eine (nicht eindeutig bestimmte) *Normalform*, nämlich eine Maschine, die die Koeffizientenfolge der Potenzreihe der Funktion in einem ausgewählten Punkt berechnet.

Bis die Richtigkeit von Vermutung 3.2.14 entschieden ist, müssen wir uns mit Spezialfällen begnügen:

Satz 3.2.15. *Sei die analytische Funktion f auf D analytisch berechenbar und habe im Punkte $z_0 \in D$ eine Potenzreihenentwicklung mit Koeffizienten aus $\{0,1\}$. Dann ist f koeffizientenberechenbar.*

Beweis. Es genügt zu zeigen, daß f in z_0 koeffizientenberechenbar ist, und wir setzen oBdA voraus, daß $z_0 = 0$ gilt. Falls ab einem i_0 für alle $i \geq i_0$ stets gilt $a_i = 0$ bzw. für alle $i \geq i_0$ stets gilt $a_i = 1$, so ist die Folge $(a_i)_{i\in\mathbb{N}}$ sehr einfach analytisch berechenbar. Wir schließen diese beiden Fälle im folgenden aus. Es ist dann also für unendlich viele i der Koeffizient $a_i = 1$, und damit der Konvergenzradius der Reihe nach Cauchy-Hadamard gleich 1, und $\frac{1}{2}$ liegt im Konvergenzbereich der Reihe. Wir berechnen nun (analytisch)

$$f\left(\tfrac{1}{2}\right) = \sum_{k=0}^{\infty} a_k 2^{-k}.$$

Soll nun der Koeffizient a_i berechnet werden, so wird bei jeder neuen Approximation von $f\left(\frac{1}{2}\right)$ die Dualentwicklung der Approximation berechnet und der Koeffizient bei 2^{-i} ausgegeben. Nach Voraussetzung an die Koeffizienten wird bei der approximativen Berechnung von $f\left(\frac{1}{2}\right)$ jede Stelle in der Dualentwicklung nach endlicher Berechnungszeit stabil (da diese nicht konstant mit 0 oder 1 endet), und damit konvergiert die Ausgabe der Maschine gegen a_i. □

3.2.4 δ-$\mathbb{Q}$-berechenbare analytische Funktionen

Bisher haben wir analytische Funktionen untersucht, die von $\mathbb{C}$-analytischen Maschinen berechnet werden können bzw. deren Koeffizientenfolgen von diesen Maschinen berechnet werden. In diesem Abschnitt wollen wir kurz auf analytische Funktionen eingehen, die δ-$\mathbb{Q}$-berechenbar sind, oder deren Koeffizientenfolge δ-$\mathbb{Q}$-berechenbar ist. Dabei werden wieder nur robuste δ-$\mathbb{Q}$-Maschinen betrachtet, da der Einfluß der Rundung bei unseren Betrachtungen keine Rolle spielt. Mit δ-$\mathbb{Q}$-Maschinen sind in diesem Zusammenhang Maschinen gemeint, die mit rationalen komplexen Zahlen rechnen (also Maschinen über dem algebraischen Erweiterungskörper $\mathbb{Q}[i]$). Die Begriffe "berechenbare analytische Funktion" und "koeffizientenberechenbare anlytische Funktion" übertragen sich sinngemäß auf Funktionen, die durch δ-$\mathbb{Q}$-Maschinen berechnet werden.
Zunächst sieht man leicht ein, daß koeffizientenberechenbare analytische Funktionen, stets auch δ-$\mathbb{Q}$-koeffizientenberechenbar sind, da die Eingabe einer Maschine, die eine Koeffizientenfolge berechnet, ganzzahlig ist, und damit eine solche Maschine stets mit rationalen Zahlen rechnet (während die Ausgabe gegen allgemeine komplexe Zahlen konvergiert). Dabei setzen wir für den Rest des Abschnitts voraus, daß keine irrationalen Konstanten verwendet werden.

Satz 3.2.16. *Jede $\mathbb{C}$-koeffizientenberechenbare Funktion, die ohne irrationale Konstanten berechnet werden kann, ist δ-$\mathbb{Q}$-koeffizientenberechenbar.*

Untersucht man den Beweis von Satz 3.2.3, so zeigt sich mit einer leichten Abschwächung, daß jede $\mathbb{C}$-koeffizientenberechenbare Funktion schon δ-$\mathbb{Q}$ approximierbar ist!

Satz 3.2.17. *Jede $\mathbb{C}$-koeffizientenberechenbare Funktion ist δ-$\mathbb{Q}$-berechenbar.*

Beweis. Der Beweis ist eine Verfeinerung des Beweises von Satz 3.2.3, es werden daher nur die wesentlichen Änderungen zu diesem Beweis ausgeführt.
Sind a_k die Koeffizienten der Potenzreihenentwicklung von f um z_0, R und M die durch die Cauchyschen Ungleichungen gegebenen Konstanten mit $|a_k| \leq \frac{M}{R^k}$, dann wird bei der Berechnung von $f(z)$ für $|z - z_0| < R$ wieder ein approximierter Koeffizient $|a_k|$ durch $\frac{M}{R^k}$ ersetzt, sollte er größer als diese Zahl sein. Für wachsendes n wird die Summe

$$\sum_{k=0}^{n} b_k^{(n)} (z - z_0)^k$$

berechnet, wobei die b_k wie in Satz 3.2.3 definiert sind. Im Unterschied zu Satz 3.2.3 liegt nun die Eingabe z jedoch nicht mehr als exakte komplexe Zahl vor, sondern lediglich als gerundete Approximation $\rho(z, \delta)$ mit Genauigkeit δ. Diese Genauigkeit δ (repäsentiert durch eine negative Zweierpotenz) muß nun hinreichend hoch sein, so daß die Summe der Fehler immer noch klein bleibt. Im Schritt n, bei dem also die n-te Partialsumme approximiert wird, wird die Genauigkeit δ so weit erhöht, daß für alle $k = 1, \ldots, n$ gilt:

$$|(z - z_0) - \rho(z - z_0, \delta)|^k \leq 2^{-n} \frac{1}{(n+1)\frac{M}{R^k}}.$$

Dabei definiert man die Zahlen $\frac{M}{R^k}$, $b_k^{(n)}$ und n_0 wie im Beweis von Satz 3.2.3. Die Rundungsgenauigkeit, die nötig ist, ist $\mathbb{R}$-berechenbar, wir bezeichnen sie mit $\delta(n)$. Wir

erhalten

$$\begin{aligned}
|f(z) - \mathcal{M}^{(n)}(z)| &= \left| f(z) - \sum_{k=0}^{n} b_k^{(n)} \big(\rho(z - z_0, \delta(n))\big)^k \right| \\
&\leq \left| f(z) - \sum_{k=0}^{n} b_k^{(n)} (z - z_0)^n \right| + \sum_{k=0}^{n} \left| b_k^{(n)} \big((z - z_0) - \rho(z - z_0, \delta(n))\big)^k \right| \\
&\leq \left| f(z) - \sum_{k=0}^{n} b_k^{(n)} (z - z_0)^n \right| + 2^{-n}
\end{aligned}$$

Der Rest des Beweises verläuft analog zum Beweis von 3.2.3. □

Anmerkung. Im allgemeinen ist mit z_0 doch eine komplexe Konstante in der Maschine gespeichert. Dies kann jedoch umgangen werden, wenn man für z_0 eine rationale Zahl wählt, und da die rationalen (komplexen) Zahlen dicht liegen, ist dies auch kein Problem.

Vollkommen analog übertragen sich die Sätze 3.2.6 und 3.2.7 auf δ-$\mathbb{Q}$-Maschinen. Bezüglich Koeffizientenberechenbarkeit verhalten sich also analytische Maschinen und δ-$\mathbb{Q}$-Maschinen gleich.

Stark δ-$\mathbb{Q}$-analytisch und Typ-2 Turing-berechenbare analytische Funktionen

Bei den stark δ-$\mathbb{Q}$-analytischen Funktionen ist es eher als bei allgemeinen analytischen Maschinen zu erwarten, daß Berechenbarkeit einer Summe einer Potenzreihe oder Berechenbarkeit der Koeffizienten einer analytischen Funktion vorliegen. Durch die stets geltende Ausgabegenauigkeitsschranke lassen sich leicht Abschätzungen machen, so daß eine Maschine stets so lange rechnet, bis eine hinreichende Genauigkeit vorliegt. Ist etwa die Koeffizientenfolge einer Potenzreihe stark δ-$\mathbb{Q}$-analytisch berechenbar, so kann die Summe der Potenzreihe (unter Voraussetzung der Schranke aus Lemma 3.2.1) wieder stark δ-$\mathbb{Q}$-analytisch berechnet werden, indem die Partialsummen mit zunehmender Genauigkeit approximiert werden. Da die Approximationsgenauigkeit der einzelnen Koeffizienten bei stark δ-$\mathbb{Q}$-analytischen Maschinen bekannt ist, kann diese nun hinreichend hoch gewählt werden.
Die Klasse der stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen ist gleich der Klasse der Typ-2 Turing-berechenbaren Funktionen (vgl. Lemma 2.2.31), und daher gelten alle Resultate für die Typ-2 Turing-berechenbaren Funktionen (TT-berechebaren) auch für die stark δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen. Die mittels Typ-2 Turing-Maschinen berechenbaren analytischen Funktionen hat Müller [Mül95] untersucht. Er zeigt, daß die Summe einer TT-berechenbaren Funktion TT-berechenbar ist, und daß die Koeffizientenfolge der Potenzreihenentwicklung einer TT-berechenbaren analytischen Funktion TT-berechenbar ist, und daß dies im Sinne der Typ-2 Theorie nicht konstruktiv ist. Er untersucht auch die analytische Fortsetzung TT-berechenbarer Funktionen. Seine Methoden verwenden jedoch die durch die Maschinen gegebene Ausgabegenauigkeitsschranke und können daher nicht für analytische Maschinen verwendet werden.

