Berechenbarkeit reellwertiger Funktionen und analytische Berechnungen

Gero Vierke

22. Oktober 1996

Diplomarbeit nach einem Thema von Prof. Dr. Günter Hotz
in der Technischen Fakultät, Fachbereich 14 – Informatik,
der Universität des Saarlandes, Saarbrücken

Hiermit versichere ich an Eides Statt, daß ich diese Arbeit nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Saarbrücken, im Juli 1995

Inhaltsverzeichnis

Einleitung

R-Maschinen sind im Sinne von [B.S.S] abstrakte Modelle von Rechenmaschinen, die in ihren Speicherzellen je eine Zahl aus einem Ring R ablegen können und deren Arithmetik identisch ist mit der Arithmetik in R. Die Größe des Speichers wird als unbeschränkt angesehen. Die Maschine verfügt über indirekte Adressierung; das heißt, daß der Inhalt einer Speicherzelle, insofern als er eine natürliche Zahl repräsentiert, als Adresse zum Lesen oder Beschreiben einer Speicherzelle verwendet werden kann. Es wird stets angenommen, daß nur endlich viele Speicherzellen zu Beginn der Berechnungen einen anderen Wert als Null enthalten. Eine Berechnung wird nur dann akzeptiert, wenn sie nach endlich vielen Schritten zum Ende kommt.

Wir betrachten in dieser Arbeit nur Maschinen mit $R = \mathbb{Q}$ (Menge der rationalen Zahlen) und $R = \mathbb{R}$ (Menge der reellen Zahlen). Wir erweitern das oben skizzierte Maschinenmodell wie folgt:

- Wir betrachten auch unendliche Berechnungen, das heißt Berechnungen, die nicht abbrechen. An die Stelle des *Haltens* der Maschine tritt hier die Konvergenz einer Ausgabe. Diese Berechnungen nennen wir analytische Berechnungen, Maschinen, auf denen analytische Berechnungen ablaufen, nennen wir analytische Maschinen.

- Wir verallgemeinern die Eingabe für die $\mathbb{Q}$-Maschine in folgendem Sinn: Es wird zugelassen, daß auf einem Eingabeband für die $\mathbb{Q}$-Maschine Zahlen x aus $\mathbb{R}$ stehen. Die $\mathbb{Q}$-Maschine verfügt aber über ein *Präzisions*-register δ derart, daß anstelle von x eine rationale Approximation $[x]_\delta$ gelesen wird mit $|x - [x]_\delta| < \varepsilon$ mit $\varepsilon = 2^{-\delta}$. Eine Vorstellung dahinter ist zum Beispiel ein Abbruch der Dualentwicklung von x nach der δ-ten Stelle hinter dem Komma.

$\mathbb{R}$-analytisch berechenbare Funktionen sind Funktionen, die durch $\mathbb{R}$-analytische Maschinen berechnet werden können. $\mathbb{Q}$-analytische Funktionen sind Funktionen, die durch $\mathbb{Q}$-analytische Maschinen berechnet werden können, wenn ihnen erlaubt ist, die Eingabe mit wachsender Präzision ($\varepsilon \to 0$) beliebig oft zu lesen. Hierdurch dehnen wir den Definitionsbereich der $\mathbb{Q}$-Maschine auf die reellen Zahlen aus. Da die analytischen $\mathbb{Q}$-Maschinen als Ausgaben konvergente Folgen erzeugen, die auch gegen reelle Zahlen konvergieren können, ist der Wertebereich ebenfalls $\mathbb{R}$. Hierdurch wird die Frage zugänglich, inwieweit $\mathbb{R}$-berechenbare oder $\mathbb{R}$-analytische Funktionen auf $\mathbb{Q}$-Maschinen analytisch berechnet werden können.
Über $\mathbb{R}$ stetige Funktionen $f : \mathbb{R} \to \mathbb{R}$ sind durch ihre Einschränkung auf $\mathbb{Q}$, $f|\mathbb{Q}$ eindeutig bestimmt. Ist $f|\mathbb{Q}$ berechenbar, dann ist auch f in gewissem Sinne berechenbar. Man wählt eine Folge rationaler Zahlen $x_1, x_2, x_3, \ldots$ deren Grenzwert die reelle Zahl

x ist. Dann erhält man, indem man f an den x_i auswertet, durch

$$f(x_1), f(x_2), f(x_3), \ldots \to f(x)$$

eine $f(x)$ approximierende Berechnung. Wir werden dieses auf $\mathbb{R}$-berechenbare Funktionen übertragen, indem wir eine Transformation der Programme für $\mathbb{R}$-Maschinen in Programme für die $\mathbb{Q}$-analytischen angeben, die diese Approximation berechnet. Die Schwierigkeit des Beweises besteht darin, daß $\mathbb{R}$-berechenbare Abbildungen im allgemeinen nicht stetig sind. Wie der Darstellungssatz für $\mathbb{R}$-berechenbare Funktionen aus [B.S.S] zeigt, kann man f durch eine abzählbare Summe rationaler Funktionen $f^\sigma : A_\sigma \to \mathbb{R}$ *zusammenstückeln*, wobei die A_σ semi-algebraische Mengen sind. Eine solche Menge A_σ braucht keine rationalen Punkte zu enthalten, was die direkte Übertragung des Verfahrens zur Approximation stetiger Funktionen unmöglich macht. Die Lösung gelingt uns durch eine geeignete Verbreiterung von A_σ, in der wir die rationalen Punkte zur Approximation von x finden. Wir geben einleitend für eine leicht verallgemeinerte Version des Darstellungssatzes einen einfachen Beweis.
In [B.S.S] werden die Einzugsbereiche der Fixpunkte der Iteration einfacher Polynome untersucht und es wird gezeigt, daß das Komplement dieser Einzugsbereiche nicht durch $\mathbb{R}$-Maschinen charakterisiert werden kann. Hierbei wird der Darstellungssatz in Verbindung mit der Tatsache verwendet, daß diese Menge aus kontinuierlich vielen nicht zusammenhängenden Komponenten bestehen kann. Wir zeigen, daß zwei wohlbekannte Kurven, nämlich die Koch'sche Kurve und die quadratfüllende Hilbertkurve nicht $\mathbb{R}$-berechenbar wohl aber $\mathbb{Q}$-analytisch sind. Hierbei verwenden wir ebenfalls den Darstellungssatz in Verbindung mit der Hausdorffdimension des Graphen der Kurven. Die Menge der $\mathbb{Q}$-analytischen Funktionen erweist sich als echt kleiner als die Menge der $\mathbb{R}$-analytischen Funktionen. Während die $\mathbb{R}$-berechenbaren Abbildungen unter der Hintereinanderausführung und unter primitiver Rekursion abgeschlossen sind, gilt das für die $\mathbb{Q}$-analytischen Funktionen nicht. Verschärft man die Definition $\mathbb{Q}$-*analytisch* zu *stark* $\mathbb{Q}$-*analytisch*, indem man verlangt, daß die $\mathbb{Q}$-Maschine stets auch die Präzision der Approximation angibt, dann erhält man die bekannten *berechenbaren reellen Funktionen* im Sinne von Grzegorczyk, deren Klasse unter den genannten Operationen abgeschlossen ist.
Erweitert man die R-Maschinen durch die Hinzunahme von Operationen $f_1, \ldots, f_m$, die stark $\mathbb{Q}$-analytisch sind, dann erhalten wir eine R-Maschine, die wir mit $\mathcal{M}_R(f_1, \ldots, f_m)$ bezeichnen. Wir zeigen, daß die durch $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ berechenbaren Funktionen $\mathbb{Q}$-analytisch sind.
Eine weitere Motivation der Untersuchung erschließt sich aus der folgenden Betrachtung: Computer sind elektronische Systeme, deren Zustand sich über der Kontinuierlichen Zeitachse stetig ändert. Sie erscheinen dem Benutzer zwar als diskrete Maschinen, da sich ihr Inhalt nur auf dem ganzzahligen Zeitgitter mit einer Taktperiode als Einheit zu erkennen gibt. In diesen Zeitpunkten befindet sich das System in *stabilen Zuständen*. Der Übergang von einem dieser Gitterpunkte zu einem folgenden Gitterpunkt besteht im *Aufruf* einer kontinuierlichen Prozedur, die sehr rasch stabil wird. Die Simulation eines solchen Systemes auf einer $\mathbb{Q}$- beziehungsweise $\mathbb{R}$-Maschine bestünde in einer Folge von Aufrufen von stark analytischen Prozeduren. Die sehr rasche Konvergenz dieser Prozeduren reduziert die Frage nach der Stabilität des kontinuierlichen Systemes auf das Halteproblem von $\mathbb{Q}$-Maschinen beziehungsweise $\mathbb{R}$-Maschinen. Der Erfolg in

den Anwendungen des Computers gibt uns somit die Gewißheit, daß das Konzept der $\mathcal{M}_R(f_1, \ldots, f_m)$-Maschinen ein vernünftiges Konzept ist [Ho].
Weiter zeigen wir, daß das analytische Äquivalent des Halteproblems, die Frage, ob die Ausgabe einer analytischen Maschine konvergiert, nicht analytisch entscheidbar ist. Diese Frage ist eng verwand mit dem Stabilitätsproblem dynamischer Systeme, das man in vielen Fällen auf die Frage zurückführen kann, ob die Iteration einer Abbildung S in einer Umgebung eines Punktes kontrahierend ist. Führt man diese Abbildung als Prozedur unserer Maschine ein, so sieht man, daß das Halteproblem unserer Analytischen Maschinen das Stabilitätsproblem solcher dynamischen Systeme mit umfaßt, für die S $\mathbb{R}$-analytisch oder $\mathbb{Q}$-analytisch berechenbar ist. Die Theorie der dynamischen Systeme, die mit den Namen Kolmogoroff, Moser, Arnold [A.A], und Siegel [S] verknüpft ist, kann in gewissen Fällen die Stabilität solcher Systeme nachweisen. Im allgemeinen aber zeigt die Theorie eine sehr sensible Abhängigkeit des Verlaufes vom Anfangspunkt der Iteration, so daß es aus numerischen Gründen für unmöglich angesehen wird, die Stabilität solcher Systeme zu entscheiden. Die Übertragung der aus der Theorie der diskreten Maschinen bekannten Konstruktion zum Nachweis der Nichtentscheidbarkeit des Halteproblems liefert die Einsicht, daß auch bei exakter $\mathbb{R}$-Arithmetik weder mit $\mathbb{R}$- noch mit $\mathbb{Q}$-analytischen Maschinen die Stabilität der Systeme mittels einer universellen Methode entschieden werden kann.
Da die analytischen Funktionen im allgemeinen nicht bezüglich der Hintereinanderausführung abgeschlossen sind, ist es naheliegend, zu untersuchen, wie sich die Mächtigkeit der analytischen Maschinen erweitert, wenn wir mehrere Grenzprozesse, das heißt mehrere unendliche Berechnungen hintereinander zulassen. Dazu führen wir weitere Funktionenklassen ein, indem wir zulassen, daß mehrere analytische Maschinen hintereinandergestellt werden und als Eingabe jeweils die exakten Ergebnisse der vorangestellten Maschinen erhalten.
Abschließend werden wir einen Hierarchiesatz zeigen, der besagt, daß i hintereinandergeschaltete analytische R-Maschinen immer echt mächtiger sind als $i-1$ Maschinen.
Ein Teil der Ergebnisse, die in dieser Arbeit vorgestellt werden, sind bereits in englischer Sprache veröffentlicht [H.V.S]. Weitere Ansätze, sich die berechenbare Analysis zu erschließen, findet man in [Ko] und [Wei].

Gliederung

In Kapitel 1 werden zunächst einige grundlegende Definitionen angegeben. Insbesondere wird auf die Begriffe *mathematischen Maschine* und *Berechnung* eingegangen. Anschließend wird eine präzise Definition der R-Maschine angegeben. Der Befehlssatz für R-Maschinen wird eingeführt, und für eine einfache Funktion wird ein Beispielprogramm angegeben.
In Kapitel 2 werden die R-berechenbaren Funktionen näher untersucht und gegen die herkömmlichen berechenbaren Funktionen und die analytischen Funktionen abgegrenzt. Zunächst werden zu einer Maschine ein Berechnungsgraph und ein Berechnungsbaum definiert. Weiterhin gibt es für jede Berechnung auf dieser Maschine eine Signatur, die für diese Berechnung den dazugehörigen Weg im Berechnungsbaum eindeutig bestimmt. Wie bereits angekündigt, wird dann der Darstellungssatz gezeigt, der besagt, daß die

Definitionsmenge einer R-berechenbaren Funktion f in maximal abzählbar unendlich viele semi-algebraische Teilmengen zerfällt; die Funktion f eingeschränkt auf eine dieser Teilmengen ist eine rationale Funktion.
Damit folgt dann, daß die charakteristischen Funktionen einiger Julia-Mengen sowie die fraktalen Kurven von Koch und Hilbert nicht R-berechenbar sind.
Im Abschnitt 2.3 wird zunächst ein Verfahren angegeben, um in rationalen oder reellen Speicherstellen Wörter über endlichen Alphabeten abzuspeichern. Dieses verwenden wir um zu zeigen, daß die $\mathbb{Q}$-berechenbaren Funktionen und die Turing-berechenbaren identisch sind, und daß jede R-berechenbare Funktion auch mit konstantem Speicherplatz berechnet werden kann.
In Kapitel 3 werden zunächst die analytischen Maschinen formal eingeführt. Besonderes Augenmerk verdient die Tatsache, daß es erforderlich ist, die verschiedenen Rundungsfunktionen gegeneinander abzugrenzen, da sich herausstellt, daß die Menge der $\mathbb{Q}$-analytischen Funktionen von der verwendeten Rundungsfunktion abhängt.
Im Abschnitt 3.2 wird, wie bereits angekündigt, gezeigt, daß jede $\mathbb{R}$-berechenbare Funktion $\mathbb{Q}$-analytisch ist.
Im Abschnitt 3.3 wird gezeigt, daß die $\mathbb{Q}$-analytischen Funktionen mächtiger sind als die $\mathbb{R}$-berechenbaren. Anhand von zwei Beispielfunktionen wird gezeigt, daß die $\mathbb{R}$-analytischen Funktionen mächtiger sind als die $\mathbb{Q}$-analytischen. Hier wird auch deutlich, daß sich die Mengen der Funktionen, die von $\mathbb{Q}$-Maschinen mit unterschiedlichen Rundungsfunktionen berechnet werden können, unterscheiden.
Im Abschnitt 3.4 wird – in Anlehnung an den μ-Operator für ganzzahlige Funktionen – ein μ-Operator für reellwertige Funktionen eingeführt. Es wird gezeigt, daß $\mu(f)$ $\mathbb{Q}$-analytisch ist, wenn f eine Lipschitz-stetige $\mathbb{R}$-berechenbare Funktion ist.
Wie bereits erwähnt, ist die Menge der $\mathbb{Q}$-analytischen Funktionen nicht abgeschlossen gegenüber der Hintereinanderausführung. Deshalb wird in Kapitel 4 die Menge der stark-$\mathbb{Q}$-analytischen Funktionen eingeführt. Es stellt sich heraus, daß die stark-$\mathbb{Q}$-analytischen Funktionen stetig sind. Weiter sind sie gegenüber der primitiven Rekursion beziehungsweise der Hintereinanderausführung abgeschlossen. Anschließend wird die Beziehung zwischen den stark-$\mathbb{Q}$-analytischen und den im Sinne von Grzegorczyk berechenbaren reellen Funktionen geklärt.
Im zweiten Teil dieses Kapitels erweitern wir unser Maschinenmodell, indem wir stark-$\mathbb{Q}$-analytische Prozeduren in R-Maschinen einbinden. Es stellt sich heraus, daß auch die Funktionen, die von diesen erweiterten Maschinen berechnet werden können, $\mathbb{Q}$-analytisch sind.
In Kapitel 5 betrachten wir zunächst das Halteproblem für analytische Maschinen, das heißt die Frage ob die Ausgabe einer analytischen Maschine konvergiert oder divergiert. Mit den aus der Berechenbarkeitstheorie bekannten Verfahren folgt, daß dieses Problem nicht von einer analytischen Maschine entschieden werden kann. Dann wird die Menge der R^i-analytischen Funktionen, das heißt die Menge der Funktionen, die von i hintereinandergestellten analytischen Maschinen berechnet werden können, eingeführt. Abschließend wird gezeigt, daß das Halteproblem für analytische Maschinen R^3-analytisch ist; daraus folgt der Hierarchiesatz, nämlich daß die R^i-analytischen Funktionen die R^{i-1}-analytischen Funktionen echt enthalten.

Danksagungen

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Günter Hotz für die interessante Themenstellung und die persönliche Betreuung dieser Arbeit bedanken.
Ein weiterer Dank gilt Herrn Björn Schieffer für seine kritische Aufmerksamkeit und viele wertvolle Anregungen. Allen Kommilitonen und Mitarbeitern des Lehrstuhls danke ich für das herrvorragende Arbeitsklima.
Ferner möchte ich mich bei meinen Freunden Alexandra Christ und Karsten Fink für geduldiges Zuhören und Gegenlesen bedanken.
Insbesondere aber danke ich meinen Eltern, die mir ein Studium erst ermöglicht haben; ihnen widme ich diese Arbeit.

