12. Theorietag

„Automaten und Formale Sprachen”

mit Workshop

„Berechenbarkeit und Komplexität in der Analysis”

Proceedings

Lutherstadt Wittenberg
23.-26.9.2002

René Mazala – Ludwig Staiger – Renate Winter (Hrsg.)

Vorwort

Die Theorietage „Automaten und Formale Sprachen” haben bereits eine lange Tradition. 1991 von der GI-Fachgruppe 0.1.5 *Automaten und Formale Sprachen* ins Leben gerufen, finden sie jedes Jahr zusammen mit einem Workshop und der Fachgruppensitzung statt. Die Serie begann 1991 in Magdeburg, wurde 1992 in Kiel, 1993 in Dagstuhl, 1994 in Herrsching, 1995 auf Schloss Rauischholzhausen, 1996 in Cunnersdorf, 1997 in Barnstorf, 1998 in Riveris, 1999 in Schauenburg-Elmshagen, 2000 in Wien und 2001 in Wendgräben durchgeführt.

In diesem Jahr findet der Theorietag mit dem vorangestellten Workshop zum Thema „Berechenbarkeit und Komplexität in der Analysis” in der Lutherstadt Wittenberg statt. Im Jubiläumsjahr anlässlich der 500-Jahrfeier der Martin-Luther-Universität Halle-Wittenberg wurde die Leucorea, eine Stiftung des öffentlichen Rechts an der Martin-Luther-Universität Halle-Wittenberg, als Veranstaltungsort gewählt.

21 Teilnehmer aus Deutschland und Österreich folgten der Einladung. Das wissenschaftliche Programm enthält 4 Workshop-Vorträge und 11 Vorträge zum Theorietag. Die Vortragenden zum Workshop sind:

- Vasco Brattka (Hagen)
- Peter Hertling (Hagen)
- Norbert Müller(Trier)
- Klaus Weihrauch (Hagen)

Die Kurzfassungen der Vorträge zum Workshop und Theorietag sind im vorliegenden Tagungsband enthalten. Außerdem finden Sie hier die Programme und eine Liste aller Teilnehmer mit ihren Adressen.

Der Gesellschaft für Informatik sowie der Stiftung Leucorea und der Georg-Cantor-Vereinigung gebühren Dank für die Unterstützung des Theorietages. Allen Teilnehmenden wünschen wir einen interessanten und erfolgreichen Theorietag sowie einen angenehmen Aufenthalt in der Lutherstadt Wittenberg.

R. Mazala

L. Staiger

R. Winter

Halle-Wittenberg, im September 2002

Inhaltsverzeichnis

Programm des Workshops

„Berechenbarkeit und Komplexität in der Analysis"

Dienstag, 24.September 2002

09.25 Uhr: Begrüßung der Teilnehmer zum Workshop

09.30 - 10.30 Uhr: KLAUS WEIHRAUCH (Hagen):
Rechnen im Bereich der reellen Zahlen

10.30 - 11.00 Uhr: Kaffeepause

11.00 - 12.00 Uhr: VASCO BRATTKA (Hagen):
Berechenbare Teilmengen metrischer Räume

12.00 - 13.30 Uhr: Mittagspause

13.30 - 14.30 Uhr: PETER HERTLING (Hagen):
Approximierbarkeit und Zufälligkeit reeller Zahlen

14.30 - 15.00 Uhr: Kaffeepause

15.00 - 16.00 Uhr: NORBERT MÜLLER (Trier):
Komplexität in der Analysis

Programm des Theorietages

„Automaten und Formale Sprachen”

Mittwoch, 25.September 2002

09.25 Uhr: Begrüßung zum Theorietag

09.30 - 09.55 Uhr: BJÖRN BORCHARDT (Dresden):
Determinization of bottom-up finite state weighted tree automata

09.55 - 10.20 Uhr: MARTIN KUTRIB (Giessen):
Self-Assembling Finite Automata

10.20 - 10.40 Uhr: Kaffeepause

10.40 - 11.05 Uhr: RENÉ MAZALA (Halle):
Berechnung der Hausdorff-Dimension regulärer ω-Sprachen

11.05 - 11.30 Uhr: JAN-HENRIK ALTENBERND (Aachen):
Tiling-Systeme über unendlichen Bildern und ihre Akzeptanzbedingungen

11.30 - 13.30 Uhr: Mittagspause

13.30 - 13.55 Uhr: JAN-THOMAS LÖWE (Giessen):
Mustererzeugung mit Zellularautomaten

13.55 - 14.20 Uhr: RENATE WINTER (Halle):
Möglichkeiten der Lösung des CLIQUE-Problems

14.20 - 14.45 Uhr: RUDOLF FREUND (Wien, Österreich):
P-Systeme mit Energiekontrolle

14.45 - 15.10 Uhr: MARION OSWALD (Wien, Österreich):
Verallgemeinerte P-Systeme mit verbotenem Kontext

15.10 - 15.30 Uhr: Kaffeepause

16.30 - 17.30 Uhr: Fachgruppensitzung

Donnerstag, 26.September 2002

09.30 - 09.55 Uhr: ANDREAS KLEIN (Kassel):
Verallgemeinerte kontextfreie Grammatiken