3.3 Fazit

In diesem Kapitel wurden die Resultate aus Kapitel 2 verwendet, um die Berechenbarkeit analytischer Funktionen zu diskutieren. Wir haben argumentiert, daß Turing-Maschinen und auch endliche $\mathbb{C}$-Maschinen zur Charakterisierung berechenbarer analytischer Funktionen ungeeignet sind. Die in Kapitel 2 eingeführten analytischen Maschinen mit unendlichen Berechnungen eignen sich besser zur Definition berechenbarer analytischer Funktionen. Dabei wird unterschieden zwischen berechenbaren analytischen Funktionen und koeffizientenberechenbaren analytischen Funktionen, also Funktionen, bei denen die Koeffizientenfolgen der Potenzreihenentwicklung eine analytisch berechenbare Folge bilden.
Wir haben gezeigt, daß eine Funktion, die in einem Punkt koeffizientenberechenbar ist, bereits im gesamten Konvergenzbereich der Potenzreihe der Funktion in diesem Punkt analytisch berechenbar ist, und daß sogar alle Ableitungen der Funktion in diesem Bereich analytisch berechenbar sind. Schließlich konnten wir zeigen, daß eine Funktion, die in einem Punkt koeffizientenberechenbar ist, bereits auf dem gesamten Konvergenzkreis ihrer Potenzreihe um diesen Punkt koeffizientenberechenbar ist.
Die klassischen analytischen Funktionen haben die Eigenschaft, daß alle Informationen über sie in ihrer Potenzreihe in einem einzigen Punkt enthalten sind. Ausgehend von einem Funktionskeim kann eine analytische Funktion bis an die Grenzen ihres natürlichen Definitionsbereichs fortgesetzt werden. Im allgemeinen ist diese Fortsetzung nicht mehr eindeutig, sondern es wird notwendig, den Definitionsbereich der Funktion zu einer Riemannschen Fläche zu erweitern. Setzt man die Funktion jedoch entlang eines bestimmten Pfades in der Ebene fort, so folgt aus dem Identitätssatz für analytische Funktionen die Eindeutigkeit der Fortsetzung. Mit Hilfe unserer Resultate folgt, daß die Fortsetzbarkeit aus einem einzigen Punkt heraus nicht nur für die Funktion selbst, sondern auch für die Berechenbarkeit folgt: Betrachtet man stets nur einen konkreten Zweig der Funktion, so ist eine in einem Punkt koeffizientenberechenbare Funktion bereits auf ihrem gesamten Definitionsbereich koeffizientenberechenbar!
Weiter haben wir gezeigt, daß die koeffizientenberechenbaren Funktionen auch unter Komposition abgeschlossen sind. Angesichts der Tatsache, daß die allgemeinen analytischen Maschinen nicht unter Komposition abgeschlossen sind, ist dies ein wichtiges Resultat, da es zeigt, daß die Koeffizientenberechenbarkeit solche Abschlußeigenschaften erzwingt. Schließlich haben wir gezeigt, daß koeffizientenberechenbare Funktionen unter lokaler Umkehr, einer weiteren grundlegenden Eigenschaft analytischer Funktionen, abgeschlossen sind.
Das Resultat, daß jede koeffizientenberechenbare Funktion auch analytisch berechenbar ist, wirft die Frage auf, inwieweit analytisch berechenbare Funktionen auch koeffizientenberechenbar sind. Dabei konnten wir mit Techniken aus Kapitel 2 zeigen, daß analytische Funktionen, die mittels einer Maschine berechnet werden können, bei denen wenigstens ein Berechnungspfad einen offenen nichtleeren Einzugsbereich hat, schon koeffizientenberechenbar sind.
Für die allgemeinen analytisch berechenbaren Funktionen gilt der Hierarchiesatz: Die Klasse der δ-$\mathbb{Q}$-analytisch berechenbaren Funktionen ist echt in der Klasse der $\mathbb{R}$-analytisch berechenbaren Funktionen enthalten, und durch Hintereinanderausführung mit

weiteren analytischen Maschinen ergibt sich eine unendliche echt aufsteigende Hierarchie von Funktionsklassen. Unsere Resultate ergeben, daß diese Hierarchie unter der Bedingung der Koeffizientenberechenbarkeit zusammenbricht. Für Funktionen, die analytisch koeffizientenberechenbar sind, haben wir auch gezeigt, daß sie δ-$\mathbb{Q}$-approximierbar sind.

Kapitel 4
Rekursive Funktionen

Während in den voranstehenden Kapiteln die Berechenbarkeit reeller und komplexer Funktionen mittels Maschinen untersucht wurde, wird im vorliegenden Kapitel eine andere Herangehensweise an die Berechenbarkeit betrachtet. In der klassischen Theorie gibt es viele äquivalente Ansätze, berechenbare Funktionen zu charakterisieren. Maschinenmodelle wie Turing-Maschinen oder Registermaschinen führen zu einer operationalen Definition der Berechenbarkeit. Es gibt auch funktionale Ansätze, wie etwa den λ-Kalkül und die Klasse der μ-rekursiven Funktionen. Mit letzterer Klasse beziehungsweise deren Teilklasse der primitiv-rekursiven Funktionen werden wir uns in diesem Kapitel beschäftigen.
Dabei interessieren uns Fortsetzungen rekursiver Funktionen zu analytischen Funktionen. Mit Hilfe von Rekursionsgleichungen werden Funktionen auf den natürlichen Zahlen definiert, beispielsweise die Fakultät wird definiert durch $0! = 1$, $(n+1)! = (n+1) \cdot n!$. Wir stellen die Frage, ob es analytische Funktionen auf möglichst großen Teilbereichen von $\mathbb{C}$ gibt, die auf den natürlichen Zahlen mit den rekursiven Funktionen übereinstimmen, und die zudem die Rekursionsgleichung auf ihrer gesamten Definitionsmenge erfüllen. Für die Fakultät z.B. ist die Gammafunktion $\Gamma(z)$ eine analytische Fortsetzung auf $\mathbb{C} - \{-n \mid n \in \mathbb{N}\}$, und hierbei ergeben sich die Singularitäten an den negativen natürlichen Zahlen zwingend aus der Rekursionsgleichung.
Die Fortsetzung berechenbarer Funktionen zu analytischen Funktionen eröffnet die Möglichkeit, die Theorie der analytischen Funktionen für Aussagen über klassische berechenbare Funktionen zu nutzen. Ist neben der Fortsetzung auch ein Eindeutigkeitskriterium für die Fortsetzung gegeben, so können Invarianten der fortgesetzten analytischen Funktionen genutzt werden, um Rückschlüsse auf die ihnen zugrunde liegenden natürlichen rekursiven Funktionen zu ziehen.
In [Gä01] wurden Fortsetzungen für eine Klasse rekursiver Funktionen gegeben, den linear primitiv-rekursiven Funktionen in einer Veränderlichen. Hier untersuchen wir, inwieweit die Beschränkung auf lineare Rekursionen die Berechnungsmächtigkeit einschränkt. Dazu definieren wir zunächst die zugrunde liegende Klasse berechenbarer Funktionen über den natürlichen Zahlen und ordnen diese im Rahmen der primitiv-rekursiven Funktionen ein. Dann fassen wir kurz die Ergebnisse aus [Gä01] zusammen, um einen Überblick zu geben, für welche linear primitiv-rekursiven Funktionen eindeutige Fortsetzungen gegeben werden können.
Schließlich gehen wir noch kurz auf nichtlineare Rekursionen ein. Wir skizzieren eine Methode, die unter Zurückführung auf die Methoden in [Gä01] eine Klasse nichtlinear rekursiver Funktionen fortsetzt.

4.1 Linear primitiv-rekursive Funktionen

In diesem Abschnitt betrachten wir die Klasse der linear primitiv-rekursiven Funktionen. Zunächst skizzieren wir zur besseren Einordnung die Klasse der μ-rekursiven Funktionen. Für eine genauere Darstellung verweisen wir auf [Rog67]. Unsere Resultate dieses Abschnitts über die Einordnung der linear primitiv-rekursiven Funktionen sind losgelöst von der Berechenbarkeit über $\mathbb{R}$ oder $\mathbb{C}$. Sie passen thematisch in die klassische Rekursionstheorie.