Kapitel 1

Das Maschinenmodell

1.1 Grundlegende Definitionen

Eine mathematische Maschine besteht aus einer *Konfigurationsmenge* beziehungsweise einem *Zustandsraum* K, der Menge der *Anfangszustände* $K_a \subset K$, der Menge der Endzustände $K_e \subset K$ und einer Zustandsübergangsfunktion (next state function)

$$\Delta : K \to K \text{ mit } \Delta|K_e = 1_{K_e}.$$

(Für die Menge M bezeichnet 1_M die identische Abbildung auf M.) Eine Berechnung der Maschine ist eine Folge von Zuständen

$$b = (k_0, k_1, k_2, \ldots)$$

mit

$$k_{i+1} = \Delta(k_i) \text{ für } i \geq 0.$$

Die Berechnung heißt *regulär*, falls $k_0 \in K_a$ ist und es ein n gibt mit $k_n \in K_e$.
Im Weiteren bezeichnen wir mit C und D zwei beliebige Untermengen von R^*, der Menge aller endlichen Tupel von Zahlen aus R. Um zu erklären, wann eine Funktion

$$f : D \to C$$

auf der Maschine *berechenbar* heißen soll, brauchen wir noch Eingabe- und Ausgabefunktionen. Wir bezeichnen diese Funktionen durch

$$in : D \to K_a \text{ bzw. } out : K \to C;$$

in und *out* sind von f unabhängige, einfache Funktionen, die wir später präzise definieren werden. *out* ist auf ganz K definiert, damit wir auch die Ausgabe einer analytischen Maschine betrachten können.
Nun definieren wir

$$\mathcal{M} := (K, K_a, K_e, \Delta, in, out)$$

als Maschine und sagen

$$f : D \to C$$

ist $\mathcal{M}$ – *berechenbar*, wenn die folgende Bedingung erfüllt ist.

Ist $x \in D$, so ist die durch $k_0 = in(x)$ auf $\mathcal{M}$ erzeugte Berechnung

$$b = (k_0, k_1, k_2, \ldots)$$

regulär. Ist $k_\infty := \lim\limits_{i \to \infty} k_i$ der Fixpunkt, in den b einmündet, dann gilt

$$out(k_\infty) = f(x).$$

Wir erweitern nun die Definition der Berechenbarkeit, indem wir voraussetzen, daß auf K eine Topologie gegeben ist und indem wir unendlich lange, aber im Sinne dieser Topologie konvergierende, Berechnungen zulassen.
Zu diesem Ziel führen wir eine Menge K_t ein mit

$$K_e \subset K_t \subset K.$$

Für eine analytischen Berechnung verlangen wir nun nicht mehr, daß sie in einen Endzustand mündet, sondern, daß sie unendlich oft einen Zustand aus K_t erreicht.
Die Menge K_t erzwingt eine gewisse Regularität in den Berechnungen, die beweistechnische Vorteile erbringt. Man erkennt, daß dieses Konzept identisch ist mit dem von Büchi eingeführten Begriff der unendlichen akzeptierenden Berechnungen von endlichen Automaten.
Damit können wir das Ziel der Arbeit genauer schildern. Die Konfigurationsmengen der Maschinen, die wir betrachten, werden unendliche Produkte von $\mathbb{Q}$ beziehungsweise $\mathbb{R}$ sein. Die Abbildungen Δ werden sich als einfache Erweiterungen von rationalen Funktionen erweisen. Wir werden je nach der zugrundeliegenden Zahlenbasis von $\mathbb{Q}$-analytisch oder $\mathbb{R}$-analytisch berechenbaren Funktionen sprechen. Wir gehen der Frage nach, in wie weit sich die $\mathbb{R}$-analytisch berechenbaren Funktionen durch eine universelle Methode aus den $\mathbb{Q}$-analytisch berechenbaren Funktionen approximieren lassen. Weiter werden wir untersuchen, wie sich die $\mathbb{Q}$-analytisch berechenbaren Funktionen zu den $\mathbb{R}$-berechenbaren Funktionen verhalten und ob das Halteproblem unserer Maschinen durch analytisch berechenbare Funktionen entschieden werden kann.

1.2 Die Definition der R-Maschinen

Wir definieren nun ein Maschinenmodell, das eine Abstraktion eines einfachen Computers ist. Es ist als R-Maschine äquivalent zu den üblichen Computermodellen in der Informatik und insbesondere zu dem in [B.S.S] verwendeten Modell. Figur 1.1 enthält eine anschauliche Beschreibung der Maschine.
Das Eingabeband kann von dem eigentlichen Rechner nur gelesen, das Ausgabeband nur beschrieben werden. Die Variablen oder Felder x_j, y_j, z_j, α können je eine Zahl aus R tragen, γ und δ je eine natürliche Zahl und β eine Zahl aus

$$[1:N] := \{j \in \mathbb{N} | 1 \leq j \leq N\}.$$

N ist die Größe des Programmspeichers. Das δ-Register spielt nur eine Rolle für die analytischen $\mathbb{Q}$-Maschinen. Dort gibt es die Präzision an, mit der die auf dem Eingabeband stehenden reellen Zahlen gelesen werden. Hierauf gehen wir näher ein, wenn wir den Befehlssatz für die R-Maschinen eingeführt haben.

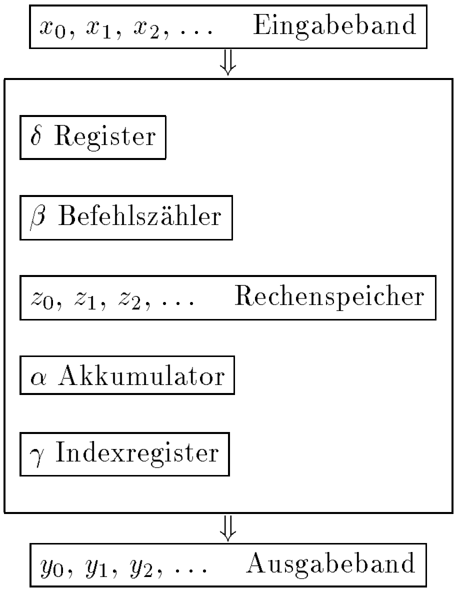


Abbildung 1.1: Schematische Darstellung der R-Maschine

Somit wird der momentane Maschinenzustand durch eine Abbildung

$$\rho : (\alpha, \beta, \gamma, \delta, x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2, \ldots) \to R \times [1:N] \times \mathbb{N} \times \mathbb{N} \times R \times R \times \ldots$$

beschrieben.
Für R-Maschinen fordern wir, daß stets nur endlich viele Variablen einen Wert $\neq 0$ haben. Hierdurch wird die Konfigurationsmenge K definiert.
Ist $k \in K$ ein momentaner Zustand, dann bezeichnen wir die Werte von $\alpha, \beta, \gamma, \delta$ und der anderen Variablen, die zu diesem Zustand gehören durch

$$\alpha(k), \beta(k), \gamma(k), \delta(k), z_j(k), x_j(k), y_j(k).$$

Ergibt sich k aus dem Zusammenhang, so lassen wir die Abhängigkeit von k auch weg.
Nun definieren wir

$$\begin{aligned} K_a := \{k \in K \mid \quad & \alpha = \gamma = \delta = 0,\ \beta = 1,\ x_0 \in \mathbb{N},\ x_j = 0 \text{ für } j > x_0, \\ & z_j = 0,\ y_j = 0 \text{ für alle } j\} \\ K_e := \{k \in K \mid \quad & \beta = N,\ y_0 \in \mathbb{N},\ y_j = 0 \text{ für } j > y_0\}. \end{aligned}$$

An dieser Stelle können wir $in : R^* \to K_a$ durch

$$\begin{aligned} in((r_1, r_2, \ldots, r_n)) \quad := \quad & (\alpha = \gamma = \delta = 0,\ \beta = 1,\ x_0 = n,\ x_i = r_i \text{ für } 0 < i \leq n, \\ & x_j = 0 \text{ für } j > n,\ y_l = z_l = 0 \text{ für } l \geq 0) \end{aligned}$$

und $out : K \to R^*$ durch

$$out(k) := (y_1(k), y_2(k), \ldots, y_{y_0(k)}(k))$$

definieren.
Wir definieren die Menge Ω der Operationen unserer Maschine wie folgt:

1. Wertzuweisungen:

   $\alpha := z_j,\ z_j := \alpha,\ \alpha := x_j,\ y_j := \alpha \qquad$ für $j \in \mathbb{N}$.

2. Arithmetische Operationen:

   $\alpha := \alpha \star z_j,\ \alpha := \alpha \star r \qquad$ für $\star \in \{+, -, \times, /\}, j \in \mathbb{N}$ und $r \in R$.

3. Bedingte Sprunganweisung:

   if $\alpha \sim 0$ then goto m else goto n, $\qquad m, n \in [1 : N]$ und $\sim \in \{=, >\}$

4. Terminierungsanweisung:

   end

5. Indizierte Befehle

   $\alpha := z(\gamma),\ z(\gamma) := \alpha,\ \alpha := x(\gamma),\ y(\gamma) := \alpha,\ \gamma := \alpha,\ \alpha := \gamma,\ \gamma := \gamma \pm 1.$

Die Semantik dieser Anweisungen versteht sich weitgehend von selbst. Die Terminierungsanweisung wird nur als letzte Anweisung des Programmes zugelassen. Wir betrachten ein Programm nur dann als korrekt, wenn für jede Eingabe eine Division durch Null vermieden wird und dem Indexregister nur natürliche Zahlen zugewiesen werden. Dieses sicherzustellen obliegt dem Programmierer; zur Erleichterung definieren wir die zusätzliche Elementaroperation

$$\text{if } \alpha \in \mathbb{N} \text{ then goto } m \text{ else goto } n, \qquad m, n \in [1 : N]$$

die jedoch auch als Makro angesehen und mit den Operationen aus Ω realisiert werden kann.
Ein Programm unserer Maschine ist durch eine Abbildung

$$\pi : [1 : N] \to \Omega$$

gegeben. Damit sind wir in der Lage $\Delta : K \to K$ zu definieren. Wir setzen für $\rho, \rho' \in K$

$$\Delta(\rho) = \rho'$$

genau dann, wenn sich ρ' aus ρ durch die Anwendung der Operation $\pi(\beta(\rho))$ auf ρ ergibt; jeder Befehl verändert nur den Inhalt der Variablen, der durch die Operation ein Wert zugewiesen wird; ausgenommen ist der Stand des Befehlszählers β. Falls keine bedingte Anweisung vorliegt, bewirkt jeder Befehl $\beta(\rho') = \beta(\rho) + 1$. Liegt eine bedingte Sprunganweisung

$$\text{if } \alpha \sim 0 \text{ then goto } m \text{ else goto } n, \qquad m, n \in [1 : N] \text{ und } \sim \in \{=, >\}$$

vor, dann ergibt sich

$$\beta(\rho') = m \text{ falls } \alpha \sim 0 \text{ erfüllt ist,}$$

und

$$\beta(\rho') = n \text{ falls } \alpha \sim 0 \text{ nicht erfüllt ist.}$$

Die durch γ indizierten Befehle

$$\alpha := z(\gamma)$$

bewirken

$$\alpha := z_\gamma.$$

Die anderen indizierten Befehle wirken entsprechend.
Zur Veranschaulichung geben wir ein kleines Beispielprogramm an, das die Betragsfunktion berechnet:

1 $\alpha := x_1$;
2 if $\alpha > 0$ then goto 4 else goto 3;
3 $\alpha := \alpha \cdot (-1)$;
4 $y_1 := \alpha$;
5 $\alpha := 1$;
6 $y_0 := \alpha$;
7 end.

Die hierdurch definierte Maschine bezeichnen wir mit $\mathcal{M}_{|.|}$.
Im Zusammenhang mit der Einführung der analytischen Maschinen werden wir die Lesebefehle

$$\alpha := [x_j]_\delta,\ \alpha := [x(\gamma)]_\delta$$

näher erläutern; diese bewirken das Einlesen einer rationalen Zahl anstelle der eventuell nicht rationalen reellen Zahl mit der Approximationsgüte δ .

Kapitel 2

Charakterisierung der R-berechenbaren Funktionen

Wir wollen nun, wie bereits in der Einleitung angekündigt, den Darstellungssatz für R-berechenbare Funktionen herleiten. Dazu definieren wir zuerst für jede Maschine einen Berechnungsgraphen und einen Berechnungsbaum.

2.1 Der Darstellungssatz für R-berechenbare Funktionen

Wir fassen nun die Zustände des Befehlszählers als Knoten eines Graphen G auf. Zwei Knoten p_1, p_2 von G sind genau dann durch eine von p_1 nach p_2 führende Kante s verbunden, wenn es ein $\rho \in K$ gibt mit $\beta(\rho) = p_1$ und $\beta(\Delta(\rho)) = p_2$. Die Kante s markieren wir mit der Operation $\pi(p_1)$, welche bei dem Übergang von ρ nach $\Delta(\rho)$ zur Anwendung kommt. Im Falle von Abfragen markieren wir die beiden von ρ ausgehenden Kanten durch das den Kanten entsprechende Resultat der Abfrage, also zum Beispiel durch $\alpha = 0$, falls die Abfrage erfüllt ist, und durch $\alpha \neq 0$, falls sie nicht erfüllt ist.
Indem man nun die Wege des Graphen, die von dem Anfangsknoten ausgehen, durch einen Baum darstellt, erhält man den zu der Maschine gehörigen Berechnungsbaum. Jeder regulären Berechnung entspricht also ein Weg vom Anfangsknoten des Baumes zu einem Endpunkt des Baumes. Dies ist im wesentlichen der Rabin'sche Decisiontree. Da wir nur endlich lange Programme zulassen, enthält der Berechnungsgraph immer nur endlich viele Knoten. Die Menge der regulären Berechnungen kann jedoch unendlich groß sein, damit ist es möglich, daß der Berechnungsbaum unendlich viele Knoten enthält.
Aus Gründen, die im Zusammenhang mit dem Beweis des Darstellungssatzes klar werden, verfeinern wir das Flußdiagramm wie folgt:
Die Menge der Knoten besteht aus $[1 : N] \times \mathbb{N}$, wenn $[1 : N]$ die Menge der möglichen Befehlszählerstände der Maschine ist. Von dem Knoten (p_1, n_1) geht genau dann eine Kante nach (p_2, n_2), wenn es ein $\rho_1 \in K$ gibt mit

$$\beta(\rho_1) = p_1, \quad \gamma(\rho_1) = n_1, \quad \rho_2 = \Delta(\rho_1)$$

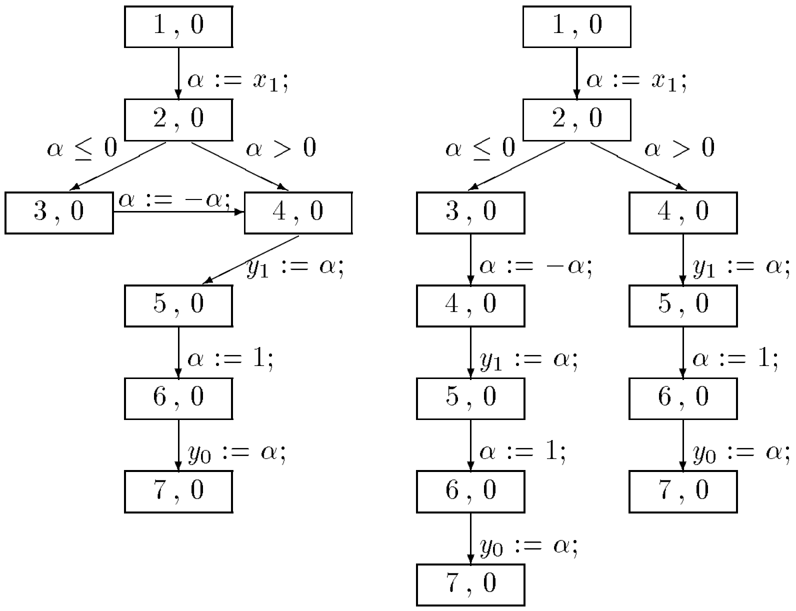


Abbildung 2.1: Berechnungsgraph und Berechnungsbaum von $\mathcal{M}_{|.|}$

und

$$\beta(\rho_2) = p_2, \quad \gamma(\rho_2) = n_2.$$

Wir nehmen also zur Charakterisierung des Betriebszustandes der Maschine zu dem Befehlszählerstand den Inhalt des Indexregisters hinzu. Dieser zur Maschine $\mathcal{M}$ gehörige Graph sei $G_\gamma(\mathcal{M})$. Nun konstruieren wir zu diesem Graph wie zuvor den Überlagerungsbaum, der die vom Anfangsknoten von $G_\gamma(\mathcal{M})$ ausgehenden Berechnungspfade enthält. Dieser Baum sei mit $\mathcal{L}_\gamma(\mathcal{M})$ bezeichnet. Figur 2.1 stellt $G_\gamma(\mathcal{M}_{|.|})$ und $\mathcal{L}_\gamma(\mathcal{M}_{|.|})$ anschaulich dar.

Ist $\rho \in K_a$, also ein Anfangszustand von $\mathcal{M}$, dann bezeichnen wir durch $\sigma(\rho)$ den zu ρ gehörigen Berechnungspfad in $\mathcal{L}_\gamma(\mathcal{M})$. Ist die zu ρ gehörige Berechnung regulär, dann endet $\sigma(\rho)$ in einem Knoten von $\mathcal{L}_\gamma(\mathcal{M})$ der den Knoten N von G überlagert. Nun gilt das

Lemma 1: Sind $\rho_0, \bar{\rho}_0 \in K_a$, und ist $\sigma(\rho_0) = \sigma(\bar{\rho}_0)$, dann beziehen sich die Instruktionen der Maschine in jedem der Folgezustände $\rho_i := \Delta^i(\rho_0)$ und $\bar{\rho}_i := \Delta^i(\bar{\rho}_0)$ auf die gleichen Speicherzellen oder Register. Insbesondere hängen also alle Resultate von den gleichen Eingabevariablen ab.

Der Beweis ergibt sich durch vollständige Induktion nach i.