09.55 - 10.20 Uhr: MARKUS HOLZER (München):
On Deterministic Finite Automata and Syntactic Monoid Size

10.20 - 10.40 Uhr: Kaffeepause

10.40 - 11.05 Uhr: ROMAN KÖNIG (Erlangen):
Berechnung von Implikationen

11.30 - 13.30 Uhr: Mittagspause und Ende des Theorietages

Computable Subsets of Metric Spaces

VASCO BRATTKA
Theoretische Informatik I, FernUniversität Hagen
58084 Hagen, Germany
e-mail: `Vasco.Brattka@FernUni-Hagen.de`

ABSTRACT

We discuss several generalizations of the classical notions of r.e., co-r.e. and recursive subsets to subsets of metric spaces. Moreover, we demonstrate how some of these notions can be applied in computable analysis.

Computable analysis is the Turing machine based theory of computable functions and subsets on Euclidean space and other topological spaces, see [8, 6, 9]. One interesting line of research considers the question how the classical notions of computable subsets (such as r.e., co-r.e. and recursive subsets) can be generalized appropriately to subsets of Euclidean space and other metric spaces, see [7, 10, 5, 12, 11, 4, 3]. Several notions which we will define precisely, are given in Figure 1, together with their logical relations. Actually, any arrow does not only indicate a logical implication, but even the existence of a uniform computable translation from one type of sets into the other. We will express these properties using the tools of the representation based approach to computable analysis [9].

$$\begin{tikzcd}
 & \text{recursive} \arrow[r] \arrow[dl] & \text{co-r.e.} \\
\text{r.e.} = \text{upper semi-located} & \text{located} \arrow[l] \arrow[u] \arrow[r] & \text{lower semi-located} \arrow[u] \\
\text{effectively separable} \arrow[u] & \text{strongly recursive} \arrow[u] \arrow[r] & \text{strongly co-r.e.} \arrow[u]
\end{tikzcd}$$

Figure 1: Notions of effectivity for closed subsets of computable metric spaces

While all these notions have to be distinguished in the general case of separable metric spaces, it turns out that the three horizontal layers of the diagram collapse to a single layer in case of Euclidean space (and other “effectively locally compact” metric spaces), see [4, 3, 2]. In any case, these notions are generalizations of the classical notions of computable subsets, as the following statement shows, see [4].

Remark A subset $A \subseteq \mathbb{N}$ is r.e., co-r.e., or recursive in the classical sense, if and only if it has the corresponding property, considered as a closed subset $A \subseteq \mathbb{R}$.

The introduced notions are not just formal generalizations of the classical notions but even useful in computable analysis. This can be conclusively demonstrated, at least for the Euclidean case, by results as the following, see [12, 9].

Theorem (Computable Graph Theorem) *A continuous function* $f : \mathbb{R}^n \to \mathbb{R}$ *is computable, if and only if* $\mathrm{graph}(f) := \{(x, f(x)) : x \in \mathbb{R}^n\}$ *is a recursive closed subset of* $\mathbb{R}^{n+1}$.

[0]Work supported by DFG Grant BR 1807/4-1

Thus, the notion of a computable function could be redefined with the notion of a recursive closed subset. This proves that recursive closed subsets occur very naturally. We will show that the same holds in the general case of computable metric spaces (which are roughly speaking, separable metric spaces with the additional property that the metric is computable). For instance one can prove the following computable version of the classical Closed Graph Theorem, see [1].

Theorem (Computable Closed Graph Theorem) *Let X, Y be computable Banach spaces and let $T : X \to Y$ be a linear operator. Then $T : X \to Y$ is computable, if and only if* $\mathrm{graph}(T) = \{(x, Tx) : x \in X\}$ *is a recursive closed subset of* $X \times Y$.

However, there is a remarkable difference between the Computable Graph Theorem and the Computable Closed Graph Theorem. While in the case of the first theorem, the relation between the function and its graph is computable in a uniform sense, that is the mapping $f \mapsto \mathrm{graph}(f)$ as well as its inverse are computable in a well-defined sense, the statement for the inverse does not hold true in case of the second theorem. That is, a description of a given closed graph of a linear operator $T : X \to Y$ cannot be tranformed into a description of T itself, in general. We will demonstrate that these different degrees of uniformity of computable versions of classical theorems can be concisely expressed in the representation based approach to computable analysis.

References

[1] Vasco Brattka. Computability of Banach space principles. Informatik Berichte 286, Fern-Universität Hagen, Fachbereich Informatik, Hagen, June 2001.

[2] Vasco Brattka. Random numbers and an incomplete immune recursive set. In P. Widmayer et. al. (eds.), *Automata, Languages and Programming*, vol. 2380 of *Lect. Notes Comp. Sci.*, pp. 950–961, Berlin, 2002. Springer. ICALP 2002, Málaga, Spain, July 8–13, 2002.

[3] Vasco Brattka and Gero Presser. Computability on subsets of metric spaces. *Theoret. Comp. Sci.*, accepted for publication.

[4] Vasco Brattka and Klaus Weihrauch. Computability on subsets of Euclidean space I: Closed and compact subsets. *Theoret. Comp. Sci.*, 219:65–93, 1999.