4.1.1 μ-rekursive Funktionen

Auf Basis der Klasse der μ-rekursiven Funktionen kann man die klassische Berechenbarkeitstheorie über den natürlichen Zahlen aufbauen. Dabei werden zunächst die primitiv-rekursiven Funktionen definiert. Diese bilden die Klasse der Funktionen, die man erhält, wenn man von einer Menge von *Basisfunktionen* ausgeht und *Einsetzungen* und *primitive Rekursionen* als Operationen zuläßt.
Dabei bestehen die Basisfunktionen aus der k-stelligen Nullfunktion, den Projektionen $\mathbb{N}^k \to \mathbb{N}$ und der Nachfolgerfunktion. Sind $\gamma_1, \ldots, \gamma_n$ k-stellige Funktionen und ist ϕ eine n-stellige Funktion, so ist die Einsetzung der γ_j in ϕ definiert durch $\psi(x) = \phi(\gamma_1(x), \ldots, \gamma_n(x))$. Die primitive Rekursion ist eine Vorschrift der Form

$$\phi(0, x) = g(x),\ \phi(n+1, x) = h(n, x, \phi(n, x)),$$

wobei ϕ $(k+1)$-stellig, g k-stellig und h $(k+2)$-stellig ist und g und h primitiv-rekursiv sind.
Formal definiert man die Klasse der primitiv-rekursiven Funktionen als die kleinste Klasse von Funktionen, die die Basisfunktionen enthält und abgeschlossen ist unter Einsetzung und primitiver Rekursion.
Die Klasse der primitiv-rekursiven Funktionen ist eine sehr große Klasse von Funktionen, sie umfaßt aber nicht alle Turing-berechenbaren Funktionen. Die *Ackermannfunktion* ist ein Beispiel einer Turing-berechenbaren Funktion, die nicht primitiv-rekursiv ist. Dazu definiert man

$$\begin{aligned} a(0, k) &= k + 1 \\ a(n+1, 0) &= a(n, 1) \\ a(n+1, m+1) &= a(n, a(n+1, m)) \end{aligned}$$

Man kann zeigen, daß diese Funktion nicht primitiv-rekursiv ist. Sie ist jedoch berechenbar, wenn man die Klasse der primitiv-rekursiven Funktionen mit dem μ-Operator zur Klasse der μ-rekursiven Funktionen ergänzt:

$$(\mu_y : \{f(x, y) = 0\}) : x \mapsto \begin{cases} \min\{y \mid f(x, y) = 0\} & \text{ex. } y \text{ mit } f(x, y) = 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Die Klasse der μ-rekursiven Funktionen ist die kleinste Klasse, die die Basisfunktionen enthält und abgeschlossen ist unter Einsetzung, primitiver Rekursion und dem μ-Operator. Es ist ein klassisches Resultat aus der Berechenbarkeitstheorie, daß die Klasse der μ-rekursiven Funktionen gleich der Klasse der Turing-berechenbaren Funktionen ist.

4.1.2 Definition der linear primitiv-rekursiven Funktionen

Wir geben nun die Definition der linear primitiv-rekursiven Funktionen. Dabei wird die allgemeine Rekursionsvorschrift der primitiven Rekursion

$$F(0,x) = A(x),\ F(n+1,x) = B(n,x,F(n,x))$$

mit bereits als primitiv-rekursiv definierten Funktionen A und B ersetzt durch die *lineare Rekursion*

$$f(0,x) = a(x),\ f(n+1,x) = g(n,x)f(n,x) + h(n,x)$$

Hierbei und auch im folgenden schreiben wir abkürzend x für das Tupel $(x_1,\ldots,x_k)$, wobei die Stelligkeit unwesentlich ist oder aus dem Zusammenhang hervorgeht.
Der Begriff "linear" bezieht sich also auf die Rekursionsvorschrift. Wir bezeichnen eine Funktion als *linear rekursiv*, wenn sie in einer endlichen Anzahl von Kompositionen und linearen Rekursionen aus den Basisfunktionen der Polynome hervorgeht. Wir präzisieren die Definition der linear rekursiven Funktionen:

1. **Basisfunktionen**
   - Konstanten
   - Arithmetische Operationen: Addition, Subtraktion, Multiplikation
   - Projektionen $\pi_k : (x_1,\ldots,x_n) \mapsto x_k$
2. **Einsetzung**
   Seien die r-stellige Funktion f und die n-stelligen Funktionen $g_1,\ldots,g_r$ gegeben.
   $$\phi(x_1,\ldots,x_n) = f(g_1(x_1,\ldots,x_n),\ldots,g_r(x_1,\ldots,x_n))$$
   ist die *Einsetzung* der g_i in f.
3. **Lineare Rekursion**
   Seien die r-stellige Funktion a und die $r+1$-stelligen Funktionen g und h gegeben. Die $r+1$-stellige Funktion f entsteht aus ihnen durch *lineare Rekursion*, falls
   $$\begin{aligned} f(0,x) &= a(x) \\ f(n+1,x) &= g(n,x)f(n,x) + h(n,x) \end{aligned}$$

Definition 4.1.1. *Die Klasse der* linear primitiv-rekursiven Funktionen *ist die kleinste Klasse von Funktionen, die die Basisfunktionen enthält und abgeschlossen ist unter Einsetzung und linearer Rekursion.*

Anmerkung.

1. Aus der Abgeschlossenheit unter Komposition folgt mit den arithmetischen Operationen, Projektionen und Konstanten sofort, daß alle (auch multivariate) Polynome linear rekursiv sind. Man könnte ebenso die Polynome als Basisfunktionen wählen.
   Da wir bei den arithmetischen Operationen auch die Subtraktion zulassen, wir aber andererseits nur Funktionen mit Werten in $\mathbb{N}^k$, $k \in \mathbb{N}$ betrachten, sei angemerkt, daß das Minuszeichen stets als *nichtnegative Differenz* zu verstehen ist. Diese ist definiert durch $a \dot{-} b = a - b$ falls $a \geq b$ bzw. durch $a \dot{-} b = 0$ falls $a < b$.

2. Die Definition als kleinste Klasse von Funktionen mit den genannten Eigenschaften ist eine Standardvorgehensweise in der Mathematik, formal definiert man sie als Durchschnitt über alle Funktionsklassen mit diesen Eigenschaften.
   Man kann die Klasse auch durch Induktion definieren als Menge von Funktionen, die ausgehend von den Basisfunktionen durch endliche Hintereinanderausführung der Operationen Einsetzung und lineare Rekursion definierbar sind. Die Äquivalenz der beiden Definitionen ist leicht zu zeigen. Sie rechtfertigt die übliche Vorgehensweise, Eigenschaften der Funktionen mittels Induktion über den Aufbau der rekursiven Funktionen zu beweisen.
3. Der Kürze halber sprechen wir im folgenden auch von *linear rekursiven Funktionen*.

Die linear rekursiven Funktionen bilden eine Unterklasse der primitiv-rekursiven Funktionen. Im folgenden wird gezeigt, daß eine große Teilklasse der primitiv-rekursiven Funktionen bereits linear rekursiv ist. Insbesondere sind alle in Polynomialzeit berechenbaren Funktionen linear rekursiv. Der Beweis ähnelt dem klassischen Beweis, daß die Turing-berechenbaren Funktionen μ-rekursiv sind. In diesem wird gezeigt, daß die Berechnungsschritte einer Turing-Maschine primitiv-rekursiv sind und schließlich die Existenz einer haltenden Berechnung mit Hilfe des μ-Operators überprüft. Wir zeigen hier, daß die Berechnungsschritte einer Turing-Maschine, die gewissen Beschränkungen unterworfen ist, sogar linear rekursiv sind.
Wir geben zunächst einige Beispiele linear rekursiver Funktionen.

Lemma 4.1.2. *Folgende Funktionen sind linear primitiv-rekursiv.*

1. *Polynome* $p(x_1, \ldots, x_n)$
2. *Vorgängerfunktion* $V(0) = 0, V(x+1) = x$
3. *Signumfunktion* $\operatorname{sgn}(x) = \begin{cases} 1 & x \neq 0 \\ 0 & x = 0 \end{cases}$
4. *Konditional: Sind* f, g, h *linear primitiv-rekursiv, so auch*

$$\texttt{if} \quad f(x_1, \ldots, x_n) = 0 \quad \texttt{then} \quad g(x_1, \ldots, x_n) \quad \texttt{else} \quad h(x_1, \ldots, x_n)$$

Beweis.

1. Sukzessive Andwendung der arithmetischen Operationen.
2. Lineare Rekursion mit $V(0) = 0$; $V(n+1) = 1 \cdot n + 0$.
3. $\operatorname{sgn}(0) = 0$; $\operatorname{sgn}(n+1) = 1$ (Elementare lineare Rekursion).
4. Die Funktion $\texttt{if} \quad f(x) = 0 \quad \texttt{then} \quad g(x) \quad \texttt{else} \quad h(x)$ ist definiert durch

$$\operatorname{sgn}(f(x))g(x) + \big(1 - \operatorname{sgn}(f(x))\big)h(x).$$

□

Bekanntlich führt die Anwendung des μ-Operators aus der Klasse der primitiv-rekursiven Funktionen heraus. Ist aber eine Obergrenze der gesuchten Nullstellen bekannt, dann ist dies nicht der Fall, d.h. der *beschränkte μ-Operator* ist primitiv-rekursiv.