Wir ordnen jedem regulären Weg σ aus $\mathcal{L}_\gamma(\mathcal{M})$ die Menge V_σ von Variablen (=Speicherzellen) zu, aus denen ausgelesen wird. V_σ enthält also i.a. Variablen $\alpha, \gamma, x_0, \ldots x_n, z_0, \ldots z_m$. V_σ enthält nicht Variablen aus $y_0, y_1, \ldots$, da $\mathcal{M}$ in diese nur schreibt. Sei S_σ die Menge der Speicherzellen in die im Verlauf der Berechnung geschrieben wird.

Nun ordnen wir jedem von der Wurzel ausgehenden Weg σ von $\mathcal{L}_\gamma(\mathcal{M})$ in Verbindung mit jeder Speicherstelle s, in die im Verlauf der Berechnung ein Wert eingeschrieben wird, eine rationale Funktion zu:

$$f_s^\sigma : D \to R \text{ für } s \in S_\sigma,\ \sigma \in \mathcal{L}_\gamma(\mathcal{M})$$

Sie ist induktiv nach der Länge des Weges σ definiert. Für $|\sigma| = 0$ setzen wir

$$f_s^\sigma(v) := 0 \text{ (Nullfunktion) für alle } v \in D \text{ und } s \in S_\sigma.$$

Sei nun $|\sigma| = n$ und sei σ' Präfix von σ mit $|\sigma'| = n-1$ und sei π_n die Beschriftung der n-ten Kante von σ. Dann definieren wir

$$f_s^\sigma := \begin{cases} f_s^{\sigma'} & \text{falls } \pi_n \text{ der Variablen } s \text{ keinen Wert zuweist,} \\ f_t^{\sigma'} & \text{falls } \pi_n = (s := t), \\ f_\alpha^{\sigma'} \star f_t^{\sigma'} & \text{falls } \pi_n = (s := \alpha \star t), \\ f_\alpha^{\sigma'} \star r & \text{falls } \pi_n = (s := \alpha \star r), r \in R, \\ f_\gamma^{\sigma'} \pm 1 & \text{falls } s = \gamma \text{ und } \pi_n = (\gamma := \gamma \pm 1). \end{cases}$$

Hierin ist stets $\star \in \{+, -, \times, /\}$, und im Falle von indizierten Variablen ist s beziehungsweise t die Variable, die durch Indizierung festgelegt wird; wenn zum Beispiel $\pi_n = (\alpha := x(\gamma))$ und $\sigma' = (\ldots, (n-1, 5))$ erhalten wir $t = x_5$ und $f_\alpha^\sigma := f_{x_5}^{\sigma'}$.
Aufgrund von Lemma 1 gilt für Berechnungen mit den Anfangszuständen $\rho_0, \bar{\rho}_0 \in K_a$ und $\sigma(\rho_0) = \sigma(\bar{\rho}_0)$, daß die Resultate dieser Berechnungen durch die gleichen rationalen Funktionen f_s^σ aus den Eingabewerten $(x_1, \ldots, x_m)$ erhalten werden können.
Wir wollen nun die Eingaben, die zu gleichen Berechnungspfaden führen, algebraisch charakterisieren. Sei also σ ein gegebener Berechnungspfad in $\mathcal{L}_\gamma(\mathcal{M})$. Wir bezeichnen mit Σ^σ die Menge der Präfixe σ' von σ, die zu den Abfragen $\alpha = 0$ oder $\alpha > 0$ oder Operationen $\gamma := \ldots$ führen.
Wir sehen, daß für alle $\sigma' \in \Sigma^\sigma$ die rationalen Funktionen $f_\alpha^{\sigma'}$ dann, je nachdem wie die entsprechende Abfrage auf dem Pfad σ ausgegangen ist, die Gleichungen beziehungsweise Ungleichungen

$$f_\alpha^{\sigma'}(x_1, \ldots, x_m) = 0, \quad \neq 0, \quad > 0, \quad \leq 0$$

erfüllen; entsprechend ist

$$f_\gamma^{\sigma'}(x_1, \ldots, x_m) = k$$

erfüllt, wenn $\sigma' = (\ldots, (n, k))$ gilt. Diese (Un-)Gleichungen bezeichnen wir mit $F^{\sigma'}(x_1, x_2, \ldots, x_m)$.
Man erkennt, daß dieses System von Gleichungen und Ungleichungen den Pfad σ eindeutig bestimmt. Dann ist

$$A_\sigma := \{(x_1, \ldots, x_m) \in D \mid F^{\sigma'}(x_1, \ldots, x_m) \text{ ist erfüllt für alle } \sigma' \in \Sigma^\sigma\}$$

die Menge der Lösungen der zu dem Pfad σ gehörigen algebraischen Bedingungen.
Die Nenner der rationalen Funktionen enthalten nach der Voraussetzung über unsere Programme keine Nullstellen. Somit geht jede rationale Ungleichung bei der Multiplikation mit ihrem Nenner in eine algebraische Ungleichung für A_σ über, so daß

wir ein System $C_\sigma(x_1, \ldots, x_m)$ polynomialer Gleichungen und Ungleichungen erhalten, die A_σ eindeutig bestimmen. A_σ ist also eine semi-algebraische Menge. Wir bezeichnen mit $a_\sigma : R^* \to \{0,1\}$ die charakteristische Funktion von A_σ als Teilmenge von $R^* = R \cup R^2 \cup R^3 \cup \ldots$. Weiter sei $S_{\mathcal{M}}$ die Menge der terminierenden Berechnungspfade in $\mathcal{L}_\gamma(\mathcal{M})$. Wir können dann unser Resultat in dem folgenden Satz zusammenfassen.

Satz 1 (Darstellungssatz): Ist $f : D \to R^*$ eine R-berechenbare Funktion, dann gilt

$$\begin{array}{ll} (1) & D = \bigcup_{\sigma \in S_{\mathcal{M}}} A_\sigma \\ (2) & f = \sum_{\sigma \in S_{\mathcal{M}}} a_\sigma \cdot (f^\sigma_{y_1}, f^\sigma_{y_2}, \ldots, f^\sigma_{y_{f^\sigma_{y_0}}}) \end{array}$$

Teil (1) des Darstellungssatzes wurde in [B.S.S] für $D \subset \mathbb{R}^k$ gezeigt. Dieser Satz ist selbst eine Verallgemeinerung eines Darstellungssatzes von Rabin für Berechnungen von beschränkter Länge, der die Basis für die Ableitung unterer Schranken für die Entscheidung $x \in D$ in Abhängigkeit von der Betti-Zahl $\beta_0(D)$ bildet [BenOr].
Man kann diesen Darstellungssatz dazu verwenden, um zu zeigen, daß gewisse Funktionen nicht $\mathbb{R}$-berechenbar sind.

2.2 Beispiele nicht $\mathbb{R}$-berechenbarer Funktionen

In [B.S.S] wurde der Darstellungssatz verwendet, um zu zeigen, daß die charakteristischen Funktionen einer Klasse von Julia-Mengen nicht $\mathbb{R}$-berechenbar sind. Es handelt sich dort um die Abbildung $\rho(z) = z^2 + c$ mit $c \in \mathbb{C}$ und z variabel über $\mathbb{C}$. Betrachtet man den Fall $|c| > 4$ und ∞ als Attraktor, dann sieht man leicht, daß die Menge der Punkte, die bei der Iteration von ρ nicht durch den unendlich fernen Punkt ∞ angezogen werden, also die Julia-Menge von (ρ, ∞), überabzählbar viele Zusammenhangskomponenten besitzt.
Wäre die charakteristische Funktion dieser Menge $\mathbb{R}$-berechenbar, dann könnte diese Menge aufgrund des Darstellungssatzes aber nur abzählbar unendlich viele Zusammenhangskomponenten besitzen.
In [Ho] wurde der Darstellungssatz verwendet um für Kurven

$$\mu[0;1] \to \mathbb{R}^2$$

zu zeigen, daß sie nicht $\mathbb{R}$-berechenbar sind. Hierzu wurde verwendet, daß die Hausdorffdimension von μ maximal 1 ist, wenn μ eine rationale Funktion ist, oder aus höchstens abzählbar unendlich vielen rationalen Funktionen zusammengestückelt ist.
Der Graph der bekannten, überall stetigen aber nirgends differenzierbaren Koch'schen Kurve hat die Hausdorffdimension

$$\frac{1}{(1 - \log_2(\sqrt{5} - 1))} \approx 1,2\,.$$

Also gibt es keine Parametrisierung der Kurve die $\mathbb{R}$-berechenbar ist. Ebenso folgt aus dem gleichen Grund, daß die ein Quadrat füllende Hilbertkurve, die also die Hausdorffdimension 2 hat, nicht $\mathbb{R}$-berechenbar ist.

2.3 Weitere Sätze

In diesem Kapitel werden wir eine Methode vorstellen, die es ermöglicht, in rationalen und reellen Zahlen beliebig lange Wörter über einem endlichen Alphabet abzulegen. Wir werden skizzieren, wie eine R-Maschine eine Turing-Maschine simulieren kann; anschließend werden wir zeigen, daß alle R-berechenbaren Funktionen auch auf einer R-Maschine mit endlichem Speicher berechnet werden können.
Im letzten Kapitel werden wir dieses Konzept so erweitern, daß wir mittels analytischer Berechnungen unendliche Wörter in reellen Zahlen erzeugen können.
Sei also $\Sigma = \{s_1, s_2, \ldots, s_n\}$ ein endliches Alphabet der Mächtigkeit $n = |\Sigma|$; $\sharp = s_1 \in \Sigma$ bezeichnen wir als das Blanksymbol. Wir definieren die Bijektion

$$\tau : \Sigma \to [0 : n-1], \text{ so daß } \tau(s_i) := i - 1;$$

und erweitern sie zu

$$\tau^* : \Sigma^* \to [0 : n-1]^*, \text{ durch } \tau^*(s_{i_1}, \ldots, s_{i_m}) := (\tau(s_{i_1}), \ldots, \tau(s_{i_m})).$$

Weiter definieren wir $\hat{\tau} : \Sigma^* \to [0, 1[$:

$$\hat{\tau}(s_{i_1} \circ s_{i_2} \circ \ldots \circ s_{i_m}) := \frac{i_1}{n} + \frac{i_2}{n^2} + \ldots + \frac{i_m}{n^m}$$

mit $i_1, i_2, \ldots, i_m \in [1 : n]$. Nun fassen wir eine reelle Zahl aus $\mathrm{Bild}(\hat{\tau}) \subset [0, 1[$ als Wort über Σ auf, indem wir die Zifferndarstellung zur Basis n betrachten. Man kann diese Zahl nun entweder als unendliches Wort betrachten, das durch unendlich viele Blanks abgeschlossen wird, oder als endliches Wort, in diesem Fall interpretiert man die unendliche Folge von Blanks als Wortende.
Im folgenden definieren wir auf den Zahlen aus $\mathrm{Bild}(\hat{\tau})$ die grundlegenden Stringoperationen. Sei $s = s_{i_1} \circ \ldots \circ s_{i_m} \in \Sigma^*$ und $r = \hat{\tau}(s)$. Dann sei

$$\mathrm{first}(r) := \lfloor r \cdot n \rfloor = i_1$$

weiter sei

$$\mathrm{rest}(r) := r \cdot n - \lfloor r \cdot n \rfloor = \frac{i_2}{n} + \ldots + \frac{i_m}{n^{m-1}}$$

und die zweistellige Funktion

$$\mathrm{concat}(j, r) := \frac{j + r}{n} \text{ für } j \in [1 : n], r \in [0, 1[.$$

Offensichtlich gilt $\mathrm{concat}(\mathrm{first}(r), \mathrm{rest}(r)) = r$. Diese drei Grundoperationen ermöglichen bereits alle weiteren, üblichen Stringoperationen, insbesondere genügen sie, um eine Turingmaschine zu simulieren.
Später werden wir noch brauchen, daß das Zerschneiden und Zusammenfügen von größeren Strings mit konstantem Speicheraufwand realisierbar ist. Es sei

$$\mathrm{left}(r, j) := \frac{\lfloor r \cdot n^j \rfloor}{n^j}$$

das Teilwort von s, das die ersten j Zeichen enthält, und

$$\mathrm{right}(r, j) := r \cdot n^j - \lfloor r \cdot n^j \rfloor$$

das Teilwort von s, das alle Zeichen nach dem j-ten Zeichen enthält, und

$$\mathrm{concatenate}(r_1, r_2) := r_1 + \frac{r_2}{n^{|r_1|}}$$

die Konkatenation der Wörter s_1 und s_2. Man kann sich schnell klar machen, daß diese Operationen mit konstantem Speicheraufwand auskommen.
Wir betrachten nun die Turingmaschine $M = (K, \Sigma, \delta, s)$. Hierbei sei K eine endliche Zustandsmenge, die den Haltezustand h nicht enthält, Σ sei ein endliches Alphabet, das das Blanksymbol $\sharp$ enthält, $s \in K$ sei der Anfangszustand und δ die Übergangsfunktion

$$\delta : K \times \Sigma \to (K \cup \{h\}) \times (\Sigma \cup \{L, R\}).$$

Die Maschine habe ein zweiseitig unendliches Band, das mit Symbolen aus Σ beschrieben werden kann; anfangs sind fast alle Felder mit $\sharp$ beschrieben.
Wir geben nun eine Simulation dieser Turingmaschine durch eine R-Maschine an, die nur endlich viele Speicherzellen verwendet.
Auch in diesem Fall sei τ eine Abbildung von Σ nach $[0 : n-1]$, $n = |\Sigma|$ wobei $\tau(\sharp) = 0$.
Wir betrachten das Band der Turingmaschine auf beiden Seiten des Schreiblesekopfes als Wörter $p, q \in \Sigma^\infty$:

$$\ldots, p_3, p_2, p_1, q_1, q_2, q_3, \ldots$$

Hierbei interpretieren wir q_1 als das Bandfeld, auf dem der Schreiblesekopf steht. Die q_i mit $i > 1$ sind die Felder rechts vom Schreiblesekopf, die p_i mit $i > 0$ sind diejenigen auf der linken Seite.
Dann kodieren wir das Band wie folgt in die beiden Speicherstellen:

$$z_0 := \frac{\tau(p_1)}{n} + \frac{\tau(p_2)}{n^2} + \frac{\tau(p_3)}{n^3} + \ldots$$

$$z_1 := \frac{\tau(q_1)}{n} + \frac{\tau(q_2)}{n^2} + \frac{\tau(q_3)}{n^3} + \ldots$$

Man beachte, daß z_0 und z_1 immer rationale Zahlen bleiben, da immer nur endlich viele Bandfelder mit Werten $\neq \sharp$ beschrieben sind. Nun ordnen wir jedem Zustand $k \in K$ eine Programmsequenz der R-Maschine zu, die folgendes leistet:

> Der Wert von q_1 =first(z_1) wird ins α-Register geladen.
>
> Es sei $(k', \rho) := \delta(k, \tau^{-1}(q_1))$.
>
> Wenn nun $\rho \in \Sigma$, dann wird z_1 durch concat($\tau(\rho)$,rest(z_1)) ersetzt.
>
> Wenn $\rho = R$, dann wird z_0 durch concat(first(z_1), (z_0)) und z_1 durch rest(z_1) ersetzt.
>
> Wenn $\rho = L$, dann wird z_1 durch concat(first(z_0), (z_1)) und z_0 durch rest(z_0) ersetzt.
>
> Schließlich wird zu der Verarbeitung des Zustandes k' verzweigt.

Da die Turingmaschine nur endlich viele Zustände annehmen kann und da Σ endlich ist, ist dieses Programm ebenfalls endlich. Auch hier sieht man schnell, daß der Aufwand an Speicher konstant bleibt. Bevor man mit der Simulation der Turingmaschine beginnt, wird man die Eingabe in z_0 und z_1 kodieren, und wenn die Simulation beendet ist, wird man das Ergebnis dekodieren und auf das Ausgabeband schreiben.
Hieraus folgt: Die $\mathbb{Q}$-berechenbaren Funktionen enthalten die Turing-berechenbaren.
Da wir bei der Definition der $\mathbb{Q}$-Maschinen nichts zugelassen haben, was die Möglichkeiten einer Turingmaschine übersteigt, folgt die Gleichheit:

Satz 2: Die Menge der $\mathbb{Q}$-berechenbaren Funktionen und die Menge der Turing-berechenbaren Funktionen sind identisch.

Satz 3: Ist $f : D \to C, D \subset \mathbb{R}^n$ und f $\mathcal{M}$-berechenbar, dann gibt es eine Maschine $\mathcal{M}'$, die f mit konstantem Speicherplatz berechnet.

Da die $\mathbb{Q}$-berechenbaren Funktionen Turing-berechenbar sind, und eine $\mathbb{Q}$-Maschine eine Turingmaschine mit konstantem Speicheraufwand simulieren kann, ist der Satz in diesem Falle offensichtlich.
Wir geben hier einen Beweis an, der den Satz auch für $\mathbb{R}$-Maschinen zeigt. Der Beweis lehnt sich an den Darstellungssatz an und baut darauf auf, daß R-Maschinen rationale Funktionen berechnen, und daß rationale Funktionen immer so dargestellt werden können, daß sie mit drei Speicherstellen ausgerechnet werden können.
Für eine Berechnung $b = (k_0, k_1, \ldots k_n)$ definieren wir die Folge

$$\varsigma(b) = ((\beta(k_0), \gamma(k_0)), (\beta(k_1), \gamma(k_1)), \ldots, (\beta(k_n), \gamma(k_n)))$$

ς enthält dieselbe Information wie die Signatur σ, jedoch erlaubt ς uns, die Berechnung von hinten zu betrachten.