[5] Xiaolin Ge and Anil Nerode. On extreme points of convex compact Turing located sets. In A. Nerode and Yu. V. Matiyasevich (eds.), *Logical Foundations of Computer Science*, vol. 813 of *Lect. Notes Comp. Sci.*, pp. 114–128, Berlin, 1994. Springer. LFCS'94, St. Petersburg, Russia, July 11–14, 1994.

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

[7] Daniel Lacombe. Les ensembles récursivement ouverts ou fermés, et leurs applications à l'Analyse récursive. *Comptes Rendus Académie des Sciences Paris*, 245:1040–1043, 1957.

[8] Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics.* Springer, Berlin, 1989.

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

[10] Klaus Weihrauch and Christoph Kreitz. Representations of the real numbers and of the open subsets of the set of real numbers. *Ann. Pure Appl. Logic*, 35:247–260, 1987.

[11] Mariko Yasugi, Takakazu Mori, and Yoshiki Tsujii. Effective properties of sets and functions in metric spaces with computability structure. *Theoret. Comp. Sci.*, 219:467–486, 1999.

[12] Qing Zhou. Computable real-valued functions on recursive open and closed subsets of Euclidean space. *Math. Logic Quart.*, 42:379–409, 1996.

Approximierbarkeit und Zufälligkeit reeller Zahlen

PETER HERTLING
Lehrgebiet Berechenbarkeit und Logik, FernUniversität Hagen, 58084 Hagen
e-mail: `peter.hertling@fernuni-hagen.de`

KURZFASSUNG

Der Vortrag soll einige Aspekte der Berechenbarkeit reeller Zahlen beleuchten. Wir beginnen mit der Frage, welche reellen Zahlen berechenbar sind. Diese Frage ist von Turing in der Arbeit beantwortet worden, in der er das nach ihm benannte Maschinenmodell entwickelt hat. Dann werden wir verschiedene Darstellungen reeller Zahlen behandeln, insbesondere Dezimalbruchdarstellungen und Darstellungen durch konvergente Folgen rationaler Zahlen. Wir betrachten eingehend neben den berechenbaren Zahlen auch die größere Klasse der nur von links effektiv approximierbaren Zahlen sowie die Klasse der Martin–Löf–zufälligen Zahlen.

Schlüsselwörter: Berechenbare Analysis, berechenbare reelle Zahlen, Zahlendarstellungen, Programmkomplexität, zufällige Zahlen, linksberechenbare reelle Zahlen, Ω–Zahlen, Solovay–Reduzierbarkeit.

Welche reellen Zahlen sind berechenbar? Diese Frage war eine der Motivationen für Turing, über Berechenbarkeit nachzudenken und das nach ihm benannte Maschinenmodell, das Turingmaschinenmodell, zu entwickeln. Die Arbeit, in der er seine Überlegungen ausgeführt hat, trägt den Titel "On computable numbers, with an application to the Entscheidungsproblem" (1936). Man beachte, dass hier mit "computable numbers" berechenbare reelle Zahlen gemeint sind. In dieser Arbeit nennt Turing eine reelle Zahl dann berechenbar, wenn ihr nichtganzzahliger Anteil eine Binärdarstellung besitzt, die eine berechenbare Zeichenfolge ist. Es ist leicht zu sehen, dass sich der Begriff der Berechenbarkeit einer reellen Zahl nicht ändert, wenn man anstelle der Binärdarstellung eine Dezimalbruchdarstellung zu irgendeiner anderen Basis betrachtet. In einer Korrektur (1937) zu der oben genannten Arbeit stellte Turing allerdings fest, dass es keinen Algorithmus gibt, der aus einer rationalen Intervallschachtelung einer reellen Zahl eine Binärdarstellung dieser Zahl berechnet. Daher führte Turing dann eine Darstellung reeller Zahlen ein, bei der eine reelle Zahl durch eine stark normierte rationale Intervallschachtelung beschrieben wird. Mit Hilfe dieser und dazu äquivalenten Darstellungen kann man nun in sinnvoller Weise nicht nur berechenbare reelle Zahlen definieren, sondern auch auf reellen Zahlen rechnen, siehe Weihrauch (2000), sowie eine realistische Komplexitätstheorie für das Rechnen mit reellen Zahlen entwickeln, siehe Ko (1991). Es ist leicht zu sehen, dass diese Darstellung über rationale Intervallschachtelungen äquivalent ist zu einer Darstellung durch schnell konvergente Folgen rationaler Zahlen: eine Zahl x ist berechenbar, wenn es eine berechenbare Folge $q_0, q_1, q_2, \ldots$ von rationalen Zahlen gibt mit $|x - q_i| \leq 2^{-i}$ für alle i. Anstelle einer derart schnellen Konvergenz kann man auch andere Bedingungen an die Konvergenz rationaler Folgen stellen und erhält dann andere Klassen von approximierbaren reellen Zahlen. Wir wollen hier einige exemplarische Ergebnisse zu Dezimalbruchdarstellungen von reellen Zahlen und zu Klassen approximierbarer reeller Zahlen vorstellen.