Tatsächlich ist die Anwendung des beschränkten μ-Operators auf eine linear rekursive Funktion wieder linear rekursiv. Wir betrachten dazu für linear rekursive Funktionen f

$$(\exists_{y\leq z} : \{f(x,y) = 0\}) : (x,z) \mapsto \begin{cases} 1 & \exists y \leq z : f(x,y) = 0 \\ 0 & \text{sonst} \end{cases}$$

und

$$(\mu_{y\leq z} : \{f(x,y) = 0\}) : (x,z) \mapsto \begin{cases} \min\{y \leq z : f(x,y) = 0\} & \exists y \leq z : f(x,y) = 0 \\ z & \text{sonst} \end{cases}$$

Wir schreiben kurz $\exists_f(x,z) = 0$ für $(\exists_{y\leq z} : f(x,y) = 0)(x,z)$ und $\mu_f(x,z)$ für $(\mu_{y\leq z} : f(x,y) = 0)(x,z)$.

Satz 4.1.3. *Für linear rekursive Funktionen f definieren der* beschränkte Existenzoperator $\exists_f(x,z)$ *und der* beschränkte μ-Operator $\mu_f(x,z)$ *wieder linear rekursive Funktionen.*

Beweis.

$$\exists_f(x,0) = \texttt{if} \quad f(x,0) = 0 \quad \texttt{then} \quad 1 \quad \texttt{else} \quad 0$$

$$\exists_f(x,z+1) = \operatorname{sgn}(f(x,z+1)) \cdot \exists_f(x,z) + \big(1 - \operatorname{sgn}(f(x,z+1))\big)$$

und

$$\mu_f(x,0) = 0$$

$$\mu_f(x,z+1) = \exists_f(x,z) \cdot \mu_f(x,z) + (1 - \exists_f(x,z)) \cdot (z+1)$$

□

Wir werden mit der Schreibweise bei diesen Operatoren im folgenden lax umgehen, insbesondere werden wir den μ-Operator auf Mengen anwenden, die durch eine Gleichung oder Ungleichung linear rekursiver Funktionen definiert sind. Es ist offensichtlich, wie solche Anwendungen des μ-Operators auf die klassische Formulierung hin umgeschrieben werden können. Als Beispiel steht $\mu_{y\leq x} : \{f(y) \geq b\}$ für $\mu_{y\leq x} : \{b - f(y) = 0\}$. Ebenso verwenden wir logische Verknüpfungen, $\mu_{y\leq x} : \{f(y) = 0 \wedge g(y) = 0\}$ steht etwa für $\mu_{y\leq x} : \{f(y) + g(y) = 0\}$ (dabei wird benutzt, daß keine negativen Zahlen vorkommen können) und $\mu_{y\leq x} : \{f(y) = 0 \vee g(y) = 0\}$ für $\mu_{y\leq x} : \{f(y) \cdot g(y) = 0\}$.
Die linear rekursive Definition des beschränkten μ-Operators stellt den wesentlichen Beweisschritt in Richtung linear rekursiver Berechenbarkeit der Turing-Berechnungsschritte dar, da hierdurch die allgemeine primitive Rekursion unter gewissen Voraussetzungen eliminiert werden kann. Wir geben ein paar Beispiele für die Anwendung der Operatoren. Einige der hier definierten Funktionen werden wir im folgenden noch benötigen.

Beispiel 4.1.4.

- Die ganzzahlige Division $\div : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $(x,y) \mapsto \left\lfloor \frac{x}{y} \right\rfloor$ ist linear rekursiv. Denn es ist

$$x \div y = \mu_{a\leq x} : \{a \cdot y > x\} - 1$$

- Der Rest der Division $x \bmod y$ ist damit auch linear rekursiv: $x \bmod y = x - y(x \dot{\div} y)$.
- Der abgerundete Logarithmus $\log : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $x \mapsto \lfloor \log_2(x) \rfloor$ wird definiert durch

$$\log x = \mu_{a \leq x} : \{2^a > x\} - 1$$

Mit Hilfe des beschränkten μ-Operators gelingt es, die *Paarungsfunktionen* linear rekursiv zu definieren.

Lemma 4.1.5. *Es gibt eine bijektive linear rekursive* Paarungsfunktion $P : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *mit linear rekursiven Umkehrfunktionen L und R, so daß $P(L(z), R(z)) = z$ und $L(P(x, y)) = x$, $R(P(x, y)) = y$.*

Beweis. Die Funktion $P(x, y) = \frac{1}{2}(x+y)(x+y+1)+x$ ist als Polynom linear primitiv-rekursiv und ist bekanntlich bijektiv. Sie stellt die diagonale Aufzählung von $\mathbb{N} \times \mathbb{N}$ dar. Für die Umkehrfunktionen L und R gilt wegen der Bijektivität $L(z) = \mu_{x \leq z} : \{\exists_{y \leq z} : \{P(x, y) = z\}\}$ bzw. $R(z) = \mu_{y \leq z} : \{\exists_{x \leq z} : \{P(x, y) = z\}\}$ □

Es ist klar, daß man nun induktiv linear rekursive Bijektionen $\mathbb{N} \to \mathbb{N}^k$ mit linear rekursiven Umkehrfunktionen erhält.
Die Klasse der allgemein μ-rekursiven Funktionen ist identisch mit der der Turing-berechenbaren Funktionen. Die μ-Rekursivität der Turing-Funktionen kann man beweisen, indem man die einzelnen Berechnungsschritte einer Turing-Maschine mit Hilfe primitiv-rekursiver Funktionen beschreibt und schließlich mit dem μ-Operator die Anzahl der Berechnungsschritte berechnet, die zu einem Endzustand führen. Der folgende Satz zeigt, daß unter gewissen Voraussetzungen sogar die linear rekursiven Funktionen ausreichen. Insbesondere folgt aus diesem Satz, daß die polynomialzeitberechenberen Funktionen auch linear rekursiv sind.

Satz 4.1.6. *Sei $F : \mathbb{N} \to \mathbb{N}$ in Zeit $t(n)$ berechenbar, wobei die Funktion t linear rekursiv sei. Dann ist F bereits linear rekursiv.*

Beweis. Der Beweis erfolgt in mehreren Schritten. Wir betrachten ein einfaches Turing-Maschinenmodell mit einem nach rechts hin unendlichen Band und dem Alphabet $A = \{0, 1\}$. Die Funktion F wird somit von einer Maschine $\mathcal{M}$ berechnet, die als Eingabe eine Binärdarstellung der Zahl n erhält und in maximal $t(\log n)$ (der Länge dieser Darstellung) Rechenschritten die Binärdarstellung der Zahl $F(n)$ auf dem Band erzeugt. Wir werden im folgenden zunächst zeigen, daß bei einer geeigneten Kodierung die Übergangsfunktion der Turing-Maschine linear rekursiv ist. Im Anschluß zeigen wir, daß dies auch für die Funktion gilt, die einer Schrittzahl k die k-te Konfiguration zuordnet. Da die Funktion F $t(n)$-zeitbeschränkt ist, kann mit dem beschränkten μ-Operator überprüft werden, ob die Maschine nach spätestens $t(n)$ Schritten in einen Endzustand gelangt.

- Wir kodieren die 0-1-Folge, die den Inhalt des Bandes der Turing-Maschine darstellt, stets als die der Folge zugeordnete natürliche Zahl, wobei wir, um führende

Nullen zu berücksichtigen, stets eine 1 als erstes Zeichen jeder Folge hinzufügen. Wir betrachten dazu die Funktion

$$\langle . \rangle : \{0,1\}^* \to \mathbb{N},\ a_1 \dots a_n \mapsto \sum_{k=1}^{n} a_k 2^{n-k} + 2^n$$

Auf diese Weise erhalten wir eine injektive Kodierung aller Bandinhalte auf die natürlichen Zahlen. Wir betrachten nun die Funktion, die einer natürlichen Zahl (die Position des Kopfes der Turing-Maschine) und einem kodierten Bandinhalt das entsprechende Element aus {0,1} zuordnet:
$\texttt{pos} : \mathbb{N} \times \mathbb{N} \to \{0,1\},\ (i, \langle a_1 \dots a_n \rangle) \mapsto a_i$. Diese Funktion ist linear rekursiv, denn es ist

$$\texttt{pos}(i, \langle a_1 \dots a_n \rangle) = ((\langle a_1 \dots a_n \rangle) \div (\log n - i)) \bmod 2.$$

Mit Hilfe von Rest-Divisionen können wir also auf beliebige Zeichen eines Blocks linear rekursiv zugreifen.

- Nun betrachten wir die Übergangsfunktion δ der Turing-Maschine $\mathcal{M}$, die auf Konfigurationen angewendet wird. Es sei $\delta(z, i, \langle B \rangle) = (z', i', \langle B' \rangle)$ die Funktion, die einer Konfiguration der Maschine bestehend aus Zustandsnummer z, Kopfposition i und kodiertem Bandinhalt $\langle B \rangle$ die nächste Konfiguration zuordnet. Da mit Hilfe der Funktion `pos` auf den Inhalt der i-ten Zelle des Bandes zugegriffen werden kann, sieht man nun, daß die Funktion δ linear rekursiv ist. Sie wird durch zwei geschachtelte Fallunterscheidungen definiert, die mit Hilfe der linear rekursiven `if then else`-Konstruktion realisiert werden.