Lemma 2: Sei $b = (k_0, k_1, \ldots, k_l, \ldots, k_n)$ eine reguläre Berechnung und seien $\varsigma(k_0, \ldots, k_l)$ sowie das dazugehörige Programm Π bekannt. Dann lassen sich die Werte $z_i(k_l), y_i(k_l), \alpha(k_l)$ $\forall i \in \mathbb{N}$ mit konstantem Speicheraufwand berechnen.

Beweis von Lemma 2: Wir beschränken uns ohne Beschränkung der Allgemeinheit auf die Berechnung von α. Wir schreiben $b' = (k_0, \ldots, k_l)$. $\varsigma(k_0, \ldots, k_l)$ und Π seien als Strings in jeweils einer Speicherzelle abgelegt.

$f_\alpha^{b'}$ ist nun die Funktion, die die Eingabe auf $\alpha(k_l)$ abbildet. Wir erzeugen nun in einer Speicherzelle einen String Ψ, der die rationale Funktion $f_\alpha^{b'}$ in gewohnter mathematischer Schreibweise beschreibt, so daß $f_\alpha^{b'} = \psi(\Psi)$ wobei ψ den String kanonisch auf eine rationale Funktion abbildet.

Zu jeder rationalen Funktion existiert eine normierte Form

$$\frac{x_{j_1} \cdot x_{j_2} \cdot \ldots \cdot x_{j_k} + \ldots + x_{j_l} \cdot \ldots \cdot x_{j_m}}{x_{j_n} \cdot \ldots \cdot x_{j_o} + \ldots + x_{j_p} \cdot \ldots \cdot x_{j_q}},$$

die mit nur drei Speicherstellen ausgerechnet werden kann, wie man sich leicht klar macht. Wir zeigen nun, daß Ψ in dieser Form erzeugt werden kann.

Dazu initialisieren wir Ψ mit "$[\alpha]/[1]$" und gehen ς von hinten nach vorne durch. Für jeden Befehl der Form

$$s := t \text{ mit } s \in \{z_i, y_i, \alpha\}, t \in \{z_i, y_i, \alpha, \alpha \star z_i | \star \in \{+, -, \cdot, /\}\}$$

ersetzen wir alle Vorkommnisse von "s" in Ψ durch "(t)".

Da der String jetzt nicht mehr in der obigen, normierten Form ist, werden wir ihn wie folgt umformen.

- Falls $t \in \{z_i, y_i, \alpha\}$, müssen nur die Klammern entfernt werden, um Ψ wieder in normierte Form zu bringen.
- Falls $t = \alpha \cdot z_i$ mit $i \in \mathbb{N}$, werden alle Vorkommnisse von "$(\alpha \cdot z_i)$" durch "$\alpha \cdot z_i$" ersetzt.
- Wenn $t = \alpha + z_i$, wird das Distributivgesetz auf das Monom, das "$(\alpha + z_i)$" enthält, angewandt. Das heißt

  $$\pm v_1 \cdot v_2 \cdot \ldots \cdot v_l \cdot (\alpha + z_i) \cdot v_{l+1} \cdot \ldots \cdot v_m \text{ mit } v_j \in \{x_p, z_p, \alpha, (\alpha + z_i)\}$$

  wird zu

  $$\pm v_1 \cdot \ldots \cdot v_l \cdot \alpha \cdot v_{l+1} \cdot \ldots \cdot v_m \pm v_1 \cdot \ldots \cdot v_l \cdot z_i \cdot v_{l+1} \cdot \ldots \cdot v_m.$$

  Auch diese Umformung kann durch Anwendung der oben definierten Stringoperationen vorgenommen werden. Man sieht, daß durch mehrfache Anwendung dieser Regel auch mehrfaches Auftreten von "$(\alpha+z_i)$" in einem Monom aufgelöst werden kann.
- Im Falle $t = \alpha - z_i$, verfährt man analog.
- Falls $t = \alpha/z_i$, wird der ganze Bruch mit z_i erweitert. "(α/z_i)" wird durch "α" ersetzt und an alle weiteren Monome (im Nenner wie im Zähler) wird "$\cdot z_i$" angehängt. Auch diese Regel läßt sich auf mehrere gleichzeitige Ersetzungen anwenden.

Sind wir durch, ersetzen wir alle Variablen aus dem Rechenspeicher sowie α und γ durch 0, da sie zu Begin der Berechnung 0 enthalten. Induktiv folgt nun, daß $f_\alpha^{b'} = \psi(\Psi)$, und daß Ψ in der normierten Form ist, die mit nur drei Speicherstellen ausgerechnet werden kann.

Beispiel: Zur Veranschaulichung des beschriebenen Verfahrens geben wir hier das Vorgehen für eine kurze Programmsequenz an:

$$\begin{array}{lrcl}
 & \text{“}[\alpha] & / & [1]\text{”} \\
\pi(4) = (\alpha := \alpha \cdot x_4); & \text{“}[(\alpha \cdot x_4)] & / & [1]\text{”} \\
 & \text{“}[\alpha \cdot x_4] & / & [1]\text{”} \\
\pi(3) = (\alpha := \alpha + x_3); & \text{“}[(\alpha + x_3) \cdot x_4] & / & [1]\text{”} \\
 & \text{“}[\alpha \cdot x_4 + x_3 \cdot x_4] & / & [1]\text{”} \\
\pi(2) = (\alpha := \alpha / x_2); & \text{“}[(\alpha / x_2) \cdot x_4 + x_3 \cdot x_4] & / & [1]\text{”} \\
 & \text{“}[\alpha \cdot x_4 + x_3 \cdot x_4 \cdot x_2] & / & [1 \cdot x_2]\text{”} \\
\pi(1) = (\alpha := x_1); & \text{“}[(x_1) \cdot x_4 + x_3 \cdot x_4 \cdot x_2] & / & [1 \cdot x_2]\text{”} \\
 & \text{“}[x_1 \cdot x_4 + x_3 \cdot x_4 \cdot x_2] & / & [1 \cdot x_2]\text{”}
\end{array}$$

Beweis von Satz 3: Aus dem Lemma folgt der Satz, wenn sich ς mit konstantem Speicheraufwand berechnen läßt. Wir zeigen dies durch Induktion nach der Länge der Berechnung.

Für eine Berechnung der Länge 1 ist dies offensichtlich. Wir nehmen also an: $\varsigma(k_0, \ldots, k_i)$ ist bekannt.

Wir berechnen nun die Werte von $\beta(k_{i+1})$ und $\gamma(k_{i+1})$. Der Wert des Indexregisters kann leicht verfolgt werden für Befehle der Form $\gamma := \gamma \pm 1$; im Falle einer Zuweisung $\gamma := \alpha$ kann α berechnet werden, da nach Annahme $\varsigma(k_0, \ldots, k_i)$ bekannt ist.

Falls $\pi(k_i)$ keine Sprunganweisung ist, so ist $\beta(k_{i+1}) = \beta(k_i) + 1$. Wenn es eine Sprunganweisung der Form

if $\alpha \sim 0$ then goto m else goto n

ist, wird wieder α berechnet, was nach Voraussetzung möglich ist; dann ist

$$\beta(k_{i+1}) := \begin{cases} m & \text{falls } \alpha \sim 0 \text{ gilt,} \\ n & \text{falls } \alpha \sim 0 \text{ nicht gilt.} \end{cases}$$

Somit ist der Satz gezeigt.

Kapitel 3

Analytische Maschinen

Man sieht leicht, daß die Menge der R-berechenbaren Funktionen unter der Hintereinanderausführung und dem cartesischen Produkt abgeschlossen ist. Hierzu braucht man nur die Ausgaberegister einer Maschine $\mathcal{M}_f$ mit den Eingaberegistern einer Maschine $\mathcal{M}_g$ zu identifizieren. Berechnet $\mathcal{M}_f$ die Abbildung f und $\mathcal{M}_g$ die Abbildung g, dann berechnet die so konstruierte Maschine die Abbildung $g \circ f$. Setzt man für eine R-berechenbare Abbildung $f : D \to D$

$$f^{(1)} := f \text{ und } f^{(n+1)} := f \circ f^{(n)} \text{ für } n = 1, 2, \ldots,$$

dann erhält man *Iterationen* von f, die nach obigen Ausführungen ebenso R-berechenbar sind. Hieraus erhält man durch *primitive Rekursion*

$$g(n, x) = f^{(n)}(x)$$

eine Funktion, die ebenfalls R-berechenbar ist, wie man mit Standardverfahren der Theorie der Berechenbarkeit zeigt. Es ist eine interessante Frage, wie man diese Funktion in natürlicher Weise von $\mathbb{N} \times D$ auf $\mathbb{R} \times D$ oder eine möglichst große Teilmenge $D' \subset \mathbb{R} \times D$ fortsetzen soll. Jede $\mathbb{R}$-Maschine, die $g(n, x)$ berechnet, legt zwar eine eigene solche Fortsetzung nahe, wird aber i.a. nicht als natürlich angesehen werden. Hier verfolgen wir die Frage, wie sich der Umfang der berechenbaren Funktionen erweitert, wenn man unendlich viele Iterationen zuläßt. Das heißt, wir interessieren uns für die Abbildung, die man durch die Grenzwertkonstruktion

$$\lim_{n\to\infty} f^{(n)}(x)$$

aus einer berechenbaren Funktion f erhält. Diese Frage ist eine Verallgemeinerung einer von Julia [J] aufgeworfenen Frage. Uns interessieren in diesem Zusammenhang zwei Probleme:

- Lassen sich die $\mathbb{R}$-analytischen Funktionen durch $\mathbb{Q}$-analytische Funktionen approximieren? Diese Frage interessiert uns, da wir im Gegensatz zu $\mathbb{R}$-Maschinen in der Lage sind, $\mathbb{Q}$-Maschinen zu bauen.

- Gibt es eine universelle Methode die Konvergenz zu entscheiden? Wie schon erwähnt, hängt diese Frage mit dem Stabilitätsproblem dynamischer Systeme zusammen.

Zunächst definieren wir genau, was wir unter $\mathbb{Q}$-analytisch und $\mathbb{R}$-analytisch verstehen wollen. Die Definitionen für $\mathbb{Q}$-analytisch und $\mathbb{R}$-analytisch unterscheiden sich, wie schon angekündigt wurde.

3.1 Definition der analytischen Maschinen

Definition 1: Ist $\mathcal{M}$ eine R-Maschine, dann heißt die Berechnung

$$\rho_0, \rho_1, \rho_2, \ldots$$

analytisch :$\Leftrightarrow$ es gelten (1) und (2).

$$(1) \quad \rho_i \in K_t \text{ für unendlich viele } i$$

$$(2) \quad out(\rho_i) \text{ konvergiert für } i \to \infty.$$

Definition 2: $\mathcal{M}$ heißt δ-$\mathbb{Q}$-Maschine :$\Leftrightarrow$ Es gelten (1),(2),(3).

(1) $\mathcal{M}$ hat die gleiche Struktur wie eine R-Maschine und enthält die gleichen Register. Die Variablen x_j des Eingabebandes können Zahlen aus $\mathbb{R}$ tragen; alle anderen Variablen der Maschine, also $z_0, z_1, \ldots$ und α sind nur für Zahlen aus $\mathbb{Q}$ zugelassen. Die Variablen δ und γ sind nur für natürliche Zahlen zugelassen, β nur für $[0:N]$.

(2) Für $\mathcal{M}$ sind alle Operationen der R-Maschine erklärt. Abweichend von der Definition der R-Maschine bleibt die Maschine bei Stand $\beta = N$ des Befehlszählers nicht stehen, sondern springt zur Adresse $\beta = 1$. Adresse 1 darf jedoch nur so angesprungen werden. Der Befehl in Adresse 1 lautet stets

$$\delta := \delta + 1;$$

dieser Befehl kommt nur an dieser Stelle im Programm vor. In Adresse N steht der Befehl

$$init;$$

dieser Befehl setzt die Register α und γ sowie alle z_i auf Null und β auf 1. Der Befehlsvorrat der Maschine wird ergänzt durch

$$\alpha := \delta$$

sowie

$$\alpha := [x_i]_\delta \text{ und } \alpha := [x(\gamma)]_\delta.$$

Die Wirkungen von $\delta := \delta + 1$ und $\alpha := \delta$ sind klar. $[\,]_\delta$ ist eine $\mathbb{R}$-berechenbare Funktion von $\mathbb{R}$ nach $\mathbb{Q}$, für die gilt:

$$[x]_\delta \in \mathbb{Q} \text{ und } |x - [x]_\delta| < 2^{-\delta}.$$

(3) Die Definition von K ergibt sich aus (1). Wir definieren K_a wie früher. K_t wird so definiert wie früher K_e, es handelt sich jedoch in diesem Fall um keine

Endzustände. Zusätzlich zeichnen wir eine weitere Menge K_b aus; dies ist durch die spezielle Rolle des Rundungsregisters δ motiviert. Wir setzen

$$K_b := \{k \mid \alpha = \gamma = 0,\ \beta = 1,\ \delta \in \mathbb{N},\ x_0 \in \mathbb{N},\ x_j = 0 \text{ für } j > x_0,\ z_j = 0\ \ \forall j\}.$$

K_b unterscheidet sich also von K_a dadurch, daß wir zulassen, daß das δ-Register einen von 0 verschiedenen Wert annimmt.

Definition 3: Ist $\mathcal{M}$ eine δ-$\mathbb{Q}$-Maschine, dann heißt die Berechnung

$$\rho_0, \rho_1, \rho_2, \ldots$$

analytisch $:\Leftrightarrow$ es gelten (1) und (2) aus Definition 1, zusätzlich gilt

(3) Wenn die Maschine K_t verläßt, springt sie zu einem Zustand aus K_b.

Erläuterung: Die Forderung (3) besagt, daß die Maschine bei jeder Iteration mit höherer Eingabepräzision keine Information aus der vorigen Runde übernehmen kann.

Definition 4: $f : D \to C$ heißt $\mathbb{R}$-analytisch, falls es eine analytische Berechnung auf einer $\mathbb{R}$-Maschine gibt mit:

$$\text{für alle } x \in D \text{ folgt: } in(x) = \rho_0, \rho_1, \rho_2, \ldots \text{ ist analytisch, und}$$

$$\lim_{i \to \infty} (out(\rho_i)) = f(x).$$

$f : D \to C$ heißt δ-$\mathbb{Q}$-analytisch, falls es eine analytische Berechnung auf einer δ-$\mathbb{Q}$-Maschine gibt mit:

$$\text{für alle } x \in D \text{ folgt: } in(x) = \rho_0, \rho_1, \rho_2, \ldots \text{ ist analytisch, und}$$

$$\lim_{i \to \infty} (out(\rho_i)) = f(x).$$

Wir werden sehen, daß die Menge der δ-$\mathbb{Q}$-analytischen Funktionen in hohem Maße von der gewählten Funktion $[]_\delta$ abhängt.

Definition 5: Eine Rundung heißt *normal* genau dann wenn (1) und (2) erfüllt sind.

(1) Für alle $x \in \mathbb{R}$ und alle $\delta \in \mathbb{N}$ existiert ein $\tilde{x} \in \mathbb{R} \setminus \mathbb{Q}$ mit $[x]_\delta = [\tilde{x}]_\delta$.

(2) $[[x]_\delta]_{\delta'} = [x]_{\delta'}$ für alle δ, δ' mit $\delta' \leq \delta$

Wie man leicht verifiziert, ist der Abbruch der Dualentwicklung hinter der δ-ten Nachkommastelle ein Beispiel für eine normale $\mathbb{R}$-berechenbare Rundung.

Definition 6: Ein Programm einer δ-$\mathbb{Q}$-Maschine heißt *robust*, wenn es mit jeder Rundung dieselbe Funktion berechnet.

Eine Funktion f heißt $\mathbb{Q}$-*analytisch*, wenn es ein robustes Programm gibt, das f analytisch berechnet.

f heißt *normal analytisch*, wenn es von einer δ-$\mathbb{Q}$-Maschine mit einer normalen Rundung analytisch berechnet werden kann.

3.2 Der Simulationssatz

Satz 4 (Simulationssatz): Jede $\mathbb{R}$-berechenbare Funktion ist $\mathbb{Q}$-analytisch.

Beweis: Sei also $f : D \to \mathbb{R}^*$ auf der $\mathbb{R}$-Maschine $\mathcal{M}$ berechenbar. In diesem Beweis werden wir f als unendliches Tupel einwertiger Funktionen $(f_1, f_2, f_3, \ldots)$ mit $f_i : D \to \mathbb{R}$ betrachten. Weiter bezeichnet $x = (x_1, x_2, \ldots, x_n) \in D$ den Eingabewert und $[x]_\delta := ([x_1]_\delta, [x_2]_\delta, \ldots, [x_n]_\delta)$.

Nach dem Darstellungssatz gilt

$$f = (f_1, f_2, f_3, \ldots) = \sum_{\sigma \in S_{\mathcal{M}}} a_\sigma \cdot (f^\sigma_{y_1}, f^\sigma_{y_2}, f^\sigma_{y_3}, \ldots),$$

worin $f^\sigma_{y_i}$ die zur Signatur σ gehörigen rationalen Funktionen sind und a_σ die charakteristische Funktion der zu σ gehörenden semi-algebraischen Menge A_σ.