Zuerst gehen wir auf die Dezimalbruchdarstellungen von reellen Zahlen ein. Eine sehr natürliche Frage ist die Frage, ob es einen Algorithmus gibt, der jede Darstellung einer jeden Zahl zu einer Basis b effektiv in eine Darstellung der gleichen Zahl zu einer anderen Basis c überführt. Es ist ein klassisches Ergebnis von Mostowski und Uspensky, dass es einen derartigen Algorithmus genau dann gibt, wenn alle Primteiler von b auch Primteiler von c sind. Dies zeigt, dass die Darstellungen von Zahlen zu verschiedenen Basen nicht äquivalent sind. Eine andere natürliche Frage ist, welche Eigenschaften von Zeichenfolgen invariant unter der Wahl der Basis sind. Wir hatten oben angemerkt, dass zum Beispiel der Begriff der Berechenbarkeit invariant ist: hat x eine berechenbare Darstellung zur Basis b, so hat x auch zu jeder anderen Basis c eine berechenbare Darstellung. Weitere interessante Begriffe sind die der einfachen Normalität (jedes Zeichen tritt mit der gleichen Häufigkeit auf), der Normalität (jedes Wort tritt mit der zu erwartenden Häufigkeit auf), der Disjunktivität (jedes Wort tritt mindestens einmal auf) und der Zufälligkeit nach Martin–Löf (1966). Wir werden erläutern, wie sich diese Begriffe bei einem Basiswechsel verhalten. Insbesondere ist auch der Begriff der Zufälligkeit invariant unter der Wahl der Basis (Calude und Jürgensen 1994). Martin–Löf hat den Begriff der Zufälligkeit durch Zufallstest eingeführt. Es ist aber ein klassisches Ergebnis (Schnorr, siehe Chaitin 1975) dass die Martin–Löf–zufälligen Zeichenfolgen gerade die Zeichenfolgen mit einer besonders hohen Programmkomplexität oder Kolmogorovkomplexität sind.

Als besonders interessant hat sich in den letzten Jahren die Untersuchung des Zusammenhangs zwischen der Programmkomplexität der Binärdarstellungen reeller Zahlen und der Approximierbarkeit reeller Zahlen herausgestellt. Zufällige Zahlen, d.h. Zahlen mit einer sehr hohen Programmkomplexität, können nicht berechenbar sein. Es gibt jedoch Zahlen, die zufällig sind, aber dennoch in einem schwächeren Sinne effektiv approximierbar sind. Gemeint sind die linksberechenbaren oder rekursiv aufzählbaren Zahlen. Eine Zahl x ist linksberechenbar, wenn es eine berechenbare, monoton wachsende Folge rationaler Zahlen gibt, die gegen x konvergiert. Man beachte, dass nichts über die Konvergenzgeschwindigkeit der Folge rationaler Zahlen vorausgesetzt wird. Jede berechenbare Zahl ist linksberechenbar, aber Specker (1949) hat bereits gezeigt, dass es linksberechenbare Zahlen gibt, die nicht berechenbar sind. Chaitin (1975) hat sogar Zahlen konstruiert, die einerseits linksberechenbar sind, andererseits aber auch in hohem Maße ineffektiv, nämlich zufällig im Sinne von Martin–Löf sind: die sogenannten Ω–Zahlen. Diese Zahlen sind auch dadurch berühmt geworden, dass ihre Binärdarstellung in besonders komprimierter Form die Fragen auf viele mathematische Fragen enthält. Um diese Zahlen besser zu verstehen, hat Solovay (1975) eine Klasse von Zahlen eingeführt, die er Ω–ähnlich nannte. Diese Zahlen haben die Eigenschaft, dass man sie nur sehr langsam effektiv von links approximieren kann: jede berechenbare, monoton wachsende Folge rationaler Zahlen, die konvergiert, tut dies höchstens so schnell (bis auf einen konstanten Faktor) wie irgendeine berechenbare, monoton wachsende Folgen rationaler Zahlen, die gegen eine Ω–ähnliche Zahl konvergiert. Solovay zeigte, dass die Ω–Zahlen von Chaitin alle Ω–ähnlich sind (daher diese Bezeichnung), und dass alle Ω–ähnlichen Zahlen zufällig (und natürlich linksberechenbar) sind. Calude, Hertling, Khoussainov und Wang (1998, 2001) haben erst vor kurzem gezeigt, dass auch die Umkehrung der ersten Inklusion gilt: tatsächlich sind alle Ω–ähnlichen Zahlen sogar Ω–Zahlen. Das Bild wurde anschließend durch Kučera und Slaman (2001) vervollständigt, die gezeigt haben, dass auch die Umkehrung der zweiten Inklusion gilt. Das heißt, die Klasse der Ω–Zahlen nach Chaitin, die Klasse der Ω–ähnlichen Zahlen nach Solovay und die Klasse der Zahlen, die linksberechenbar und zufällig sind, sind alle drei identisch.

In seiner Untersuchung der Ω–ähnlichen Zahlen hatte Solovay eine Relation zwischen linksberechenbaren Zahlen eingeführt, die solche Zahlen daraufhin vergleicht, wie schnell man sie durch berechenbare, monoton wachsende Folgen rationaler Zahlen approximieren kann. Zu dieser Relation und zu ihrem Verhältnis zur Programmkomplexität der Binärdarstellungen der jeweiligen Zahlen gibt es eine Reihe von neuen und interessanten Ergebnissen von Downey, Hirschfeldt, Nies und anderen. Auch Rettinger, Zheng und andere haben in einer Reihe von Arbeiten linksberechenbare reelle Zahlen daraufhin untersucht, wie schnell man sie effektiv von links approximieren kann. In diesem Gebiet gibt es auch noch sehr interessante offene Probleme.