- Um nun die k-Schritt-Funktion der Turing-Maschine zu erhalten, müssen wir die Kodierung dahingehend erweitern, daß nicht nur Bandinhalte, sondern Konfigurationen bestehend aus Zustandsnummer, Kopfposition und Bandinhalt kodiert werden. Die Kodierung erfolgt analog der Kodierung der Bandinhalte, d.h. die Zustandsnummer und die Kopfposition werden anhand ihrer Binärdarstellung kodiert. Nun soll die Kodierung einer zusätzlichen Bedingung genügen: Sie soll auf eine vorgegebene Länge normiert werden. Genauer betrachten wir die Kodierungsfunktion $c : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, (l, n) \mapsto \langle \text{bin}(n) 10^{l - |\text{bin}(n)|} \rangle$. Dabei ist $\text{bin} : \mathbb{N} \to \{0,1\}^*$ die Binärdarstellung und $|.|$ die Längenfunktion. c berechnet also zunächst die Binärdarstellung der Eingabe n, fügt dieser Binärdarstellung eine 1 an und hängt so viele Nullen an, daß die Länge der erhaltenen Zeichenkette gerade l ist. Von dieser Zeichenkette wird anschließend die vermöge der injektiven Abbildung $\langle . \rangle$ zugehörige natürliche Zahl bestimmt. Die Abbildung c ist linear rekursiv, denn es gilt $c(l, n) = \langle \text{bin}(n) \rangle \cdot 2^{l - (\log(n)+1)} + 2^{l - \log(n)}$. Mit Hilfe der Division $\div$ und des Restoperators mod kann auch die Dekodierungsfunktion $d : \mathbb{N} \times \mathbb{N} \to \mathbb{N},\ (l, C) \mapsto n$ mit $d(l, c(l, n)) = n$ linear rekursiv berechnet werden.
  Sei $*$ der Konkatenationsoperator. Die Kodierung einer Konfiguration $(z, i, \langle B \rangle)$ zu vorgegebener Länge l besteht nun in der Konkatenation der Kodierungen $c(l, z) * c(l, i) * c(l, \langle B \rangle)$. Hierbei wird vorausgesetzt, daß l hinreichend groß ist, so daß die Kodierungsabbildung wohldefiniert ist. Wir werden sehen, daß dies bei geeigneter Wahl von l stets möglich ist.

- Wir betrachten nun zu vorgegebenem k eine Folge von Konfigurationen

  $$(z_0, i_0, \langle B_0 \rangle), \ldots, (z_k, i_k, \langle B_k \rangle).$$

  Zu gegebener Länge $l(k)$ sei

  $$C(k) := c(l(k), z_0) * c(l(k), i_0) * c(l(k), \langle B_0 \rangle) * \ldots * c(l(k), z_k) * c(l(k), i_k) * c(l(k), \langle B_k \rangle)$$

  die Konkatenation der Kodierungen der Konfigurationen.
  Die Länge $l(k)$ wählen wir als das Maximum der Größen $\log |Z| + 1$ und $t(\log n) + 1$, wobei $|Z|$ die Anzahl der Zustände der Turing-Maschine $\mathcal{M}$ und $t(n)$ die oben vorausgesetzte zeitbeschränkende Funktion seien.
  Da in dieser Kodierung alle Zahlen mit gleicher Länge kodiert werden, läßt sich linear rekursiv mit Hilfe der Funktionen $\div$ und mod aus der Kodierung $C(k)$ bei vorgegebenem j die Konfiguration $(z_j, i_j, \langle B_j \rangle)$ bestimmen. Es sei $\texttt{conf}(l, C, i)$ die Funktion, die ein Wort C als Kodierung der Form $C(k)$ interpretiert und die i-te Konfiguration auswählt. Die Funktion `conf` ist, wie eben besprochen, linear rekursiv.

- Im folgenden speichern wir die Information über den Index k und die zugehörige Kodierung $C(k)$ mit Hilfe der Paarungsfunktion P in einer einzigen natürlichen Zahl $w = P(k, C(k))$. Die Funktion $l(k)$ sei die zur Maschine $\mathcal{M}$ gehörende Längenfunktion. Wir definieren nun die Funktion $\texttt{start} : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$. Dazu betrachten wir eine Kodierung einer Konfigurationsfolge C, die zusammen mit der Länge k in einer Zahl w wie im vorigen Absatz beschrieben kodiert wird. Es sei $\texttt{start}(n, w) = 1$ genau dann, wenn die unter C kodierte Folge von Konfigurationen als erste Konfiguration einem Startzustand mit Eingabe n entspricht. Die Funktion $\texttt{end} : \mathbb{N} \to \{0, 1\}$ erfülle $\texttt{end}(w) = 1$ genau dann, wenn die letzte kodierte Konfiguration einem Endzustand entspricht. `start` und `end` sind offenbar wieder linear rekursiv.

- Mit Hilfe der bisher definierten linear rekursiven Funktionen läßt sich nun die ursprüngliche Funktion F linear rekursiv definieren. Wenn eine Zahl w existiert, so daß $w = P(k, C)$ und C eine Kodierung einer Folge von Konfigurationen mit Kodierungslänge k darstellt, weiter das erste Glied der Folge eine Startkonfiguration mit Eingabe n darstellt, das letzte Glied der Folge einem Endzustand entspricht und schließlich alle Konfigurationsübergänge korrekt sind, dann entspricht die Folge der Konfigurationen C gerade einer haltenden Berechnung der Turing-Maschine $\mathcal{M}$. Da die Maschine Zeit t-beschränkt ist, können wir mit Hilfe des beschränkten μ-Operators das kleinste solche w bestimmen. Es gilt also:

  $$F(n) = \texttt{val}\Big(\mu_{w \le s(n)} : \{ \quad k = L(w) \wedge C = R(w) \wedge \texttt{start}(n, w) = 1 \wedge \texttt{end}(w) = 1 \\ \bigwedge_{i=1}^{k} : \{\delta(\texttt{conf}(l(k), C, i-1)) = \texttt{conf}(l(k), C, i)\}\}\Big)$$

  Hierbei bestimmt die linear rekursive Funktion `val` noch aus einer Kodierung $w = P(k, c)$ den Bandinhalt der letzten Konfiguration und interpretiert diesen als natürliche Zahl.

Die obere Schranke $s(n)$ des beschränkten μ-Operators ist ebenfalls eine linear rekursive Funktion, denn mit

$$s(n) := P(n, t(\log n) \cdot 3 \cdot l(n))$$

läßt sich die Größe der Zahl w nach oben abschätzen. Die Schranke ergibt sich daraus, daß die Maschine $\mathcal{M}$ als $t(n)$-zeitbeschränkte Maschine höchstens $t(n)$ Schritte benötigt, um in einen Endzustand zu gelangen. Die Kodierungslänge jeder Konfiguration beträgt $3 \cdot l(n)$.
Bei der oben angegebenen linear rekursiven Definition der Funktion $F(n)$ ist noch die lineare Rekursivität der Konjunktion $\bigwedge_{i=1}^{k} : \{\delta(\mathtt{conf}(l(k), C, i-1)) = \mathtt{conf}(l(k), C, i)\}$ nachzuweisen. Dies läßt sich mit Hilfe des beschränkten μ-Operators leicht wie folgt realisieren:

$$\bigwedge_{i=1}^{k} : \big\{\delta(\mathtt{conf}(l(k), C, i-1)) = \mathtt{conf}(l(k), C, i)\big\}$$
$$\Leftrightarrow \big\{\mu_{i \le k} : \{\delta(\mathtt{conf}(l(k), C, i-1)) \neq \mathtt{conf}(l(k), C, i)\} = k\big\}$$

Schließlich die vollständige linear rekursive Beschreibung der Funktion F:

$$F(n) = \mathtt{val}\Big(\mu_{w \le s(n)} : \big\{k = L(w) \wedge C = R(w) \wedge \mathtt{start}(n, w) = 1 \wedge \mathtt{end}(w) = 1 \wedge (\mu_{i \le k} : \{\delta(\mathtt{conf}(l(k), C, i-1)) \neq \mathtt{conf}(l(k), C, i)\} = k)\big\}\Big)$$

□

Satz 4.1.6 besagt also, daß jede Funktion, die einer linear rekursiven Laufzeitschranke unterworfen ist, wieder linear rekursiv ist. Daraus folgt insbesondere, daß alle *in Polynomialzeit und in Exponentialzeit berechenbaren Funktionen linear rekursiv* sind.