Wir betrachten zunächst den Fall, daß $x \in A_\sigma$ und A_σ eine offene Menge ist. In diesem Fall gilt

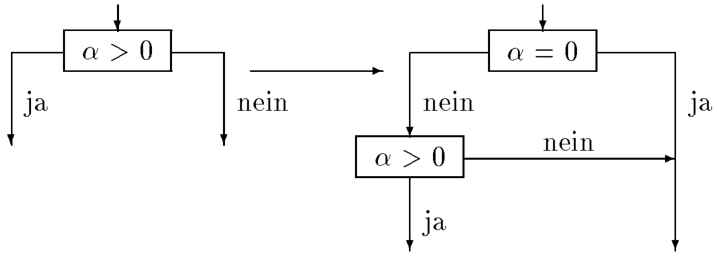
$$\lim_{\delta \to \infty} f^\sigma_{y_i}([x]_\delta) = f^\sigma_{y_i}(x) \text{ für alle } i,$$

so daß $f(x)$ in diesem Fall durch eine δ-$\mathbb{Q}$-Maschine analytisch berechnet werden kann, deren Programm folgendermaßen definiert ist:

$\pi_{\mathcal{M}}$ von $\mathcal{M}$	$\pi_{\mathcal{M}'}$ von $\mathcal{M}'$
$\pi_{\mathcal{M}}(1)$	$\pi_{\mathcal{M}'}(1) :=$ "$\delta := \delta + 1;$"
$\pi_{\mathcal{M}}(2)$	$\pi_{\mathcal{M}'}(2) := \pi_{\mathcal{M}}(1)$
$\vdots$	$\vdots$
$\pi_{\mathcal{M}}(N-1)$	$\pi_{\mathcal{M}'}(N) := \pi_{\mathcal{M}}(N-1)$
$\pi_{\mathcal{M}}(N) =$ "*end;*"	$\pi_{\mathcal{M}'}(N+1) :=$ "*init;*"

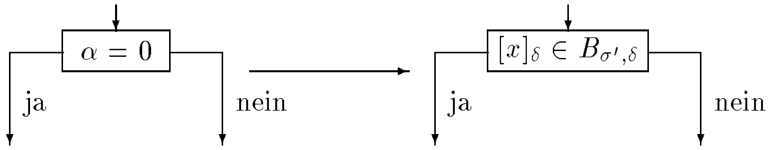
Das *end* von π wird also durch ein "init" ersetzt, so daß nach jedem Programmdurchlauf die Präzision, mit der x gelesen wird, verdoppelt wird. Da A_σ offen ist, gibt es ein $M \in \mathbb{N}$ mit $[x]_\delta \in A_\sigma$ für $\delta > M$, so daß die Berechnungen mit der Eingabe $[x]_\delta$ für $\delta > M$ den *gleichen* Berechnungspfad nehmen wie bei der Eingabe x. Da die $f^\sigma_{y_i}$ als rationale Funktionen stetig sind, konvergiert in diesem Fall die Berechnung der Maschine $\mathcal{M}'$ gegen $f(x)$. $\mathcal{M}'$ wird aber im allgemeinen nicht das richtige Resultat liefern, wenn A_σ nicht offen ist, und wenn $x \in A_\sigma$ auf dem Rand liegt: Sei wie oben $\sigma' \in \Sigma^\sigma$ ein Präfix von σ, das zu einer Abfrage $\alpha > 0$ oder $\alpha = 0$ führt. Dann ist es möglich, daß $f^{\sigma'}_\alpha(x) = 0$ aber $f^{\sigma'}_\alpha([x]_\delta) \neq 0$ für alle δ, so daß die Simulation für jede Genauigkeit an dieser Stelle falsch verzweigt. Diese Situation kann in zwei Fällen auftreten: Wenn die Abfrage $\alpha > 0$ nicht erfüllt ist, oder wenn die Abfrage $\alpha = 0$ erfüllt ist.

Wir werden nun in π alle Abfragen $\alpha > 0$ folgendermaßen ersetzen:



Damit können wir uns auf die Problematik des zweiten Falles beschränken.

Wir werden dieses Problem lösen, indem wir in π' jede Abfrage $\alpha = 0$ durch eine Prozedur $[x]_\delta \in B_{\sigma,\delta}$ ersetzen, deren Berechenbarkeit wir später zeigen. So erhalten wir ein Programm $\tilde{\pi}$. Wir substituieren also:



$B_{\sigma',\delta}$ ist eine offene Umgebung der Menge $\{x \mid f_\alpha^{\sigma'}(x) = 0\}$ und wird so definiert, daß sie sich für $\delta \to \infty$ auf diese zusammenzieht. Außerdem muß für alle δ gelten

$$f_\alpha^{\sigma'}(x) = 0 \Rightarrow [x]_\delta \in B_{\sigma',\delta}.$$

Wir definieren also:

$$B_{\sigma',\delta} := \{\tilde{x} \mid \exists x \in \mathbb{R}^n, (\sum_{i=1}^{n} (x_i - \tilde{x}_i)^2 < n \cdot 2^{-\delta},\ f_\alpha^{\sigma'}(x) = 0)\}$$

Wenn wir die rationale Funktion $f_\alpha^{\sigma'}$ durch ihren Zähler, das Polynom $P_\alpha^{\sigma'}$, ersetzen, folgt:

$$B_{\sigma',\delta} = \{\tilde{x} \mid \exists x \in \mathbb{R}^n, (\sum_{i=1}^{n} (x_i - \tilde{x}_i)^2 < n \cdot 2^{-\delta},\ P_\alpha^{\sigma'}(x) = 0)\}$$

Indem wir $\tilde{\pi}$ durch dieselbe Konstruktion wie oben in ein analytisches Programm verwandeln erhalten wir die Maschine $\tilde{\mathcal{M}}$.

Nun beobachten wir hinsichtlich der Berechnungen von $\mathcal{M}$ und $\tilde{\mathcal{M}}$ folgendes:

Ist die Abfrage $\alpha = 0$ in $\mathcal{M}$ erfüllt, das heißt es liegt $x \in A_\sigma$, dann verläuft die Berechnung auf $\tilde{\mathcal{M}}$ für jedes δ in gleicher Weise wie die Berechnung für x auf $\mathcal{M}$, denn $B_{\sigma',\delta}$ wurde hinreichend groß gewählt, so daß jede Rundung $[x]_\delta \in B_{\sigma',\delta}$ ist, falls $f_\alpha^{\sigma'}(x) = 0$. Liegt $x \notin A_\sigma$, dann gibt es ein $M \in \mathbb{N}$ mit: $[x]_\delta \notin B_{\sigma',\delta}$ für alle $\delta > M$.

Die Berechnung für $[x]_\delta$ verzweigt für $\delta > M$ also genauso wie die für x auf $\mathcal{M}$. Setzt man dies induktiv an jeder Verzweigung fort, so wird klar, daß $\tilde{\mathcal{M}}$ analytisch die Funktion f berechnet.

Wir haben dabei noch zwei Schwierigkeiten unterschlagen: Es kann nämlich sein, daß $f^\sigma([x]_\delta)$ unendlich wird für $[x]_\delta \in B_{\sigma',\delta} \setminus A_\sigma$: bei der Anweisung $\alpha := \alpha/x_1$ könnte zum Beispiel gelten, daß $[x_1]_\delta = 0 \neq x_1$. In diesem Fall springen wir auf die Anweisung "init" und beginnen in der Berechnung eine neue Runde. Da f^σ auf A_σ keinen Pol hat, gilt das auch für $B_{\sigma',\delta}$ für hinreichend großes δ.

Das zweite Problem ist, daß $\tilde{\mathcal{M}}$ beim Rechnen mit Näherungswerten unter Umständen nicht anhält, da nicht gewährleistet ist, daß alle Berechnungspfade terminieren. Wir lösen dieses Problem indem wir die Anzahl der Verzweigungen der Maschine beispielsweise auf δ beschränken, indem wir nur δ Verzweigungen ausführen und dann zur Anweisung "init" zurückspringen. Irgendwann wird δ so groß, daß $\mathcal{M}$ an jeder Verzweigung richtig springt und nicht vorzeitig abbricht.

Nun bleibt uns noch zu zeigen, daß die Abfrage $[x]_\delta \in B_{\sigma',\delta}$ durch eine Prozedur aufgelöst werden kann. Offensichtlich gilt

$$B_{\sigma',\delta} = \{\tilde{x} \mid \exists x \in \mathbb{R}^n, y \in \mathbb{R}, (y^2 + \sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 - n \cdot 2^{-\delta} = 0,\, P_\alpha^{\sigma'}(x) = 0)\}.$$

Man erhält also $B_{\sigma',\delta}$ als die Projektion $\tilde{x}$ einer algebraischen Mannigfaltigkeit über $x, \tilde{x}$ und y. Die Abfrage $\tilde{x} \in B_{\sigma',\delta}$ ist äquivalent zu

$$B_{\sigma,\delta}(\tilde{x}) = \{(x,y) \mid y^2 + \sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 - n \cdot 2^{-\delta} = 0,\, P_\alpha^{\sigma'}(x) = 0\} \neq \emptyset$$

$P_\alpha^{\sigma'}(x)$ ist berechenbar, also ist $B_{\sigma',\delta}(\tilde{x}) = \emptyset$ entscheidbar [T], [Col]. Wir können also die Abfrage $\tilde{x} \in B_{\sigma',\delta}$ durch eine Prozedur ersetzen, die die Abfrage entscheidet, womit der Beweis des Satzes abgeschlossen ist. Da wir außer $|x - [x]_\delta| < 2^{-\delta}$ keine Eigenschaft der Rundung verwendet haben, ist diese Simulation robust.

3.3 Nähere Einordnung der Funktionenklassen

Satz 5: Die Menge der $\mathbb{R}$-berechenbaren Funktionen ist echt in der Menge der δ-$\mathbb{Q}$-analytischen Funktionen enthalten.

Beweis: Wie wir bereits erwähnt haben, sind die Koch'sche Kurve und die quadratfüllende Hilbertkurve nicht $\mathbb{R}$-berechenbar. Man sieht jedoch, daß sie $\mathbb{Q}$-analytisch sind, wenn man sich klar macht, daß diese Kurven durch eine unendliche Folge $\mathbb{Q}$-berechenbarer Kurven definiert sind, die gegen die Koch'sche Kurve beziehungsweise die Hilbertkurve konvergieren.

Wir betrachten zwei Funktionen, die uns später noch von Nutzen sein werden. Sei $d_i : [0,1] \to \mathbb{R}$ für $i = 1,2$ definiert durch

$$d_1(x) = \begin{cases} \frac{1}{q} & \text{für } x = \frac{p}{q},\ (p,q) = 1,\ p,q \in \mathbb{N} \\ \\ 0 & \text{für } x \in [0,1] - \mathbb{Q}. \end{cases}$$

$$d_2(x) = \begin{cases} 1 & \text{für } x \in \mathbb{Q} \cap [0,1] \\ \\ 0 & \text{für } x \in [0,1] - \mathbb{Q}. \end{cases}$$

Lemma 3: d_1 ist $\mathbb{Q}$-analytisch.

Beweis: Die Maschine liest x mit der Präzision δ, das heißt die rationale Zahl $[x]_\delta$. Dann sucht sie die kleinste rationale Zahl $\frac{p}{q} \in \left[[x]_\delta - 2^{-\delta}, [x]_\delta + 2^{-\delta}\right]$ mit minimalem q,[1] und gibt $\frac{1}{q}$ aus. Falls x irrational ist, so konvergiert $\frac{1}{q} \to 0$; andernfalls gibt es ein δ', so daß für alle $\delta > \delta'$ die Eingabe $x = \frac{p}{q}$ im Eingabeintervall den minimalen Nenner hat und die Maschine konstant $\frac{1}{q}$ ausgibt.

In den Beweisen der folgenden Lemmata bezeichnen wir mit $y_i(x,\delta)$ den Wert, den die δ-$\mathbb{Q}$-Maschine mit der Eingabe x, die mit der Präzision δ gelesen wurde, in die Speicherzelle y_i schreibt.

Lemma 4: d_2 ist $\mathbb{R}$-analytisch.

Beweis: Die Maschine zählt alle rationalen Zahlen auf und vergleicht die auf dem Eingabeband stehende Zahl x mit der aufgezählten Zahl q_i. Ist $x = q_i$, dann schreibt sie 1 nach y_1 und *hält* [2]. Offensichtlich gilt

$$\lim_{i \to \infty} (y_1(x,i)) = d_2(x).$$

Wir untersuchen nun, in welchen Fällen d_2 auch $\mathbb{Q}$-analytisch ist und in welchen das nicht der Fall ist.

Lemma 5: Es gibt Rundungen $[]_\delta$, so daß d_2 δ-$\mathbb{Q}$-analytisch ist.

Beweis: Mit einer Rundung, die das Kriterium

$$[x]_\delta = \frac{p}{q} \text{ mit } q \leq \delta \quad \Rightarrow \quad \frac{p}{q} = x$$

erfüllt, ist d_2 $\mathbb{Q}$-analytisch. Die Maschine muß lediglich $[x]_\delta$ in der Form $\frac{p}{q}$ darstellen und prüfen ob $q \leq \delta$. Falls ja, gibt sie 1 aus, falls nein gibt sie 0 aus. Somit gibt es für jedes $x \in \mathbb{Q}$ ein $M \in \mathbb{N}$, so daß $y([x]_\delta, \delta) = 1$ für alle $\delta > M$.

[1] Man mache sich klar, daß dieses leicht zu realisieren ist, indem man die natürlichen Zahlen $q \in \mathbb{N}$ in natürlicher Weise aufzählt, und überprüft, ob ein Vielfaches von $\frac{1}{q}$ im Eingabeintervall liegt.

[2] Da es sich um eine analytische Maschine handelt, hält sie nicht wirklich, sondern läuft weiter, ohne die Ausgabe zu verändern.

Man macht sich wie folgt klar, daß es eine $\mathbb{R}$-berechenbare Funktion gibt, die eine solche Rundung realisiert: Zuerst wird geprüft, ob $x = \frac{p}{q}$ mit $q \leq \delta$, wenn ja wird x präzise ausgegeben, andernfalls wird eine beliebige Zahl $\frac{s}{t}$ mit $t > \delta$ und $|x - \frac{s}{t}| < 2^{-\delta}$ gesucht und ausgegeben. Eine solche Rundung ist jedoch nicht normal, da sie die Bedingung (1) für normale Rundungen verletzt.

Lemma 6: Für alle normalen Rundungsfunktionen $[]_\delta$ ist d_2 nicht δ-$\mathbb{Q}$-analytisch berechenbar.

Beweis: Wir werden nun die Annahme, daß d_2 auch mit einer normalen Rundung $\mathbb{Q}$-analytisch ist, widerlegen.

Sei also $\frac{1}{2} > \varepsilon > 0$ beliebig. Unter dieser Annahme gilt: Ist x eine rationale Zahl, dann gibt es ein δ_x mit $y_1(x, \delta) > 1 - \varepsilon$ für $\delta > \delta_x$. Ist x nicht rational, dann gibt es ein δ_x mit $y_1(x, \delta) < \varepsilon$ für $\delta > \delta_x$.

Wir führen die Annahme zum Widerspruch, indem wir mittels einer konvergenten Folge eine Zahl konstruieren, für die die Berechnung von $y_1(x, \delta)$ zwischen Werten $> 1 - \varepsilon$ und $< \varepsilon$ oszilliert.

Man begint mit einer beliebigen Zahl $v_1 \in \mathbb{Q}$. Dann gibt es ein δ_1, so daß $y_1(v_1, \delta_1) > 1 - \varepsilon$.

Nun gibt es, gemäß der Bedingung (1) für normale Rundungen, ein irrationales v_2 für das gilt $[v_2]_{\delta_1} = [v_1]_{\delta_1}$ (Hierbei ist $\delta = \delta' = \delta_1, v_2 = \tilde{x}$ und $v_1 = x$). Mit der Bedingung (2) folgt dann $[v_2]_\delta = [[v_2]_{\delta_1}]_\delta = [[v_1]_{\delta_1}]_\delta = [v_1]_\delta$ für alle $\delta \leq \delta_1$. Wegen $v_2 \in \mathbb{R} \setminus \mathbb{Q}$ gibt es ein $\delta_2 > \delta_1$, so daß $y_1(v_2, \delta_2) < \varepsilon$. Dann setzen wir $v_3 := [v_2]_{\delta_2} \in \mathbb{Q}$, und finden ein $\delta_3 > \delta_2$ mit $y_1(v_3, \delta_3) > 1 - \varepsilon$.

Wir finden also für alle geraden i ein irrationales v_i mit $[v_i]_\delta = [v_{i-1}]_\delta$ für alle $\delta \leq \delta_{i-1}$ und ein $\delta_i > \delta_{i-1}$, so daß $y_1(v_i, \delta_i) < \varepsilon$.

Weiter gibt es für alle ungeraden i ein rationales v_i mit $[v_i]_\delta = [v_{i-1}]_\delta$ für alle $\delta \leq \delta_{i-1}$ und ein $\delta_i > \delta_{i-1}$, so daß $y_1(v_i, \delta_i) > 1 - \varepsilon$.

Da $2^{-\delta_i} \to 0$ für $i \to \infty$ und da v_{i+1} aus dem Intervall $[v_i - 2^{-\delta_i}, v_i + 2^{-\delta_i}]$ gewählt wird, existiert der Grenzwert $\lim_{i\to\infty} v_i$. Sei also $v := \lim_{i\to\infty} v_i$, so haben wir $[v]_\delta = [v_j]_\delta$ für alle j und alle $\delta \leq \delta_j$, damit folgt $y_1(v, \delta_i) > 1 - \varepsilon$ falls i ungerade und $y_1(v, \delta_i) < \varepsilon$ falls i gerade. Also divergiert die Ausgabe der Maschine.

Mit diesen Lemmata folgt

Satz 6: Die Menge der normalen $\mathbb{Q}$-analytischen Funktionen ist echt in der Menge der $\mathbb{R}$-analytischen Funktionen enthalten.