Komplexität in der Analysis

Norbert Th. Müller
Abteilung Informatik, Universität Trier
D-54286 Trier
e-mail: mueller@uni-trier.de

KURZFASSUNG
Wir betrachten effiziente Algorithmen für reelle Funktionen. Im Falle ausreichender analytischer Eigenschaften bleiben gute Komplexitätseigenschaften auch nach Anwendung grundlegender Operatoren (Fixpunkt-Operatoren, Differentiation, Integration, ...) erhalten. Ansonsten ergeben sich Zusammenhänge mit Problemen aus der diskreten Komplexitätstheorie wie $P = NP$ oder $FP = \#P$.

Schlüsselwörter: TTE, Zeit-Komplexität

1. Einleitung

Basis einer Komplexitätstheorie in der Analysis ist das zugrundeliegende Berechenbarkeitsmodell. Hier gibt es viele Möglichkeiten, manche davon realitätsnah (d.h. einer Implementierung zugänglich), manche weniger. Wir werden hier das Berechenbarkeitsmodell der „Type-2 Theorie of Effectivity“ verwenden, das z.B. in [We00] detailliert untersucht und in [Mu01] implementiert wird. Dabei definieren wir die Zeit-Komplexität reeller Funktionen $\mathbb{FR} := \{f :\subseteq \mathbb{R}^n \to \mathbb{R}^m \mid n, m \in \mathbb{N}\}$ wie folgt:

- Gegeben seien $f \in \mathbb{FR}$ und eine Teilmenge $G \subseteq \mathrm{Def}(f)$. f ist in Zeit $t : \mathbb{N} \to \mathbb{N}$ auf G berechenbar, wenn t eine obere Schranke für die Zahl der Rechenschritte zur Bestimmung von Approximationen a mit $|a - f(\overline{x})| \leq 2^{-n}$ für beliebige $\overline{x} \in G$ ist.

Sinnvoll anwendbare und ausreichend allgemeine Definitionen der Komplexität reeller Operatoren gibt es noch nicht. Statt dessen betrachten wir die folgende Fragestellung:

- Gegeben sei ein Operator $\Phi :\subseteq \mathbb{FR} \to \mathbb{FR}$. Welche Rückschlüsse auf die Komplexität von $\Phi(f)$ kann man aus der Komplexität einer Funktion f und evtl. spezieller Eigenschaften von Φ bzw. f treffen?

2. Komplexität von Zahlen und Funktionen

Startpunkt sind die Algorithmen für die elementare reelle Arithmetik, die man aus den Algorithmen für natürliche Zahlen ableiten kann (vgl. [Kn81]). Die folgenden Schranken gelten jeweils für beliebige kompakte Teilmengen der Definitionsbereiche der Funktionen:

Addition und Subtraktion reeller Zahlen sind in linearer Zeit möglich, für Multiplikation und Division reicht die Zeit $\mathcal{O}(M(n))$ mit $M(n) := n \cdot \log n \cdot \log \log n$. Rationale Funktionen und x^q mit $q \in \mathbb{Q}$ sind ebenfalls in Zeit $\mathcal{O}(M(n))$ berechenbar.

Das arithmetisch-geometrische Mittel bildet die Basis für die schnellsten bekannten Algorithmen vieler Standardfunktionen: Die Konstanten π, e, $\ln(2)$ ebenso wie die Funktionen $\sin(x)$, $\cos(x)$, $\ln(x)$, e^x,... sind in Zeit $\mathcal{O}(M(n) \cdot \log n)$ berechenbar. Hier ist speziell die Arbeit [Bt75] zu nennen, in der die Komplexität von Fließkommaarithmetik mit variabler Genauigkeit untersucht wurde. Die dortigen Ergebnisse sind in unser Modell übertragbar.

3. Reelle Operatoren

Die Konstruktion von Algorithmen für reelle Funktionen entspricht sehr häufig der Anwendung spezieller Operatoren Φ. So ist etwa die Quadratwurzel $\sqrt{x}$ als *Inverse* der Funktion $x \mapsto x^2$ bzw. als *Lösung der Nullstellensuche* bei $y \mapsto y^2 - x$ interpretierbar (und oft sogar so implementiert).

Bei der Abschätzung der Komplexität der entstehenden Algorithmen wird dabei immer wieder die folgende Eigenschaft vieler Schrankenfunktionen wichtig, die z.B. bei Funktionen der Form $n^\alpha(\log n)^\beta(\log\log n)^\gamma$ für $\alpha \geq 1$ und $\beta, \gamma \geq 0$ vorliegt:

- Eine Funktion $t : \mathbb{N} \to \mathbb{N}$ heißt regulär, wenn sie monoton wächst und die Eigenschaft $2 \cdot t(n) \leq t(2n) \leq c \cdot t(n)$ für ausreichend große n hat.