4.1.3 Grenzen der linearen Rekursion

Wir zeigen nun, daß nicht alle primitiv-rekursiven Funktionen auch linear rekursiv sind. Dazu betrachten wir die *Turmfunktion*, die durch folgende primitive Rekursion definiert ist:

$$T(0, x) = x,\ T(n+1, x) = 2^{T(n,x)}$$

Es ist also

$$T(n, x) = \overbrace{2^{2^{\cdot^{\cdot^{x}}}}}^{n}$$

ein exponentieller Turm der Höhe n. Das extrem starke Wachstum der Turmfunktion werden wir ausnutzen, um zu zeigen, daß sie nicht linear rekursiv ist. Dazu zeigen wir, daß linear rekursive Funktionen gewissen Wachstumsbeschränkungen unterliegen.

Satz 4.1.7. *Sei $f(x_1, \ldots, x_k)$ eine linear rekursive Funktion. Dann gibt es eine Konstante c, so daß gilt:*

$$f(x_1, \ldots, x_k) \leq T(c, x_1 + \ldots + x_k)$$

Beweis. Wir zeigen die Aussage durch Induktion über den Aufbau der linear rekursiven Funktionen.

- *Basisfunktionen*: Daß die Aussage für die Basisfunktionen wie Konstanten, Projektionen und Polynome gilt, ist klar.

- *Einsetzung*: Sei $\phi(x) = \phi(x_1, \ldots, x_k) = f(g_1(x), \ldots, g_k(x))$ mit linear rekursiven ϕ und g_i, und nach Induktionsvoraussetzung gelte für alle $y_1, \ldots, y_k$ und für alle x:

  $$f(y_1, \ldots, y_k) \leq T(c_f, y_1 + \ldots + y_k),\ g_i(x) \leq T(c_{g_i}, x_1 + \ldots + x_k)\, \forall\, 1 \leq i \leq k$$

  Dann gilt mit $c_g := \max(c_{g_1}, \ldots, c_{g_k})$

  $$\begin{aligned}
  \phi(x_1, \ldots, x_k) &= f(g_1(x), \ldots, g_n(x)) \leq T(c_f, g_1(x) + \ldots + g_n(x)) \\
  &\leq T(c_f, T(c_{g_1}, x_1 + \ldots + x_k) + \ldots + T(c_{g_k}, x_1 + \ldots + x_k)) \\
  &\leq T(c_f, T(c_g, x_1 + \ldots + x_k) + \ldots + T(c_g, x_1 + \ldots + x_k)) \\
  &\leq T(c_f, kT(c_g, x_1 + \ldots + x_k)) \\
  &\leq T(c_f, T(\tilde{c}_g + 1, x_1 + \ldots + x_k)) \quad \text{sei } \tilde{c}_g \text{ hinr. groß} \\
  &= T(c_f + \tilde{c}_g + 1, x_1 + \ldots + x_k)
  \end{aligned}$$

  Dabei wurde die Monotonie der Turmfunktion in beiden Argumenten ausgenutzt und die Rechenregel $T(a, T(b, n)) = T(a + b, n)$ verwendet.

- *Lineare Rekursion*: Seien $a(x_1, \ldots, x_k), g(x_0, \ldots, x_k), h(x_0, \ldots, x_k)$ linear rekursiv und entstehe $f(x_0, \ldots, x_k)$ aus der linearen Rekursion

  $$\begin{aligned}
  f(0, x_1, \ldots, x_k) &= a(x_1, \ldots, x_k) \\
  f(n+1, x_1, \ldots, x_k) &= g(n, x_1, \ldots, x_k) f(n, x_1, \ldots, x_k) + h(n, x_1, \ldots, x_k)
  \end{aligned}$$

  und es sei nach Induktionsvoraussetzung

  $$\begin{aligned}
  a(x_1, \ldots, x_k) &\leq T(c_a, x_1 + \ldots + x_k) \\
  g(x_0, x_1, \ldots, x_k) &\leq T(c_g, x_0 + \ldots + x_k) \\
  h(x_0, x_1, \ldots, x_k) &\leq T(c_h, x_0 + \ldots + x_k)
  \end{aligned}$$

Dann gilt mit $c := \max(c_g, c_h, c_a)$

$$\begin{aligned}
f(n+1, x_1, \ldots, x_k) &= g(n, x_1, \ldots, x_k) f(n, x_1, \ldots, x_k) + h(n, x_1, \ldots, x_k) \\
&\leq T(c_g, n + x_1 + \ldots + x_k) f(n, x_1, \ldots, x_k) + T(c_h, n + x_1 + \ldots + x_k) \\
&\leq T(c, n + x_1 + \ldots + x_k)(f(n, x_1, \ldots, x_k) + 1) \\
&\leq T(c, n + x_1 + \ldots + x_k) 2 f(n, x_1, \ldots, x_k) \qquad (\text{oBdA } f(n, x_1, \ldots, x_k) \geq 1) \\
&\leq 2^2 T(c, n + x_1 + \ldots + x_k) T(c, n - 1 + x_1 + \ldots + x_k) f(n-1, x_1, \ldots, x_k) \\
&\leq \cdots \quad (\text{Aufrollen der Rekursion}) \\
&\leq 2^n \prod_{i=0}^{n} \left(T(c, n - i + x_1 + \ldots + x_k)\right) a(x_1, \ldots, c_k) \\
&\leq 2^n \left(T(c, n + x_1 + \ldots + x_k)\right)^n T(c_a, x_1 + \ldots + x_n) \\
&\leq 2^n T(c, n + x_1 + \ldots + x_k)^{n+1} \\
&= 2^{n + (n+1) T(c-1, n + x_1 + \ldots + x_k)} \\
&\leq T(\tilde{c}, n + 1 + x_1 + \ldots + x_k) \quad \text{für hinreichend großes } \tilde{c}
\end{aligned}$$

Dabei wurde erneut die Monotonie der Turmfunktion in beiden Argumenten, die Rekursionsformel der Turmfunktion und die Rechenregel $T(a, T(b, n)) = T(a + b, n)$ ausgenutzt.

□

Korollar 4.1.8. *Die Turmfunktion ist nicht linear rekursiv.*

Beweis. Wäre T linear rekursiv, so gäbe es ein c_T mit $T(x, y) \leq T(c_T, x + y)$, also insbesondere für alle $n \in \mathbb{N}$

$$T(n, 0) \leq T(c_T, n)$$

Dies kann aber nicht sein, da die Turmhöhe links mit n wächst, rechts aber fest ist und dort n nur in der höchsten Turmpotenz steht. □

4.1.4 Analytische linear rekursive Funktionen in einer Variablen

Im letzten Abschnitt haben wir die Klasse der linear rekursiven Funktionen definiert und im Rahmen der primitiv-rekursiven Funktionen eingeordnet. Wir sind an Fortsetzungen dieser Funktionen zu analytischen Funktionen interessiert. Dabei interessieren nicht einfach Interpolationen, die an Stellen mit ganzzahligen Koordinaten mit den natürlichen Funktionen übereinstimmen, sondern die auch der Rekursionsgleichung auf den komplexen Zahlen genügen. Der Übersicht halber fassen wir einige ältere Resultate zusammen, die die Fortsetzung der linear rekursiven Funktionen in einer Variablen betreffen.
Eine einfache linear rekursive Funktion ist die Fakultät $n! = 1 \cdot 2 \cdot \cdots \cdot n$. Diese Funktion hat eine "natürliche" Fortsetzung hin zu einer analytischen Funktion: Die Gammafunktion $\Gamma(z)$. Diese Funktion erfüllt auf ihrem gesamten Definitionsbereich die gleiche Rekursionsgleichung, die die Fakultät über den natürlichen Zahlen erfüllt. Sie ist eindeutig

unter der Voraussetzung der logarithmischen Konvexität. In [Gä01] wurden Verallgemeinerungen der Gammafunktion betrachtet, und es wurden allgemeine Lösungen zur Fortsetzung linear primitiv-rekursiver Funktionen in einer Variablen gegeben. Wir fassen kurz einige Ergebnisse aus [Gä01] zusammen.
Eine Rekursionsvorschrift der Form

$$\begin{aligned} f(0) &= a_0 \\ f(n+1) &= g(n)f(n) + h(n) \end{aligned}$$

bezeichnen wir als *lineare Rekursion in einer Variablen*. In [Gä01] wurden analytische Fortsetzungen für die linear rekursiven Funktionen gesucht, also analytische Funktionen, die der Gleichung

$$f(z+1) = g(z)f(z) + h(z)$$

genügen. Darüber hinaus wurden Kriterien für die Eindeutigkeit der Lösungen angegeben.
Die allgemeine lineare Rekursion läßt sich auf die Spezialfälle

$$f(z+1) = f(z)g(z) \tag{4.1}$$

und

$$f(z+1) = f(z) + h(z) \tag{4.2}$$

zurückführen. Bereits Hurwitz [Hur97] hat Lösungen der Gleichung für auf $\mathbb{C}$ meromorphe Funktionen angegeben. In [GH01] wurden diese Lösungen unabhängig gefunden, und zusätzlich wurden dort Eindeutigkeitskriterien für die Lösungen betrachtet, also Bedingungen, unter denen die Lösung der Rekursionsgleichung durch eine analytische Funktion bei gegebenem Startwert eindeutig bestimmt ist. Die Lösungen erhält man wie folgt:
Seien die Bernoulli-Polynome[1] $\phi_k(w)$ für $k \geq 1$ definiert durch die Potenzreihenentwicklung der Funktion

$$\frac{e^{wz}-1}{e^z-1} = 1 + \sum_{k=0}^{\infty} \frac{\phi_k(w)}{k!} z^k.$$

und sei $\phi_0(w) = w - 1$. Dann erfüllen die Bernoulli-Polynome ϕ_k die Gleichung

$$\phi_k(z+1) = \phi_k(z) + z^k.$$

Ist nun $h(z) = \sum_{k=0}^{\infty} a_k z^k$ eine analytische Funktion, so definiert man formal die *Bernoulli-Transformation*

$$(Bh)(z) := \sum_{k=0}^{\infty} a_k \phi_k(z).$$

[1]Die Bezeichnung in der Literatur ist uneinheitlich; gelegentlich werden die hier so definierten Funktionen als Bernoulli-Polynome bezeichnet, gelegentlich auch die durch die Entwicklung $\frac{ze^{wz}}{e^z-1} = \sum_{k=0}^{\infty} \frac{B_k(w)}{k!} z^k$ definierten Funktionen.