Weiter folgt nun leicht

Satz 7: Die Menge der δ-$\mathbb{Q}$-analytischen Funktionen ist nicht notwendigerweise abgeschlossen unter der Hintereinanderausführung.

Beweis: Man kann d_2 berechnen, indem man zunächst d_1 berechnet und anschließend wie folgt

$$d_2(x) = \begin{cases} 1 & \text{falls } d_1(x) \neq 0 \\ 0 & \text{falls } d_1(x) = 0 \end{cases}$$

Das ist ein sehr unbefriedigendes Resultat, denn die Hintereinanderausführung berechenbarer Funktionen sollte berechenbar sein. Aus diesem Grund verfolgen wir die Theorie in eine zusätzliche Richtung weiter. Wir verschärfen die Anforderungen an *analytische* Berechnungen zu *stark analytisch*.
Vorher werden wir jedoch einen μ-Operator für reellwertige Funktionen definieren und untersuchen unter welchen Umständen er $\mathbb{Q}$-analytisch ist.

3.4 Ein μ-Operator für reellwertige Funktionen

Wir betrachten eine stetige Funktion $f : \mathbb{R} \to \mathbb{R}$, die zwischen zwei Werten x_1 und x_2 das Vorzeichen wechselt, das heißt $f(x_1) < 0$ und $f(x_2) > 0$. Mit dem Zwischenwertsatz folgt, daß zwischen x_1 und x_2 ein x_3 existiert, mit $f(x_3) = 0$.
Wenn f $\mathbb{R}$-berechenbar ist, so ist x_3 $\mathbb{Q}$-analytisch. Da $f|\mathbb{Q}$ $\mathbb{Q}$-berechenbar ist, kann x_3 durch Intervallschachtellung angenähert werden.
Wir wollen nun auch Nullstellen finden, die sich nicht in einem vorgegebenen Intervall befinden und die eventuell die x-Achse nicht schneiden sondern nur berühren.
Aus der klassischen Berechenbarkeitstheorie kennen wir den Begriff des μ-Operators [P]: Sei $f : \mathbb{N}^{r+1} \to \mathbb{N}$ eine berechenbare Funktion, dann ist $\mu f : \mathbb{N}^r \to \mathbb{N}$ mit

$$\mu f(x) := \begin{cases} \min\{n | f(n,x) = 0 & \text{und für alle } m < n \text{ ist } f(m,x) \text{ definiert}\} \\ & \text{falls ein solches } n \text{ existiert} \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

ebenfalls berechenbar. Hier bedeutet *nicht definiert*, daß die Turingmaschine mit dieser Eingabe nicht hält. Die Maschine wird die $n \in \mathbb{N}$ aufzählen und die $f(n,x)$ berechnen. Sobald sie ein n findet, so daß $f(n,x) = 0$, gibt sie n aus und hält. Falls die Maschine auf ein n stößt, so daß $f(n,x)$ nicht definiert ist, oder falls es kein n mit $f(n,x) = 0$ gibt, hält sie nicht.
Wir werden nun einen μ-Operator für $\mathbb{R}$-berechenbare Funktionen, die die nichtnegativen reellen Zahlen auf die reellen Zahlen abbilden, einführen und zeigen, daß er für Lipschitz-stetige Funktionen $\mathbb{Q}$-analytisch ist. Eine Funktion $f : D \to \mathbb{C}$ heißt Lipschitz-stetig [3] auf D, wenn es eine Konstante L_f gibt, so daß für alle $x, y \in D$ gilt:

$$|f(x) - f(y)| \leq L_f |x - y|.$$

Definition 7: Sei $f : \mathbb{R}_0^+ \to \mathbb{R}$ eine stetige $\mathbb{R}$-berechenbare Funktion; dann ist μf definiert durch:

$$\mu f := \begin{cases} \min\{x \geq 0 | f(x) = 0\} & \text{falls so ein } x \text{ existiert} \\ -1 & \text{falls es ein } \varepsilon > 0 \text{ gibt, so daß kein} \\ & x \text{ existiert mit } |f(x)| < \varepsilon, \forall x \in \mathbb{R}_0^+ \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

Satz 8: Für alle Lipschitz-stetigen $\mathbb{R}$-berechenbaren Funktionen $f : \mathbb{R}_0^+ \to \mathbb{R}$ ist μf $\mathbb{Q}$-analytisch.

[3] Die Definitionen von Lipschitz-stetigen beziehungsweise gleichmäßig stetigen Funkltionen stammen aus [Kön].

Hier bedeutet *nicht definiert*, daß die Ausgabe der $\mathbb{Q}$-Maschine divergiert. Wir gehen so ähnlich vor wie beim Beweis des Simulationssatzes: solange wir nicht ausschließen können, daß in einem Intervall eine Nullstelle liegt, nehmen wir dies an.

Beweis: Es gilt

$$|f(x) - f(y)| \leq L_f|x - y|.$$

Wenn nun $|x - y| \leq \frac{1}{L_f}\varepsilon$ und $|f(x)| > \varepsilon$ dann folgt $f(y) \neq 0$, denn die Annahme $f(y) = 0$ führt zu dem Widerspruch:

$$|f(x) - 0| = |f(x)| \leq L_f|x - y| \leq L_f\frac{1}{L_f}\varepsilon = \varepsilon$$

Wenn wir also ein x kennen mit $|f(x)| > \varepsilon$, dann gibt es in der Umgebung $[x - \frac{\varepsilon}{L_f}, x + \frac{\varepsilon}{L_f}]$ keine Nullstellen. Um mögliche Nullstellen auf einem Intervall $[a, b]$ zu finden, genügt es also, das Intervall in ein Raster

$$R^{[a,b]} := \{a, a + \frac{\varepsilon}{L_f}, a + \frac{2\varepsilon}{L_f}, \ldots, a + \frac{n\varepsilon}{L_f}\}$$

mit $b - (a + \frac{n\varepsilon}{L_f}) \leq \frac{\varepsilon}{L_f}$ einzuteilen und die Funktion an den Rasterpunkten auszuwerten. Wird kein $x \in R^{[a,b]}$ gefunden mit $|f(x)| \leq \varepsilon$, folgt daß f auf $[a, b]$ keine Nullstelle hat. Gibt es ein $x \in R^{[a,b]}$ mit $|f(x)| \leq \varepsilon$, so könnte in der $\frac{\varepsilon}{L_f}$-Umgebung von x eine Nullstelle liegen.

Die Idee dieses Beweises ist nun das ε gegen Null gehen zu lassen. Dann geht auch der Abstand $\frac{\varepsilon}{L_f}$ gegen Null; und falls eine Nullstelle existiert, konvergiert die Folge der Rasterpunkte, für die $|f(x)| \leq \varepsilon$ gegen diese Nullstelle.

Weiterhin müssen wir die obere Grenze des Intervalls, in dem wir nach Nullstellen suchen, gegen unendlich gehen lassen. Und da wir L_f nicht notwendigerweise kennen, verwenden wir eine Variable i, die wir hochzählen. Unser Algorithmus wird richtige Ergebnisse liefern sobald $i \geq L_f$.

Wir werden eine Folge von berechenbaren Funktionen $(\mu^f)_i$ angeben, die gegen μf konvergiert.

$$\mu_i^f := \begin{cases} \min(x \in R^{[0,i]} | f(x) \leq \varepsilon_i) & \text{falls so ein } x \text{ existiert} \\ -1 & \text{sonst} \end{cases}$$

Hierbei ist $\varepsilon_1 := \frac{1}{2}$,

$$\varepsilon_i := \begin{cases} \varepsilon_{i-1} & \text{falls } \mu_{i-1}^f = -1 \\ \frac{\varepsilon_{i-1}}{2} & \text{sonst.} \end{cases}$$

und

$$R^{[0,i]} := \{0, \frac{\varepsilon_i}{i}, \frac{2\varepsilon_i}{i}, \frac{3\varepsilon_i}{i}, \ldots, i\}$$

Wenn also das Raster $R^{[0,i]}$ kein x enthält, mit $f(x) < \varepsilon_i$ ist $\mu_i^f = -1$. Falls $i \geq L_f$ folgt damit, daß es kein $\tilde{x} \leq i$ gibt mit $f(\tilde{x}) = 0$.

Falls es jedoch ein $x \in R^{[0,i]}$ gibt mit $f(x) < \varepsilon_i$, wird angenommen, daß es sich um eine Nullstelle handelt. Wir betrachten nun die drei Fälle:

1. Es gibt ein $\varepsilon > 0$, so daß es kein $x \in \mathbb{R}_0^+$ gibt mit $|f(x)| < \varepsilon$. In diesem Fall gibt es ein j, so daß $\varepsilon_j \leq \varepsilon$ Dann wird $\mu_i^f = -1$ für alle $i \geq j$.
2. Es existiert eine minimale Nullstelle x_0. Nun gibt es ein j mit $j \geq L_f$ und $j \geq x_0$. Dann ist für alle $i \geq j$ das Raster $R^{[0,i]}$ fein genug um die Nullstelle x_0 zu erkennen. Nun ist es möglich, daß Funktionswerte zwischen 0 und x_0 kleiner als ε_i sind für einige i, und deshalb fälschlich für Nullstellen gehalten werden. Da jedoch $\varepsilon_i \to 0$ für $i \to \infty$ folgt $\mu_i^f \to x_0$ für $i \to \infty$.
3. Es gibt keine Nullstelle, aber für jedes $\varepsilon > 0$ gibt es ein x mit $f(x) < \varepsilon$. Die Funktion hat also kein betragsmäßiges Minimum. Da aber eine stetige Funktion auf einem abgeschlossenem Intervall ein betragsmäßiges Minimum hat, geht in diesem Fall die Folge der x_j, für die $f(x_j) < \varepsilon_j$ gilt, gegen unendlich. Damit ist der Grenzwert der μ_i^f nicht definiert: Es können i existieren, für die es keine x gibt mit $|f(x)| \leq \varepsilon_i$, μ_i^f also -1 wird. Es wird aber auch unendlich viele i geben, so daß es $x_i \in R^{[0,i]}$ gibt mit $|f(x_i)| \leq \varepsilon_i$, daß also $\mu_i^f = x_i$ wird.

Der Algorithmus berechnet also in allen Fällen die richtige Funktion. Damit wäre dieser Beweis abgeschlossen.

Wir haben die Lipschitz-stetigkeit gebraucht, um eine Abschätzung zu haben, wie groß der Abstand von Funktionswerten von nahe beieinanderliegenden Urbildern werden kann.
Eine Funktion $f : D \to \mathbb{C}$ heißt gleichmäßig stetig auf D, wenn es zu jedem $\varepsilon > 0$ ein $\delta > 0$ gibt, so daß für alle $x, y \in D$ mit $|x - y| < \delta$ gilt:

$$|f(x) - f(y)| < \varepsilon$$

Wenn wir zu einer gleichmäßig stetigen Funktion f eine berechenbare Abschätzung $g^f : \mathbb{R}^+ \to \mathbb{R}^+$ mit $g^f(\varepsilon) \leq \delta$ für alle ε, δ haben, können wir auch μf analytisch berechnen, indem wir $g^f(\varepsilon)$ als Rastereinteilung verwenden.
Da jedoch die gleichmäßige Stetigkeit lediglich die Existenz eines δ verlangt, kann man im Allgemeinen so ein g^f nicht angeben.

Kapitel 4

Stark-$\mathbb{Q}$-analytische Maschinen

Wie angekündigt verschärfen wir nun die Anforderungen an unsere analytischen Funktionen, und erhalten eine bezüglich der Hintereinanderausführung abgeschlossene Klasse von Funktionen.

4.1 Definitionen und Abgeschlossenheit

Definition 8: Eine Berechnung

$$\rho_0, \rho_1, \rho_2, \dots$$

heißt *stark analytisch* :⇔

(1) Die Berechnung ist analytisch

(2) $|\lim_{j\to\infty} y_l(\rho_j) - y_l(\rho_i)| \leq y_1(\rho_i)$, für alle y_l und alle $\rho_i \in K_t$

(3) $y_1(\rho_j) \to 0$ für $j \to \infty$.

Im Falle, daß eine Berechnung stark analytisch ist, kennen wir also stets die Präzision der erzielten Approximation.

Lemma 7: Die stark-$\mathbb{Q}$-analytischen Funktionen sind stetig.

Beweis: Wir nehmen an, es gäbe eine stark-$\mathbb{Q}$-analytische Funktion f mit einer Unstetigkeitsstelle $\tilde{x}$. O.B.d.A. sei $f(x) > f(\tilde{x}) + \varepsilon$ für ein $\varepsilon > 0$ und alle $x > \tilde{x}$. Dann gibt es ein M, so daß für alle $\delta \geq M$ gilt:

$$y_1(\tilde{x}, \delta) < \frac{\varepsilon}{3}$$

Nun gibt es jedoch Rundungen $[]_\delta$ und $x > \tilde{x}$, so daß $[\tilde{x}]_\delta = [x]_\delta$ für alle $\delta \leq M$. Damit folgt $y_2(\tilde{x}, M) = y_2(x, M)$ und $y_1(\tilde{x}, M) = y_1(x, M) < \frac{\varepsilon}{3}$ obwohl $|f(x) - f(\tilde{x})| > \varepsilon$. Damit haben wir einen Widerspruch zur Definition 8.

Im folgenden Abschnitt zeigen wir, daß die Menge der stark $\mathbb{Q}$-analytischen Funktionen abgeschlossen ist gegenüber der primitiven Rekursion.

Lemma 8: Seien $g : \mathbb{R}^r \to \mathbb{R}$ und $h : \mathbb{R}^{r+2} \to \mathbb{R}$ zwei stark $\mathbb{Q}$-analytische Funktionen, und sei $f : \mathbb{N} \times \mathbb{R}^r \to \mathbb{R}$ definiert durch

$$f(0, x) := g(x)$$

und

$$f(n+1, x) := h(n, f(n, x), x)$$

für $n \in \mathbb{N}$ und $x \in \mathbb{R}^r$. Dann ist f stark $\mathbb{Q}$-analytisch.

Beweis: Es ist naheliegend, $f(n, x)$ zu berechnen, indem man $n+1$ δ-$\mathbb{Q}$-Maschinen hintereinander stellt, und die Ausgabe der einen als Eingabe für die anderen verwendet.

Da nun aber die einzelnen Maschinen unendlich lange rechnen, müssen alle Maschinen gleichzeitig arbeiten und als Eingabe die Approximationswerte der Vorgängermaschinen verwenden.

Es sind also folgende Punkte zu zeigen:

1. Die Ausgabe einer δ-$\mathbb{Q}$-Maschine erfüllt – soweit sie eine stark analytische Funktion berechnet – alle Voraussetzungen, um einer weiteren δ-$\mathbb{Q}$-Maschine als Eingabe zu dienen.
2. Die Zugriffe auf das Ein- beziehungsweise Ausgabeband lassen sich im Hauptspeicher der Maschine simulieren. Dieser Punkt ist offensichtlich.
3. Beliebig viele, hintereinandergeschaltete δ-$\mathbb{Q}$-Maschinen lassen sich von einer einzigen simulieren.

ad 1: Sei $\mathcal{M}$ eine Maschine, die eine stark analytische Funktion berechnet und deren Ausgabe gegen die reelle Zahl $\bar{y}$ konvergiert. Das heißt: $y_1(\rho_j) \to 0$ und $y_2(\rho_j) \to \bar{y}$ für $j \to \infty$, wobei $|\bar{y} - y_2(\rho_j)| \leq y_1(\rho_j)$ für alle $\rho_j \in K_t$.

Weiter sei $\mathcal{M}'$ eine δ-$\mathbb{Q}$-Maschine, die als Eingabe $\bar{y}$ erhalten soll. Da die speziellen Eigenschaften der Rundung keine Rolle spielen, genügt es, wenn $|[\bar{y}]_\delta - \bar{y}| < 2^{-\delta}$. Die Maschine $\mathcal{M}$ rechnet irgendwann so genau, daß $y_1 < 2^{-\delta}$; dann ist auch $|\bar{y} - y_2| < 2^{-\delta}$. Man kann also y_2 als $[\bar{y}]_\delta$ interpretieren und unmodifiziert als Eingabe übernehmen, sobald $y_1 < 2^{-\delta}$.

ad 3: Wir zeigen, daß zwei Maschinen $\mathcal{M}_1$ und $\mathcal{M}_2$ von einer Maschine $\mathcal{M}$ simuliert werden können, damit folgt induktiv, daß auch n Maschinen von einer simuliert werden können.

Den Rechenspeicher von $\mathcal{M}$ teilen wir in 4 Folgen von Zellen ein:

$$\begin{array}{lllll} Z^0: & z_1, & z_2, & z_3, & z_4, \quad z_5 \\ Z^1: & z_6, & z_9, & z_{12}, & \dots \\ Z^2: & z_7, & z_{10}, & z_{13}, & \dots \\ Z^3: & z_8, & z_{11}, & z_{14}, & \dots \end{array}$$

Nun identifizieren wir Z^1 mit dem Rechenspeicher von $\mathcal{M}_1$, Z^2 mit dem Ausgabeband von $\mathcal{M}_1$ beziehungsweise dem Eingabeband von $\mathcal{M}_2$ und Z^3 mit dem

Rechenspeicher von $\mathcal{M}_2$. Das Eingabeband von $\mathcal{M}_1$ ist das von $\mathcal{M}$, das Ausgabeband von $\mathcal{M}_2$ ist das von $\mathcal{M}$.

Z^0 verwenden wir um die Register zwischenzuspeichern. Es seien $z_{\alpha,1} := z_1$, $z_{\gamma,1} := z_2$ die α- und γ-Register von $\mathcal{M}_1$ und $z_{\alpha,2} := z_3$, $z_{\gamma,2} := z_4$, $z_{\delta,2} := z_5$ die α-, γ- und δ-Register von $\mathcal{M}_2$; das δ-Register von $\mathcal{M}$ wird als das von $\mathcal{M}_1$ betrachtet.