Für viele interessante Operatoren lässt sich die Komplexität des Resultates $\Phi(f)$ bei der Anwendung auf Argumente f aus der Komplexität t_f von f abschätzen, wenn f ausreichend gute analytische Eigenschaften hat. Im folgenden sei t_f eine reguläre Zeitschranke; implizit ist f damit von polynomialer Zeitkomplexität. Einige wichtige Beispiele sind:

- ξ sei Fixpunkt von f. Konvergiert f linear gegen ξ, so hat ξ die Komplexität $\mathcal{O}(n \cdot t_f(n))$. Ist die Konvergenz superlinear, so hat ξ die Komplexität $\mathcal{O}(t_f(n))$.
- Ist f zweimal stetig differenzierbar, so hat f' die Komplexität $\mathcal{O}(t_f(n))$. Ist dabei $f'(x) \neq 0$ auf einem rationalen Intervall $[a, b]$, so ist auch die Umkehrfunktion f^{-1} in Zeit $\mathcal{O}(t_f(n))$ auf $[a, b]$ berechenbar. Damit ist auch jede einfache Nullstelle von f in $\mathcal{O}(t_f(n))$.
- Ist f analytisch (d.h. in eine Potenzreihe entwickelbar) auf einem rationalen Intervall $[a, b]$, so ist das Integral $\int_a^x f(y)dy$ in $\mathcal{O}(n^2 \cdot t_f(n))$.

4. Reelle vs. diskrete Komplexitätstheorie

Ist eine Funktion f analytisch, so ist sie bereits durch den Funktionsverlauf auf einem beliebig kleinen Intervall festgelegt. Für viele Operatoren Φ hat dann auch $\Phi(f)$ eine Komplexität, die mit der von f vergleichbar ist. Andererseits lassen sich Beispiele finden, in denen $\Phi(f)$ kompliziert ist, obwohl $f \in C^\infty(\mathbb{R})$ und polynomzeit-berechenbar ist (kurz: $P{+}C^\infty$). Hier ist es nämlich möglich, diskrete Probleme in reellen Funktionen zu codieren. Diese Querbeziehungen zur diskreten Komplexitätstheorie finden sich bereits in [Ko91]:

- Jedes berechenbare $x \in \mathbb{R}$ ist Nullstelle eines streng monoton wachsenden $f \in P{+}C^\infty$.
- $P = NP \Leftrightarrow$ für jedes $f \in P{+}C^\infty$ ist $\max_{0 \leq y \leq 1} f(y)$ polynomzeit-berechenbar.
- $FP = \#P \Leftrightarrow$ für jedes $f \in P{+}C^\infty$ ist $\int_0^x f(y)dy$ polynomzeit-berechenbar.

Literatur

[Bt75] R.P. Brent, The complexity of multiple precision arithmetic, *Proc. Seminar on Complexity of Computational Problem Solving*, Queensland U. Press, Brisbane, Australia (1975) 126-165

[Kn81] Knuth, Donald Ervin : The Art Of Computer Programming (Volume 2 / Seminumerical Algorithms). Second Edition. *(Addison-Wesley, Reading, 1981)*

[Ko91] K. Ko, Complexity Theory of Real Functions, *(Birkhäuser, Boston 1991)*

[Mu01] N.Th. Müller, The iRRAM: Exact Arithmetic in C++, *Computability and Complexity in Analysis - CCA2000*, Lecture notes in computer science 2064 *(Springer, Berlin, 2001)* 222-252

[We00] K. Weihrauch, Computable Analysis. An Introduction, *Springer, Berlin, 2000*

Rechnen im Bereich der reellen Zahlen

Klaus Weihrauch
FernUniversität Hagen, D-58084 Hagen
e-mail: `Klaus.Weihrauch@FernUni-Hagen.de`

Weltweit werden zahllose Computer für das Rechnen im Bereich der reellen Zahlen eingesetzt. Sie werten reelle Funktionen aus, sie bestimmen Nullstellen, Eigenwerte und Integrale, sie berechnen geometrische Figuren und sie lösen Differentialgleichungen.

Während die analytischen Aspekte des Rechnens im Reellen von den Mathematikern seit Jahrhunderten mit großem Erfolg untersucht werden, sind trotz wichtiger Fortschritte in den letzten Jahren unsere Kenntnisse über die algorithmischen Grundlagen wie Berechenbarkeit, Programmierung, Semantik, Verifikation und Komplexität im Reellen immer noch unzureichend.

Anders als für die diskrete Berechenbarkeit werden für das Rechnen im Bereich der reellen Zahlen bis heute verschiedene nicht äquivalente Modelle diskutiert und propagiert. Im Vortrag soll eines dieser Modelle vorgestellt werden, und zwar der Ansatz über Darstellungen (TTE, Type-2 Theory of Effectivity).