Unter der Voraussetzung der Konvergenz löst die Bernoulli-Transformation $f(z) = (Bh)(z)$ die Gleichung (4.2).
In [GH01] wird gezeigt, daß die Bernoulli-Transformation für ganze holomorphe Funktionen h konvergiert, deren Koeffizientenfolge die Bedingung $\sum_{k=0}^{\infty} k! \frac{|a_k|}{(2\pi)^k} < \infty$ erfüllt. Die Menge dieser Funktionen wird als *Bernoulli-Raum* $\mathcal{B}$ bezeichnet. Versehen mit einer geeigneten Norm $||\cdot||_{\mathcal{B}}$ wird dieser Raum ein Banach-Raum, und die Bernoulli-Transformation eine stetige lineare Abbildung von $\mathcal{B}$ in den Raum der ganzen Funktionen.
Diese Vorgehensweise ermöglicht ein einfaches Kriterium für die Eindeutigkeit der Fortsetzung dieser Klasse rekursiver Funktionen: Die Fortsetzung einer ganzzahligen Funktion, die die Gleichung (4.2) mit einem Startwert für $f(0)$ erfüllt, ist eindeutig unter der Voraussetzung, daß die Fortsetzung eines Polynoms wieder ein Polynom ist, und daß die Rekursion ein *stetiger* Operator bezüglich der angegebenen Norm ist.
Die allgemeine Lösung von Hurwitz [Hur97] ergibt sich, indem die Konvergenz der Bernoulli-Transformation durch konvergenzerzeugende Summanden erzwungen wird. Dazu werden periodische Funktionen $\pi_n(z)$ (dies sind im Effekt Approximationen der Fourier-Entwicklung der Bernoulli-Polynome) definiert, so daß

$$(\tilde{B}h)(z) := \sum_{k=0}^{\infty} a_k(\phi_k(z) - \pi_k(z))$$

konvergiert. Die Eindeutigkeit der Lösung, wie sie bei der Bernoulli-Transformation vorlag, ist auf diese Weise aber nicht mehr gegeben.

4.2 Ausblick: Fortsetzung iterativer Rekursionen

Während in den vergangenen Abschnitten *lineare* Rekursionen betrachtet wurden, gehen wir nun kurz auf nichtlineare Rekursionen ein. Wir betrachten Rekursionen der Form

$$f(z+1) = \phi(f(z)) \tag{4.3}$$

ϕ sei in diesem Fall eine bereits bekannte Funktion. Ein Beispiel für solche Funktionen ist die im letzten Abschnitt definierte *Turmfunktion*, die eine Fortsetzung der Prinzipien von Addition, Multiplikation und Potenzierung darstellt. Wir betrachten den Spezialfall $t(n) = T(n, 2)$, für den die rekursive Definition wie folgt aussieht:

$$t(0) = 1,\ t(n+1) = 2^{t(n)}$$

Also ist $t(n) = 2^{2^{\cdot^{\cdot^{2}}}}$ ein Turm der Höhe n.
In der mathematischen Literatur spricht man bei der Fortsetzung der Iteration auf die reellen und komplexen Zahlen von *fraktionaler Iteration*. Es gibt aber keine einheitliche Theorie, die diese Fortsetzungen abdeckt, sondern nur verschiedene Ansätze zur Fortsetzung einzelner Klassen von Funktionen.
Wir skizzieren im folgenden einen Ansatz, der für eine Klasse von Funktionen die Lösung der Gleichung (4.3) auf die Fortsetzung der linear rekursiven Funktionen, die in Abschnitt 4.1.4 besprochen wurden, zurückführt. Die Darstellung hat den Charakter eines Ausblicks, ob die dargestellte Vorgehensweise tatsächlich zu analytischen Lösungen der Rekursionsgleichungen führt, ist offen.
Wir betrachten solche Funktionen ϕ, die an der Stelle 0 einen Fixpunkt haben und deren erste Ableitung an der Stelle 0 gleich 1 ist, d.h. für die gilt $\phi(0) = 0$ und $\phi'(0) = 1$. Sei also

$$\phi(z) = z + a_2 z^2 + a_3 z^3 + \cdots$$

Wir untersuchen nun die Iteration von ϕ. Dazu machen wir den Ansatz

$$\phi^n(z) = z + A_2(n) z^2 + A_3(n) z^3 + \cdots$$

wobei $\phi^n(z)$ die n-fache Iteration von ϕ und $A_k(n)$ den k-ten Koeffizienten der Potenzreihendarstellung von ϕ^n bezeichne. Ausgehend von dieser Darstellung berechnen wir nun die $(n+1)$-te Iteration von ϕ. Für z aus dem Konvergenzbereich der Reihe können wir die Reihen entsprechend äquivalent umordnen:

$$\begin{aligned}
\phi^{n+1}(z) &= \phi^n(\phi(z)) \\
&= \phi(z) + A_2(n)(\phi(z))^2 + \cdots \\
&= z + (A_2(n) + a_2) z^2 + (A_3(n) + 2a_2 A_2(n) + a_3) z^3 + \cdots
\end{aligned}$$

Es gilt also

$$\begin{aligned}
A_2(n+1) &= A_2(n) + a_2 \\
A_3(n+1) &= A_3(n) + 2a_2 A_2(n) + a_3 \\
&\vdots
\end{aligned}$$

Man sieht, daß die $A_i(n)$ linear rekursiv vom Typ $f(z+1) = f(z) + h(z)$ sind. Induktiv sieht man weiter, daß die additive Funktion immer durch ein Polynom definiert werden kann, da die Lösung f einer Gleichung der Form $f(z+1) = f(z)+h(z)$ stets als Polynom gewählt werden kann, wenn h bereits ein Polynom ist. Wir können also mit Hilfe der Bernoulli-Polynome alle $A_i(n)$ fortsetzen. Wir präzisieren unsere Beobachtung, indem wir die Gleichungen für die $A_i(n)$ explizit bestimmen.
Ordnen wir nun zunächst $\phi(z)^k$, $k \in \mathbb{N}$ nach Potenzen von z und setzen $P(k,n) := \{(i_1,\ldots,i_n) \,|\, i_1,\ldots,i_n \geq 1,\ i_1+\cdots+i_k = l\}$, so erhalten wir

$$\phi(z)^k \;=\; \sum_{l=k}^{\infty} \Big(\sum_{(i_i,\ldots,i_k)\in P(k,n)} a_{i_1}\cdots a_{i_k} \Big) z^l$$

Nun ordnen wir die Reihe

$$\phi^{n+1}(z) = \phi(z) + \sum_{k=2}^{\infty} A_k(n)(\phi(z))^k$$

nach Potenzen von z und erhalten durch Koeffizientenvergleich für $l \in \mathbb{N}$

$$A_l(n+1) \;=\; A_l(n) + \sum_{k=2}^{l} A_k(n) \sum_{(i_i,\ldots,i_k)\in P(k,n)} a_{i_1}\cdots a_{i_k}$$

Induktiv sieht man, daß die $A_l(n)$ linear-rekursive Funktionen sind, die wir mit den bekannten Methoden fortsetzen können.

Lemma 4.2.1. *Sei $\phi(z) = z + \sum_{k=2}^{\infty} a_k z^k$, sei*

$$h_{k,l} = \sum_{(i_i,\ldots,i_k)\in P(k,n)} a_{i_1}\cdots a_{i_k},$$

und seien die linear primitiv-rekursiven Funktionen $A_l(n)$ induktiv definiert durch

$$A_1(n) = 1 \qquad n \in \mathbb{N}$$
$$A_l(0) = 0,\ A_l(n+1) = A_l(n) + \sum_{k=1}^{l-1} A_k(n) h_{k,l} \quad (l > 0) \tag{4.4}$$

Dann existieren Polynome $F_l(z)$, die die Funktionen $A_l(n)$ auf $\mathbb{C}$ fortsetzen und auf $\mathbb{C}$ der Rekursion

$$F_l(z+1) = F_l(z) + \sum_{k=1}^{l-1} F_k(z) h_{k,l}$$

genügen.