Nun gehen wir wie folgt vor:

- α- und γ-Register von $\mathcal{M}_1$ werden aus $z_{\alpha,1}$ und $z_{\gamma,1}$ geladen.
- $\mathcal{M}_1$ wird bis zur Erhöhung von δ simuliert. Es ist leicht zu sehen, daß mit kleinen Veränderungen im Programm auf Z^1 beziehungsweise Z^2 anstelle des vollen Rechenspeichers beziehungsweise des Ausgabebandes zugegriffen werden kann. In Z^2 wird ein Zwischenergebnis abgelegt.
- α- und γ-Register werden wieder gesichert.
- Es wird getestet, ob das Zwischenergebnis genau genug ist, um $\mathcal{M}_2$ als Eingabe zu dienen,
- falls nein, wird zu $\mathcal{M}_1$ zurückgesprungen.
- falls ja, werden die α- und γ-Register von $\mathcal{M}_2$ aus $z_{\alpha,2}$ und $z_{\gamma,2}$ geladen.
- $\mathcal{M}_2$ wird bis zur Erhöhung von δ simuliert, das heißt $z_{\delta,2}$ wird erhöht. $\mathcal{M}_2$ schreibt auf das Ausgabeband von $\mathcal{M}$.
- α- und γ-Register werden wieder gesichert, δ wird erhöht und es wird mit der Simulation von $\mathcal{M}_1$ fortgefahren.

Da die Ausgabe von $\mathcal{M}_1$ konvergiert und die Ausgabe von $\mathcal{M}_2$ mit konvergenten Eingabedaten ebenfalls konvergiert (siehe Punkt 1), folgt induktiv die Abgeschlossenheit gegenüber der primitiven Rekursion, die zu zeigen war.

Wir wollen nun den Zusammenhang zwischen den stark analytischen Funktionen und den im Sinne von Grzegorczyk berechenbaren reellen Funktionen klären. Wir gehen dabei nicht auf die originale Definition zurück, sondern wählen eine dazu äquivalente Definition, die man in ([Ko], Seite 51) findet. Diese Definition beruht auf einer Turingmaschine mit Orakel. Das Orakel leistet für die Turingmaschine dasselbe wie unsere Eingabe mit Präzision δ. Der wesentliche Unterschied in der Definition zu *stark analytisch* besteht darin, daß die Turingmaschine aus einer vorgegebenen Präzision der zu berechnenden Approximation $\tilde{y}$ von $f(x)$, nämlich der Forderung

$$|\tilde{y} - f(x)| \leq 2^{-n}$$

eine Anfrage an das Orakel berechnet, die in der Vorgabe der Präzision 2^{-m} für die Ausgabe $\tilde{x}$ des Orakels besteht: $|x - \tilde{x}| \leq 2^{-m}$. Mittels $\tilde{x}$ als Eingabe berechnet die Turingmaschine die Ausgabe $\tilde{y}$. f heißt *reell berechenbar*, wenn es eine Maschine gibt, die für jede Approximation $\tilde{x}$ mit $|x - \tilde{x}| \leq 2^{-m}$ ein Resultat $\tilde{y}$ mit $|\tilde{y} - f(x)| \leq 2^{-n}$ erzielen kann.

Man sieht ohne weiteres, daß unsere Maschine die Turingmaschine simulieren kann und daß die reell berechenbaren Funktionen somit stark analytisch sind.
Ebenso erkennt man, daß die Menge der stark analytischen Funktionen reell berechenbar ist. Die Robustheit war dabei wesentlich.
Wenn wir jedoch die Rundung

$$[x]_\delta = \frac{p}{q} \text{ mit } q \leq \delta \quad \Rightarrow \quad \frac{p}{q} = x$$

verwenden, gibt es für jede rationale Zahl eine Präzision δ, so daß für alle $\delta' \geq \delta$ klar ist, ob $x = [x]_{\delta'}$ gilt oder nicht gilt, wie wir bereits gesehen haben. Auf diese Weise können wir auch Funktionen mit rationalen Unstetigkeitsstellen stark δ-$\mathbb{Q}$-analytisch berechnen. Da die reell berechenbaren Funktionen stetig sind, stimmen in diesem Fall die beiden Definitionen nicht überein. Unbefriedigend ist, daß damit die Definition von *stark analytisch* von dem Rundungsverfahren für reelle Zahlen abhängt. Andererseits ist es unbefriedigend, daß die Stufenfunktion $\Theta(x)$ mit $\Theta(x) := 0$ für $x \leq 0$ und $\Theta(x) := 1$ für $x > 1$ nicht reell berechenbar ist.

4.2 Prozedurtechniken

Wir erweitern nun unser Maschinenmodell, indem wir zulassen, daß (robuste) stark-$\mathbb{Q}$-analytische Funktionen $\{f_1, \ldots, f_m\}$ als Prozeduren verwendet werden.
Sei nun $\mathcal{M}_{f_i}$ eine $\mathbb{Q}$-Maschine, deren Ausgabe gegen eine stark-$\mathbb{Q}$-analytische Funktion f_i konvergiert.
Wir definieren nun

$$\alpha := f_i(p_1, p_2, \ldots, p_n)$$

als Befehl für eine $\mathbb{R}$-Maschine, der α den Grenzwert zuweist, gegen den die Ausgabe von $\mathcal{M}_{f_i}$ mit der Eingabe $p_1, p_2, \ldots, p_n$ konvergiert, wobei $p_j \in \{\alpha, x_1, x_2, \ldots, z_1, z_2, \ldots\}$ für $j \in \{1, 2, \ldots, n\}$.
Wir werden $\mathbb{R}$-Maschinen, auf denen Programme laufen, die solche Prozeduren $\{f_1, \ldots, f_m\}$ verwenden, $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ nennen.
Wenn wir stark $\mathbb{Q}$-analytische Funktionen in $\mathbb{Q}$-Maschinen einbinden wollen, dürfen wir lediglich Rundungen übergeben, da Speicherstellen von $\mathbb{Q}$-Maschinen nur rationale Werte übergeben werden können. Wir definieren also den Befehl

$$\alpha := [f(p_1, p_2, \ldots, p_n)]_\delta.$$

für $\mathbb{Q}$-Maschinen. Hierbei gilt $[f(p_1, p_2, \ldots, p_n)]_\delta \in \mathbb{Q}$ und $|[f(p_1, p_2, \ldots, p_n)]_\delta - f(p_1, p_2, \ldots, p_n)| < 2^{-\delta}$. $\mathbb{Q}$-Maschinen, in die Prozeduren $\{f_1, \ldots, f_m\}$ eingebunden werden, bezeichnen wir mit $\mathcal{M}_{\mathbb{Q}}(f_1, \ldots, f_m)$.
Wir untersuchen nun, inwieweit eine Maschine, die sich dieser Prozedurtechniken bedient, durch eine herkömmliche $\mathbb{Q}$-analytische Maschinen simuliert werden kann.
Da sich für jedes stark $\mathbb{Q}$-analytische f die Nährung $[f(p_1, p_2, \ldots, p_n)]_\delta$ in endlicher Zeit berechnen läßt, stoßen wir im Fall von $\mathcal{M}_{\mathbb{Q}}(f_1, \ldots, f_m)$ auf keinerlei Probleme.
Wir untersuchen also die Maschinen $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$, denen das Resultat exakt zur Verfügung gestellt wird, und erhalten den folgenden Satz, der eine Verschärfung des Simulationssatzes darstellt.

Satz 9: Alle Funktionen, die von $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ berechnet werden können, sind $\mathbb{Q}$-analytisch.

Beweis: Die Idee des Beweises ist, die Grundrechenoperationen über $\mathbb{R}$ durch stark-$\mathbb{Q}$-analytische Funktionen zu simulieren. Deren endlich häufige Hintereinanderausführung ist, Lemma 8 zufolge, ebenfalls stark-$\mathbb{Q}$-analytisch.

Wir werden nun zeigen, wie eine δ-$\mathbb{Q}$-Maschine konstruiert wird, die ab einer gewissen Anfangsgenauigkeit δ' (in Abhängigkeit von der Eingabe) an jeder Verzweigung richtig verzweigt.

Wir nutzen aus, daß wir zu jedem berechneten beziehungsweise approximierten Wert x die Präzision kennen und diese als ein Intervall

$$[x]_\delta - 2^{-\delta} =: x_{min}(\delta) \leq x \leq x_{max}(\delta) := [x]_\delta + 2^{-\delta}$$

darstellen können mit $x_{min}(\delta) \to x$ und $x_{max}(\delta) \to x$ für $\delta \to \infty$.

Dann gilt im Falle $x = 0$ immer $x_{min}(\delta) \leq 0 \leq x_{max}(\delta)$ für alle δ,

und im Falle $x \neq 0$ gibt es immer ein n, so daß für $\delta > n$: $0 < x_{min}(\delta) \leq x_{max}(\delta)$ oder $x_{min}(\delta) \leq x_{max}(\delta) < 0$ gilt.

Nun nehmen wir im Falle $x_{min}(\delta) \leq 0 \leq x_{max}(\delta)$ immer an, $x = 0$ und verzweigen entsprechend. Falls $x = 0$ verzweigen wir richtig; falls $x \neq 0$ werden wir unseren Fehler bei wachsender Genauigkeit erkennen und revidieren können.

Da eine $\mathbb{R}$-Maschine für jede Eingabe nur endlich oft verzweigt, folgt: es gibt ein δ', so daß für alle $\delta \geq \delta'$ die $\mathbb{Q}$-Maschine an jeder Stelle richtig verzweigt.

Wie im Beweis des Simulationssatzes können Schwierigkeiten mit Polstellen oder endlosen Berechnungen auftreten; diese werden analog behandelt.

Auch in diesem Fall folgt die Robustheit, da wir keine spezifischen Eigenschaften der Rundung verwenden.

Wir können jedoch im Allgemeinen (wenn an mindestens einer Verzweigung noch $x = 0$ angenommen wird) nicht sagen, ob der Zeitpunkt, von dem an die Maschine richtig rechnet, schon erreicht ist.

Bemerkung: Daß die Funktionen, die von $\mathcal{M}_{\mathbb{R}}(f_1, \ldots, f_m)$ analytisch berechnet werden können, nicht $\mathbb{Q}$-analytisch sind, ist offensichtlich, da sie die $\mathbb{R}$-analytischen Funktionen enthalten, die schon nicht $\mathbb{Q}$-analytisch sind.

Kapitel 5

Der Hierarchiesatz

5.1 Das Halteproblem für analytische Maschinen

Die Annahme, eine Turingmaschine könne das Halteproblem für Turingmaschinen lösen, führt zu dem Widerspruch, daß es eine Maschine gibt, die genau dann hält, wenn sie nicht hält. Mit denselben Mitteln folgt, daß auch das Halteproblem der R-Maschinen nicht R-berechenbar ist. Es ist jedoch R-analytisch, wie man sich schnell klarmachen kann. Das Halteproblem für $\mathbb{R}$-Maschinen ist jedoch nicht $\mathbb{Q}$-analytisch: Man betrachte die $\mathbb{R}$-Maschine, die die rationalen Zahlen aufzählt, mit der Eingabe vergleicht und hält, wenn sie eine Zahl findet, die mit der Eingabe identisch ist. Gäbe es eine analytische $\mathbb{Q}$-Maschine, die das Halteproblem dieser $\mathbb{R}$-Maschine lösen könnte, so wäre die Funktion d_2 aus Abschnitt 3.3 $\mathbb{Q}$-analytisch.
Wir interessieren uns nun für das analytische Äquivalent des Halteproblems, das *Konvergenzproblem*: Ist es möglich, mittels einer analytischen Berechnung zu entscheiden, ob die Ausgabe einer beliebigen analytischen Maschine konvergiert oder divergiert?
Im weiteren verzichten wir darauf, irrationale Konstanten in $\mathbb{R}$-Maschinen einzubinden, und betrachten diese als zusätzliche Eingabewerte. Durch diese Einschränkung erreichen wir, daß es für jedes Programm eine endliche Darstellung gibt. Dann gibt es abzählbar unendlich viele, korrekte Programme für R-Maschinen. Sei Π die Menge aller korrekten Programme und sei für $\pi \in \Pi$ mit $\mathcal{M}_\pi$ die Maschine bezeichnet, auf der das Programm π abläuft. Es gibt eine berechenbare injektive Abbildung $\phi : \Pi \to \mathbb{Q}$, für die auch ihre Umkehrabbildung ϕ^{-1} berechenbar ist. Ferner definieren wir die Abbildung erg : $\Pi \times \mathbb{R}^* \to \mathbb{R} \cup \{u\}$, so daß für alle $x \in \mathbb{R}^*$ und $\pi \in \Pi$ gilt: $erg(\pi, x) = y$ falls die Ausgabe von $\mathcal{M}_\pi$ mit der Eingabe x gegen $y \in \mathbb{R}$ konvergiert und $erg(\pi, x) = u$ falls die Ausgabe von $\mathcal{M}_\pi$ divergiert.
Nun nehmen wir an, es gibt eine Maschine $\mathcal{M}_{\pi'}$, die entscheiden kann, ob die Ausgabe von einer beliebigen Maschine $\mathcal{M}_\pi$ mit einer Eingabe $x \in \mathbb{R}^*$ konvergiert oder divergiert. Wenn sie dies kann, dann kann sie es auch für die spezielle Eingabe $x = \phi(\pi)$. Also für alle $\pi \in \Pi$

$$erg(\pi', \phi(\pi)) := \begin{cases} 1 & \text{falls } erg(\pi, \phi(\pi)) = u \\ 2 & \text{sonst.} \end{cases}$$

Dann definieren wir ein π'', indem wir hinter π' eine Endlosschleife setzen, die abwechselnd 0 und 1 ausgibt, wenn $\mathcal{M}_{\pi'}$ Werte in der Nähe von 2 ausgibt. Dann gilt:

$$erg(\pi'', \phi(\pi)) = \begin{cases} 1 & \text{falls } erg(\pi, \phi(\pi)) = u \\ u & \text{sonst.} \end{cases}$$

Daraus folgt

$$erg(\pi'', \phi(\pi'')) = \begin{cases} 1 & \text{falls } erg(\pi'', \phi(\pi'')) = u \\ u & \text{falls } erg(\pi'', \phi(\pi'')) = 1 \end{cases}$$

Aus diesem Widerspruch folgt, daß die Annahme falsch war und daß die Frage der Konvergenz für R-Maschinen im Allgemeinen nicht R-analytisch endscheidbar ist.

5.2 Herleitung des Hierarchiesatzes

Zuerst werden wir das Verfahren aus Abschnitt 2.3, endliche Wörter in rationalen Speicherstellen abzulegen, erweitern, indem wir mittels analytischer Berechnungen unendliche Wörter in reellen Zahlen erzeugen. Wir werden dieses Verfahren dazu verwenden, um das Konvergenzproblem durch mehrere, hintereinandergestellte Berechnungen zu entscheiden. Damit folgt direkt der Hierarchiesatz.
Wir betrachten nun ein Verfahren, binäre Informationen in einer reellen Speicherzelle zu kodieren. Sei $r \in [0, 1]$ eine reelle Zahl und $r = \sum_{i=1}^{\infty} \frac{r_i}{2^i}$ mit $r_i \in \{0, 1\}$ die Binärdarstellung von r. Falls r ein endlicher Binärbruch ist, existieren für r zwei Darstellungen:

$$\sum_{i=1}^{n-1} \frac{r_i}{2^i} + \frac{1}{2^n} = \sum_{i=1}^{n-1} \frac{r_i}{2^i} + \frac{0}{2^n} + \sum_{i=n+1}^{\infty} \frac{1}{2^i}$$

Für alle anderen Zahlen sind die r_i für alle i eindeutig bestimmt. Wenn nun die r_i berechenbar sind für alle i, so ist r analytisch berechenbar. Ebenso sieht man, daß sich aus r die r_i berechnen lassen:

$$r_1 = \lfloor r \cdot 2 \rfloor$$

$$r_i = \lfloor r \cdot 2^i \rfloor - \lfloor r \cdot 2^{i-1} \rfloor \cdot 2$$

Es ist also möglich, eine reelle Zahl in ihre abzählbar unendlich vielen Binärstellen zu zerlegen ebenso wie abzählbar viele binäre Werte zu einer reellen Zahl zusammenzusetzen. Demnach sollte es auch möglich sein, abzählbar unendlich viele reelle Zahlen in einer darzustellen.
Sei $S = \{s^1, s^2, \ldots\}$ eine abzählbar unendliche Menge reeller Zahlen, und sei s^i_j die j-te Binärstelle von s^i, also ist

$$s^i = \sum_{j=1}^{\infty} \frac{s^i_j}{2^j}$$

Dann sei

$$c^S := \sum_{e=1}^{\infty} \frac{c^S_e}{2^e}$$

durch die c^S_e wie folgt definiert:

$$c^S_{g(i,j)} := s^i_j \qquad \text{wobei } g(i,j) := \sum_{l=1}^{i+j-1} l + j - 1$$

eine bijektive Abbildung von $\mathbb{N} \times \mathbb{N}$ nach $\mathbb{N}$ ist.
Wenn nun alle s^i berechenbar sind, so folgt, daß c^S analytisch berechenbar ist.
Wir betrachten nun den Fall, daß die s^i lediglich analytisch sind. Wir betrachten zwei Fälle:

1. Wenn s^i kein endlicher Binärbruch ist, gibt es für jedes j einen Zeitpunkt in der analytischen Berechnung, von dem an die j-te Binärstelle s^i_j präzise vorliegt. Dieser Zeitpunkt ist im Falle einer stark analytischen Berechnung bekannt, im Allgemeinen jedoch nicht.