In der herkömmlichen Berechenbarkeitstheorie rechnen Turingmaschinen auf (endlichen) Wörtern $w \in \Sigma^*$. Wörter kann man als Namen z.B. rationaler Zahlen verwenden (Standard-Notation $\nu_{\mathbb{Q}} :\subseteq \Sigma^* \to \mathbb{Q}$). Eine Turingmaschine kann dann eine Funktion $h : \mathbb{Q} \to \mathbb{Q}$ auf den rationalen Zahlen berechnen, indem sie aus jedem Namen $w \in \Sigma^*$ einer rationalen Zahl $x \in \mathbb{Q}$ einen Namen $f_M(w)$ von $h(x) \in \mathbb{Q}$ berechnet (d.h. $\nu_{\mathbb{Q}} f_M(w) = h(x)$, falls $\nu_{\mathbb{Q}}(w) = x$). Auf entsprechende Weise kann man auf jeder abzählbaren Menge S durch Festlegen einer Notation $\nu :\subseteq \Sigma^* \to S$ ein Berechenbarkeitskonzept definieren.

In TTE wird dieses Konzept verallgemeinert, indem nun statt endlicher Wörter $w \in \Sigma^*$ unendliche Zeichenfolgen $p \in \Sigma^\omega$ als Namen dienen. Dazu werden die berechenbaren Funktionen $f :\subseteq (\Sigma^\omega)^k \to \Sigma^\omega$ auf unendlichen Folgen von Symbolen definiert, z.B. über Typ-2 Maschinen, d.h. Turingmaschinen mit unendlichen Einweg Eingabe- und Ausgabebändern. Man sieht leicht, dass jede berechenbare Funktion bzgl. der Standard-Konvergenz auf Σ^ω stetig ist. Dann werden unendliche Symbolfolgen als Namen von reellen Zahlen, von offenen Teilmengen von $\mathbb{R}^n$, von stetigen Funktionen $f : \mathbb{R} \to \mathbb{R}$ usw. verwendet.

Beispiel Dezimaldarstellung $\delta_{Dez} :\subseteq \Sigma^\omega \to \mathbb{R}, \quad \delta_{Dez}(3{\bullet}14159\ldots) = \pi$.
Für Darstellungen $\delta :\subseteq \Sigma^\omega \to M$ und $\delta' :\subseteq \Sigma^\omega \to M'$ erklärt man in naheliegender Weise, wann δ nach δ' stetig übersetzbar ($\delta \leq_t \delta'$) und berechenbar übersetzbar ($\delta \leq \delta'$) ist, und leitet daraus stetige bzw. berechenbare Äquivalenz ab. Weiter erklärt man, wann eine Funktion $h :\subseteq \Sigma^\omega \to \Sigma^\omega$ auf Namen eine Funktion $f :\subseteq M \to M'$ *realisiert*, und nennt f (δ, δ')-stetig ((δ, δ')-berechenbar), wenn f eine stetige (berechenbare) Realisierung hat.

Beispiel $x \mapsto x/3$ ist $(\delta_{Dez}, \delta_{Dez})$-berechenbar, aber $x \mapsto 3x$ ist nicht $(\delta_{Dez}, \delta_{Dez})$-stetig.

Da die Menge Σ^ω dieselbe Mächtigkeit wie die Menge $\mathbb{R}$ der reellen Zahlen hat, kann man über Darstellungen Berechenbarkeit auf allen kontinuumsmächtigen Mengen einführen.

Satz 1 *Zwei Darstellungen induzieren auf einer Menge dieselbe Stetigkeit (Berechenbarkeit), wenn sie stetig äquivalent (berechenbar äquivalent) sind.*

Nur ganz wenige der unzähligen nicht zueinander äquivalenten Darstellungen der reellen Zahlen usw. sind von Interesse. Im Vortrag werden zunächst für einige Mengen, und zwar die reellen Zahlen $\mathbb{R}$, die Menge $C(\mathbb{R})$ der stetigen Funktionen $f : \mathbb{R} \to \mathbb{R}$ und die Menge der Wahrscheinlichkeitsmaße auf dem Intervall $[0;1]$, „Standard"-Darstellungen explizit eingeführt und diskutiert. Jede dieser Darstellungen spielt bzgl. der auf der jeweiligen Menge erklärten Konvergenz eine ausgezeichnete Rolle: sie ist „zulässig" („admissible").

Es wird dann die Theorie der *zulässigen Darstellungen von Limesräumen* diskutiert [Schröder 2001]. Für zulässige Darstellungen stimmt die direkte Folgenstetigkeit einer Funktion mit der durch die Darstellungen induzierten Stetigkeit überein. Es werden zulässige Darstellungen für berechenbare metrische Räume und allgemeiner für T_0-Räume mit abzählbarer Basis eingeführt. Es werden dann Operationen erklärt, welche aus zulässigen Darstellungen neue zulässige Darstellungen erzeugen, und zwar die Einschränkung, die Konstruktion einer *initialen Darstellung* mit dem kartesischen Produkt als Spezialfall, der *inverse Limes* und die *Exponentiation*. Die Exponentiation liefert dabei eine zulässige Darstellung der Menge aller stetigen Funktionen.

Als nicht-triviales Beispiel konstruieren wir eine zulässige Darstellung der Menge der *Distributionen*. (Distributionen werden zur mathematischen Beschreibung physikalischer Vorgänge und in der Theorie der Differentialgleichungen verwendet.) Es zeigt sich, dass diese Konstruktion einen Berechenbarkeitsbegriff auf den Distributionen liefert, der genau zu dem in der mathematischen Theorie verwendeten Konvergenzbegriff passt.