Beweis. Zunächst sei bemerkt, daß die Funktionen in Gleichung (4.4) nicht verschränkt rekursiv definiert sind. Sind die primitiv-rekursiven Funktionen $A_r(n)$, $r < l$ bereits definiert, so ist damit die primitiv-rekursive Funktion $A_l(n)$ definiert. Wir schreiben

$$h_l(n) := \sum_{k=1}^{l-1} A_k(n) h_{l,k}.$$

Damit nehmen die Rekursionen aus Gleichung (4.4) die Form

$$A_l(n+1) = A_l(n) + h_l(n) \tag{4.5}$$

an, wobei in $h_l(n)$ nur bereits definierte Funktionen vorkommen. Die Rekursionen in Gleichung (4.5) sind aber linear und mit den Methoden aus Abschnitt 4.1.4 lösbar. Induktiv können wir Funktionen $F_l(z), h_l(z)$ definieren mit

$$F_l(z+1) = F_l(z) + h_l(z), \tag{4.6}$$

indem wir einfach $F_l(z) := \mathcal{B}[h_l](z)$ setzen, wobei $\mathcal{B}$ die Bernoulli-Transformation bezeichne. Da die $h_{k,l}$ Konstanten sind, folgt induktiv sofort auch, daß die $F_l(z)$ Polynome sind, denn die Bernoulli-Transformation führt nicht aus der Menge der Polynome heraus. □

Die Idee, wie man für komplexe w die Iteration ϕ^w erklären könnte, liegt nun auf der Hand. Wir definieren zunächst formal

$$F(w,z) = \sum_{k=0}^{\infty} F_k(w) z^k$$

Unter der Voraussetzung der Konvergenz gilt nun

$$\begin{aligned} \phi(F(w,z)) &= F(w,z) + a_2 \left(F(w,z)\right)^2 + a_3 (F(w,z))^3 + \cdots \\ &= F(w,z) + a_2 \left(\sum_{k=0}^{\infty} F_k(w) z^k \right)^2 + \cdots \end{aligned}$$

Umordnung nach Potenzen von z ergibt

$$\begin{aligned} \phi(F(w,z)) &= z + (F_2(w) + a_2) z^2 + \cdots \\ &= z + \sum_{k=2}^{\infty} (F_k(w) + h_k(w)) z^k \\ &= z + \sum_{k=2}^{\infty} F_k(w+1) z^k \\ &= F(w+1, z) \end{aligned}$$

Wir haben also

Satz 4.2.2. *Sei $\phi(z) = z + a_2 z^2 + a_3 z^3 + \cdots$ analytisch und seien die Funktionen $A_l(n), h_l(n)$ rekursiv wie oben definiert. Weiter seien $F_l(z)$ die durch die Bernoulli-Polynome gegebenen Lösungen der Gleichungen $F_l(z+1) = F_l(z) + h_l(z)$. Sei nun z_0 vorgegeben, so daß die Reihe*

$$IT_\phi(w, z_0) := \sum_{k=0}^{\infty} F_k(w) z_0^k$$

in w und z in einer Umgebung von z_0 gleichmäßig konvergiert. Dann gilt

$$IT_\phi(w+1, z_0) = \phi(IT_\phi(w, z_0)).$$

Weiter sei z_0 so gewählt, daß $\phi(z_0) = a$. Dann ist mit

$$F(w) := IT_\phi(w, z_0)$$

eine Lösung der Gleichung

$$F(w+1) = \phi(F(w)),\ F(0) = a$$

gegeben.

Es ist offen, für welche Funktionen ϕ die in Satz 4.2.2 definierte Reihe $IT_\phi(w, z_0)$ in einer Umgebung von z_0 tatsächlich gleichmäßig konvergiert, und ob überhaupt Funktionen ϕ existieren, für die dies der Fall ist. Möglicherweise ist es notwendig, die Konvergenz mit *konvergenzerzeugenden Summanden* zu erzwingen, die die Rekursionsgleichungen der Funktionen nicht verändern. Auch hier ist offen, ob dies möglich ist.

4.3 Fazit

In diesem Kapitel wurde die Betrachtung der Berechenbarkeit von der Maschinendefinition gelöst, und ein Überblick über die Berechenbarkeit durch Rekursionsgleichungen gegeben. Es gibt keinen einheitlichen Begriff der Berechenbarkeit durch rekursive Funktionen über den rellen oder komplexen Zahlen. Eine Möglichkeit zur Definition dieser Berechenbarkeit besteht darin, die über den natürlichen Zahlen rekursiven Funktionen zu analytischen Funktionen fortzusetzen. Dabei ist mit Fortsetzung nicht lediglich die Interpolation der Funktion durch eine analytische Funktion an den ganzzahligen Stellen, sondern eine Fortsetzung auch der Rekursionsgleichung gemeint. Die fortgesetzte Funktion soll die Rekursionsgleichung als Funktionalgleichung auf ihrem Definitionsbereich erfüllen.
Die Fortsetzung rekursiver Funktionen zu analytischen Funktionen ermöglicht es, die Theorie der analytischen Funktionen auch für die Berechenbarkeitstheorie zu nutzen, etwa durch Schluß von Invarianten der fortgesetzten Funktion auf die ursprüngliche Funktion über den natürlichen Zahlen.
In [Gä01] wurden Fortsetzungen linear rekursiver Funktionen zu analytischen Funktionen betrachtet. In diesem Kapitel haben wir nun untersucht, inwieweit die lineare Rekursion eine Einschränkung gegenüber der allgemeinen primitiven Rekursion darstellt. Dazu haben wir die Klasse der linear primitiv-rekursiven Funktionen über den natürlichen Zahlen definiert und gezeigt, daß diese Klasse die in Polynomialzeit berechenbaren Funktionen umfaßt. Ferner haben wir die Klasse nach oben abgegrenzt, indem wir gezeigt haben, daß die linear primitiv-rekursiven Funktionen einer Wachstumsbeschränkung genügen, und daß die Turmfunktion nicht linear primitiv-rekursiv ist.
Nachdem im ersten Abschnitt des Kapitels lineare Rekursionen betrachtet wurden, gaben wir schließlich noch eine Idee für eine Methode an, mit der eine Klasse nichtlinear rekursiver Funktionen zu analytischen Funktionen fortgesetzt werden kann. Diese Methode führt die Fortsetzung der Funktionen in dieser Klasse auf die Fortsetzung der linear primitiv-rekursiven Funktionen zurück.

Literaturverzeichnis

[BCSS98] Leonore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and Real Computation*. Springer, New York, 1998.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer, Secaucus, NJ, USA, 2006.

[BSS89] Leonore Blum, Michael Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21:1–46, 1989.

[CH99] Thomas Chadzelek and Günter Hotz. Analytic machines. *Theoretical Computer Science*, 219:151–167, 1999.

[Cha98] Thomas Chadzelek. *Analytische Maschinen*. Dissertation, Universität des Saarlandes, Saarbrücken, 1998.

[FL94] Wolfgang Fischer and Ingo Lieb. *Funktionentheorie*. Vieweg, Braunschweig, 1994.

[Gä01] Tobias Gärtner. *Primitive Recursive Functions in one Variable*. Diplomarbeit, Universität des Saarlandes, Saarbrücken, 2001.

[GH01] Tobias Gärtner and Günter Hotz. Primitive recursive functions in one variable. *ECCC*, 8:4, 2001.

[Grz57] Andrzej Grzegorczyk. On the definition of computable real continuous functions. *Fundamenta Mathematicae*, 44:61–71, 1957.

[Hot94] Günter Hotz. Über Berechenbarkeit fraktaler Strukturen. *Abhandlungen der mathematisch-naturwissenschaftlichen Klasse der Akademie der Wissenschaften und der Literatur*, 1994.

[Hur97] Adolf Hurwitz. Sur l'integrale finie d'une fonction entière. *Acta Mathematica*, 20:285–312, 1897.

[HVS95] Günter Hotz, Gero Vierke, and Björn Schieffer. Analytic machines. *ECCC*, 2:25, 1995.

[Ko91] Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, Boston, 1991.

[Mül95] Norbert Th. Müller. Constructive aspects of analytic functions. *Informatik Berichte FernUniversität Hagen*, 190:105–114, 1995.

[Osg02] William F. Osgood. Note on the functions defined by infinite series whose terms are analytisc functions of a complex variable. *Annals of Mathematics*, 2. Ser. 3:25–34, 1902.

[Pen91] Roger Penrose. *The Emperor's new Mind*. Penguin, 1991.

[Rog67] Hartley Rogers. *Theory of recursive functions and effective computability*. McGraw-Hill, New York, 1967.

[SW96] Uwe Storch and Harmut Wiebe. *Lehrbuch der Mathematik. Band 1: Analysis einer Veränderlichen*. Spektrum Akademischer Verlag, 1996.

[Vie96] Gero Vierke. *Berechenbarkeit reellwertiger Funktionen und analytische Berechnungen*. Diplomarbeit, Universität des Saarlandes, Saarbrücken, 1996.

[Wei95] Klaus Weihrauch. A simple introduction to computable analysis. *InformatikBerichte FernUniversität Hagen*, 171, 1995.

[Wei00] Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

[Zie07] Martin Ziegler. Real computability and hypercomputation. Technical Report C-07013, KIAS, July 2007.