2. Falls s^i ein endlicher Binärbruch ist, hängt die Konvergenz der Binärstellen davon ab, wie die analytische Berechnung konvergiert: Sind die Approximationswerte betragsmäßig fast immer größer als s^i, so gibt es für jedes j einen Zeitpunkt, von dem ab s^i_j wie in der endlichen Darstellung vorliegt. Sind die Approximationswerte betragsmäßig fast immer kleiner als s^i, gibt es für jedes j einen Zeitpunkt, von dem ab s^i_j wie in der unendlichen Darstellung, das heißt mit unendlich vielen Einsen am Ende, vorliegt. Falls jedoch die Approximationswerte um s^i oszillieren, werden die Binärstellen nicht fest.

Wenn wir nun den letzten Fall ausschließen können, dann sind für alle $i, j \in \mathbb{N}$ die Binärziffern s^i_j analytisch berechenbar in dem Sinne, daß es in der Berechnung einen Zeitpunkt gibt, von dem an sie präzise vorliegen. Dieser Zeitpunkt ist jedoch nicht bekannt. Wenn nun alle c^S_e analytisch berechnet werden[1] gibt es für jedes e einen Zeitpunkt von dem an die c^S_k für $k \leq e$ präzise sind. Also ist c^S analytisch berechenbar.
Wir fassen unsere Resultate zusammen:

Lemma 9: c^S ist eine Kodierung der Menge S.

Wenn alle s^i analytisch berechenbar sind, und es für alle endlichen Binärbrüche unter den s^i einen Algorithmus gibt, dessen Ausgabe nicht oszilliert, so ist auch c^S analytisch berechenbar.

Wird c^S angegeben, so kann daraus durch eine analytische Berechnung jedes s^i zurückgewonnen werden.

Definition 9: Für $i \in \mathbb{N}$ sei die Menge der *R^i-analytischen Funktionen* die Menge aller Funktionen, die durch i-fache Hintereinanderausführung von R-analytischen Funktionen gebildet werden, nämlich die Menge

$$\{g | g(x) = f_1(f_2(\ldots f_i(x)\ldots)) \text{ mit } f_1, \ldots, f_i \text{ sind } R\text{-analytisch}\}.$$

Satz 10: Das Konvergenzproblem einer analytischen R-Maschine ist R^3-analytisch.

[1] Es ist mittels diagonalisierender Verfahren möglich, unendlich vielen Berechnungen unendlich viel Rechenzeit zuzuteilen.

Beweis: Sei $\mathcal{M}$ eine beliebige analytische Maschine. Wir werden drei analytische Maschinen $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ angeben, die das Konvergenzproblem von $\mathcal{M}$ lösen, wenn die Ausgabe von $\mathcal{M}_1$ als Eingabe von $\mathcal{M}_2$ verwendet wird, und deren Ausgabe als Eingabe für $\mathcal{M}_3$.

Sei $(u)_i := (u_1, u_2, \ldots)$ die Folge der Ausgabewerte von $\mathcal{M}$. Jedes u_i ist berechenbar.

Wir betrachten zuerst den Fall, daß $(u)_i$ unbeschränkt ist.

Es ist

$$\max_{j \leq i}(|u_j|) = \max(|u_1|, |u_2|, \ldots, |u_i|)$$

berechenbar, und

$$\lim_{i\to\infty} \frac{1}{\max\limits_{j\leq i}(|u_j|)} \begin{cases} = 0 & \text{falls } (u)_i \text{ unbeschränkt ist,} \\ \neq 0 & \text{sonst.} \end{cases}$$

ist analytisch, somit können wir das Konvergenzproblem auf den Fall, daß $(u)_i$ beschränkt ist, einschränken.

Wir betrachten also den Fall $(u)_i$ beschränkt.

Es sei

$$S_i := \min_{l \in \mathbb{N}}(2^l | 2^l \geq \max_{j \leq i}(|u_j|))$$

die kleinste Potenz von 2, die eine obere Schranke für die $|u_j|\, j \leq i$ darstellt. Da $(u)_i$ beschränkt, gibt es ein i, so daß $S_j = S_i$ für alle $j > i$ und $-S_i \leq u_j \leq S_i$ für alle $j \in \mathbb{N}$.

Weiter sei $P := \{2, 3, 5, 7, \ldots\}$ die Menge aller Primzahlen. p_i, die i-te Primzahl, ist berechenbar.

Wir zerteilen die Intervalle $[-S_i, S_i]$ in p_i Unterintervalle. Sei $I_i = \{J_{i,1}, \ldots, J_{i,p_i}\}$ die Menge der Unterintervalle von $[-S_i, S_i]$ mit $J_{i,j} = [-S_i + (j-1)\frac{2S_i}{p_i}, -S_i + j\frac{2S_i}{p_i}]$. Wir bemerken, daß die Punkte, die auf Intervallgrenzen liegen, immer die Form haben:

$$-S_i + j\frac{2S_i}{p_i} = \frac{-S_i p_i + 2js_i}{p_i} = \frac{(2j - p_i)S_i}{p_i} \text{ mit } 0 < j < p_i$$

Im Falle $i = 1$ ist $p_1 = 2$ und $j = 1$, ist also $\frac{(2j-p_i)S_i}{p_i} = \frac{(2-2)S_i}{p_i} = 0$.
In allen anderen Fällen ist $2j - p_i \neq 0$ und p_i teilt weder $(2j - p_i)$ noch S_i, also ist $\frac{(2j-p_i)S_i}{p_i}$ ein teilerfremder Bruch. Damit folgt, daß es keine i, j, m, n mit $i \neq j$ gibt, so daß $J_{i,m}$ und $J_{j,n}$ exakt einen gemeinsamen Punkt haben. Dies gilt auch wenn $S_i \neq S_j$.

Da nach Voraussetzung $(u)_i$ beschränkt ist, hat $(u)_i$ zwei oder mehr Häufungspunkte, falls $(u)_i$ divergiert und exakt einen, falls $(u)_i$ konvergiert. Für alle i, k, l ist

$$\Phi_{k,l}(u_i) = \begin{cases} 1 & \text{falls } u_i \in J_{k,l} \\ 0 & \text{sonst} \end{cases}$$

berechenbar, also ist

$$\frac{1}{\sum_{i=1}^{n} \Phi_{k,l}(u_i)}$$

berechenbar für alle n, und

$$\Psi^u(k,l) := \lim_{n\to\infty} \frac{1}{\sum_{i=1}^{n} \Phi_{k,l}(u_i)}$$

ist analytisch. Man erkennt, daß

$$\Psi^u(k,l) \begin{cases} = 0 & \text{falls } (u)_i \text{ in } J_{k,l} \text{ einen Häufungspunkt hat,} \\ \neq 0 & \text{sonst.} \end{cases}$$

Dann sei wie oben:

$$c^\Psi := c^{\{\Psi^u(k,l) | k,l \in \mathbb{N}, l \leq p_k\}}$$

c^Ψ ist eine Kodierung aller $\Psi^u(k,l)$. Mit dem Lemma 9 folgt, daß c^Ψ analytisch berechenbar ist.

$\mathcal{M}_1$ berechnet also

$$c^\Psi \text{ und } \lim_{i\to\infty} \frac{1}{\max_{j\leq i}(|u_j|)}$$

Wir bemerken, daß die $\Psi^u(k,l)$ und $c(\Psi)$ auch dann definiert und analytisch berechenbar sind, wenn $(u)_i$ unbeschränkt ist.

Sei nun

$$\lambda^u(k,l) := \begin{cases} 0 & \text{falls } \Psi^u(k,l) = 0 \\ 1 & \text{sonst.} \end{cases}$$

Aus der Eingabe $c(\Psi)$ ist nun $\lambda^u(k,l)$ analytisch berechenbar, indem man die Binärstellen von $\Psi^u(k,l)$ ausliest und $\lambda^u(k,l) = 0$ annimmt, solange bis eine Stelle $\neq 0$ gefunden ist, dann kann man $\lambda^u(k,l) = 1$ setzen.

Die $\lambda^u(k,l)$ sind also in dem Sinne analytisch, daß sie irgendwann präzise vorliegen. Hiermit folgt, daß

$$\Lambda := \sum_{k=1}^{\infty} \sum_{l=1}^{p_k} \frac{\lambda^u(k,l)}{2^t} \text{ mit } t := \sum_{i=1}^{k-1} p_i + l$$

analytisch ist. $\mathcal{M}_2$ berechnet analytisch Λ.

$\mathcal{M}_3$ erhält nun Λ als Eingabe und berechnet daraus

$$\Omega^u := \begin{cases} 1 & \text{falls } u \text{ konvergiert} \\ 0 & \text{sonst.} \end{cases}$$

Die $\lambda^u(k,l)$ lassen sich aus Λ berechnen; ebenso $\sum\limits_l \lambda^u(k,l)$ für alle k, die Anzahl der $J_{k,l}$, die einen Häufungspunkt von $(u)_i$ enthalten.

Wenn nun $\sum\limits_l \lambda^u(k,l) = 2$ gibt es zwei $J_{k,l}$, die einen Häufungspunkt enthalten. Dieser könnte jedoch die gemeinsame Intervallgrenze sein. Da jedoch kein Punkt

Grenze von mehr als einem Intervallpaar ist, steht die Divergenz fest, wenn ein k' gefunden wird, so daß $\sum_l \lambda^u(k', l) = 2$. Ebenso ist es klar, daß $(u)_i$ divergiert, falls $\sum_l \lambda^u(k, l) > 2$.

$\mathcal{M}_3$ gibt nun 1 aus, solange nicht erwiesen ist, daß $(u)_i$ divergiert. Damit konvergiert $\mathcal{M}_3$ gegen Ω^u.

Wir betrachten nun die Frage, ob es für $i > 1$ eine Folge von i analytischen Maschinen gibt, die analytisch entscheiden können, ob eine beliebige Folge von i analytischen Maschinen mit einer beliebigen Eingabe konvergiert. Diese Frage wird mit nein beantwortet werden müssen, da auch in diesem Fall, analog zum Halteproblem, eine Paradoxie konstruiert werden kann. Jedoch erkennt man, daß $3i$ Maschinen immer in der Lage sind, die Konvergenz von i Maschinen zu entscheiden. Nun folgt der

Satz 11 (Hierarchiesatz): Für alle $i \in \mathbb{N}$ ist die Menge der R^i-analytischen Funktionen echt in der Menge der R^{i+1}-analytischen Funktionen enthalten.

Beweis: Der Beweis wird mittels Induktion über i geführt.

Annahme: Die R-analytischen Funktionen sind genauso mächtig wie die R^2-analytischen. Damit würde folgen, daß die R-analytischen Funktionen genau so mächtig wie die R^3-analytischen Funktionen wären. Das wäre ein Widerspruch dazu, daß das Konvergenzproblem der R-analytischen Maschinen nicht von einer R-analytischen Maschine entschieden werden kann.

Annahme: Die R^i-analytischen Funktionen sind genauso mächtig wie die R^{i+1}-analytischen. Daraus würde folgen, daß die R^i-analytischen Funktionen die Mächtigkeit der R^j-analytischen Funktionen besäßen für alle $j > i$. Dieses wäre ein Widerspruch dazu, daß das Konvergenzproblem der R^i-analytischen Maschinen nicht durch eine R^i-analytische Funktion, wohl aber durch eine R^{3i}-analytische Funktion entscheidbar ist.

Damit ist der Satz gezeigt.

Ausblicke

Der Hierarchiesatz besagt, daß die R^i-analytischen Funktionen die R^{i-1}-analytischen Funktionen echt enthalten, läßt aber offen, wie die $\mathbb{Q}^i$-analytischen Funktionen zu den $\mathbb{R}^j$-analytischen stehen. Es läßt sich zeigen, daß jede $\mathbb{R}^i$-analytische Funktion von $2i$ analytischen $\mathbb{Q}$-Maschinen berechnet werden kann. Das scheinen recht viele zu sein, möglicherweise kommt man mit weniger $\mathbb{Q}$-Maschinen aus.
Eventuell kann man die Funktionenklassen noch weiter verfeinern. Es gibt Funktionen, die nicht R-analytisch sind, jedoch von einer analytischen Maschine und einer Maschine, die die Ergebnisse der analytischen Maschine weiterverarbeitet und terminiert, berechnet werden können. Wir brauchen also in diesen Fällen nur einen Grenzprozess durchzuführen. Man könnte diese Funktion beispielsweise als $R^{1\frac{1}{2}}$-analytisch bezeichnen.
Im Hinblick auf die Anwendbarkeit sind die stark analytischen Funktionen interessant, da sie bezüglich der Hintereinanderausführung abgeschlossen sind und für jede Iteration der Berechnung eine obere Schranke für den Fehler angegeben werden kann. Damit ist ein Großteil der stetigen Funktionen, mit denen man es in der Praxis zu tun hat, stark analytisch. Bedauerlich ist, daß unstetige Funktionen an den Unstetigkeitsstellen nicht stark analytisch sind.
Was in dieser Arbeit völlig ausgespart wurde, sind Laufzeitabschätzungen, die natürlich für die Anwendbarkeit in der Praxis entscheidend sind. Gibt es Probleme, die sich mittels Algorithmen, die unseren analytischen oder stark analytischen Funktionen nachempfunden sind, effizienter lösen lassen als mit herkömmlichen numerischen Verfahren?
Die Menge der Funktionen, die von einer analytischen R-Maschine berechnet werden können, die nur endlich oft ein unpräzises Ergebnis liefert, irgendwann jedoch das exakte Resultat ausgibt, ist ebenfalls gegenüber der Hintereinanderausführung abgeschlossen. Wenn wir nun eine analytische Maschine betrachten, die eine Fehlerabschätzung für ihre Ausgabe berechnet, die endlich oft falsch ist, irgendwann jedoch korrekt wird, erhalten wir eine Klasse von *quasi stark analytischen* Funktionen, die ebenfalls bezüglich der Hintereinanderausführung abgeschlossen sind, und die auch unstetige Funktionen umfaßt. Die $\mathbb{R}$-berechenbaren Funktionen gehören beispielsweise dieser Klasse an. Somit wird man auch diese Funktionen als Prozeduren in eine $\mathbb{R}$-Maschine einbauen können, ohne den Simulationssatz zu verlieren.
Die stark $\mathbb{Q}$-analytischen Funktionen sind bezüglich der Hintereinanderausführung abgeschlossen. Umgekehrt ist es möglich, einige stark analytische Funktionen in mehrere zu zerlegen – im Extremfall sogar bis zu den Elementaroperationen, die auch stark analytisch sind. Sei nun g eine stark analytische Funktion, die in n stark analytische Funktionen $f_1, f_2, \ldots, f_n$ zerlegt werden kann, so daß $g(x) = f_n(f_{n-1}(\ldots f_2(f_1(x))\ldots))$. Nun ist es möglich, g durch n Maschinen $\mathcal{M}_1, \ldots, \mathcal{M}_n$ zu berechnen, wenn $\mathcal{M}_i$ f_i berechnet, und wenn $\mathcal{M}_{i+1}$ als Eingabe die Ausgabe von $\mathcal{M}_i$ erhält, sobald diese hinreichend genau ist.
Wieviel Zeit durch diese *Parallelisierung* gewonnen werden kann, hängt unter anderem damit zusammen, in welcher Weise sich die Ausgangsfunktion zerlegen läßt, und wie lange die Maschinen $\mathcal{M}_2, \ldots, \mathcal{M}_n$ auf ihre Vorgänger warten müssen. Dieses könnte Gegenstand weiterer Untersuchungen sein.

Literaturverzeichnis

[A.A] Arnold, V.I.; Avez, A.
Ergodic Problems of Classical Mechanics
Addison-Wesley, Advanced Book Classics (1989)

[BenOr] Ben-Or, M.
Lower Bounds For Algebraic Computation Trees
Proceedings 15th ACM STOC, 1983, 80-86

[B.S.S] Blum, L.; Shub, M.; Smale, St.
On a theory of computation and complexity over the real numbers: NP completeness, recursive functions and universal machines
Bull AMS Vol 21 (1989)

[Col] Collins, G.E.
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition
Proceedings of EUROSAM 74, SIGSAM Bulletin, Vol. 8,No. 3 (1974)

[Grz] Grzegorczyk, A.
On the definitions of computable real continuous functions
Fund. Math. 44, 61-71

[Ho] Hotz, G.
Über Berechenbarkeit fraktaler Strukturen
Abhandlungen der Akademie der Wissenschaften und Literatur, Math.-Nat. Klasse. (1994, Nr. 1)

[H.V.S] Hotz, G.; Vierke, G.; Schieffer, B.
Analytic Machines
Electronic Colloquium on Computational Complexity (1995)

[J] Julia, G.
Sur l'iteration des fonctions rationelles
Jour. de Math. Pure et Appl. 8 (1918)

[Ko] Ko, Ker-I
Complexity Theory of Real Functions
Birkhäuser Boston (1991)

[Kön] Königsberger, K.
Analysis 1
Springer Verlag (1990)

[P] Paul, W.J.
Komplexitätstheorie
B.G. Teubner Stuttgart (1978)

[S] Siegel, C.L.
Vorlesungen über Himmelsmechanik (Kap. 3)
Springer Verlag (1956)

[T] Tarski, A.
A Decision Method for Elementary Algebra and Geometry
second ed., rev., Univ. of California Press, Berkeley. (1951)

[Wei] Weihrauch, K.
A Simple Introduction to Computable Analysis
Informatik-Berichte Nr. 171, Fernuniversität, Hagen (1995)