Quellen:

M. Schröder, Admissible representations of limit spaces, LNCS 2064, Springer, 2001
K. Weihrauch, Computable Analysis, Springer, 2000

weitere Information: über die CCA-page

http://www.informatik.fernuni-hagen.de/cca

Tiling systems over infinite pictures and their acceptance conditions

Jan-Henrik Altenbernd, Wolfgang Thomas and Stefan Wöhrle
Lehrstuhl für Informatik VII, RWTH Aachen
e-mail: {altenbernd,thomas,woehrle}@i7.informatik.rwth-aachen.de

In the theory of automata over infinite words, many types of acceptance conditions have been studied, such as Büchi and Muller acceptance. In the framework of nondeterministic automata, three kinds of acceptance conditions have been singled out to which all other standard conditions can be reduced ([6], [5], [1]): Referring to a nondeterministic automaton $\mathcal{A}$, an ω-word α is

1. A-accepted if some complete run of $\mathcal{A}$ on α exists,
2. E-accepted if some complete run of $\mathcal{A}$ on α exists, reaching a state in a given set F of final states,
3. Büchi-accepted if some complete run of $\mathcal{A}$ on α exists reaching infinitely often a state in a given set F of final states.

It is well-known that these acceptance conditions lead to a strict hierarchy of three classes of ω-languages in the listed order.

The purpose of the present paper is to study these acceptance conditions over two-dimensional infinite words, i.e. labeled $(\omega \times \omega)$-grids or "infinite pictures". We use a model of "nondeterministic automaton" which was introduced under the name "tiling system" in [3] (see also the survey [2]). While the notion of run of a tiling system on a given infinite picture is natural, there are several versions of using the above acceptance conditions; for example one may refer to the occurrence of states on arbitrary picture positions, or one only considers the diagonal positions. As a preparatory step, we give a reduction to the latter case and thus use only the diagonal positions for visits of final states.

The first main result says that over infinite pictures we obtain the same hierarchy of languages as mentioned above for ω-languages. Whereas in the case of ω-words one can use simple state repetition arguments for the separation proofs, we need here different arguments, combining König's Lemma with certain boundedness conditions.

In the second part of the paper we show that the class of Büchi recognizable infinitary picture languages is not closed under complementation. We use a recursion theoretic result on infinitely branching infinite trees, namely that such trees with only finite branches constitute a set which is not in the Borel class Σ_1^1. From this we easily obtain a picture language that is not Σ_1^1 and thus not Büchi recognizable: One uses pictures which consist of a code of an infinitely branching finite-branch tree in the first row and otherwise contain dummy symbols. The hard part of the nonclosure result on complementation is to show the Büchi recognizability of pictures which code infinite-branch trees. For this, it is necessary to implement precise comparisons between an infinity of segments of the first row, using only finitely many states. It turns out that this is possible by using the "work space" $\omega \times \omega$.

This nonclosure proof should be compared with a corresponding result of Kaminski and Pinter [4] where a kind of Büchi acceptance is used over arbitrary acyclic graphs; in that case one has much more freedom to construct counter-examples and thus does not obtain the nonclosure result over pictures.

References

[1] J. Engelfriet and H. Hoogeboom. X-automata on ω-words. *Theoretical Computer Science*, 110:1–51, 1993.

[2] D. Giammarresi and A. Restivo. Two-dimensional languages. In *Handbook of Formal Languages*, volume III, pages 215–267. 1997.

[3] D. Giammarresi, A. Restivo, S. Seibert, and W. Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, 125(1):32–45, 1996.

[4] M. Kaminski and S. Pinter. Finite automata on directed graphs. *Journal of Computer and System Sciences*, 44:425–446, 1992.

[5] L. Staiger. Research in the theory of ω-languages. *Journal of Information Processing and Cybernetics EIK*, 23:415–439, 1987.

[6] K. Wagner. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979.

Determinization of Finite State Weighted Tree Automata

Björn Borchardt and Heiko Vogler
Dresden University of Technology
Faculty of Computer Science
D-01062 Dresden
e-mail: {borchard,vogler}@tcs.inf.tu-dresden.de

ABSTRACT

We investigate the determinization of nondeterministic bottom-up / top-down finite state weighted tree automata over some semiring A and compare the resulting four classes of formal tree series with each other. In fact, we generalize well known theorems on classes of tree languages (cf. [GS84] Chapter II, Theorems 2.6 and 2.10, Example 2.11), viz. if A is a commutative and locally finite semifield, then (i) nondeterministic bottom-up, (ii) deterministic bottom-up, and (iii) nondeterministic top-down finite state weighted tree automata are equally powerful. Moreover, if the input alphabet is not trivial, then deterministic top-down finite state weighted tree automata are strictly less powerful than the aforementioned classes.

Keywords: weighted tree automata, determinization, recognizable tree series

1. Introduction and Preliminaries

In the present paper we consider finite state weighted tree automata (for short w-fta). Such automata can be seen as a generalization of finite state tree automata and finite state weighted automata in the sense that an w-fta accepts trees and outputs costs, which are taken from a semiring A. According to the way in which a given input tree t is traversed, there are bottom-up finite state weighted tree automata (for short bu-w-fta, cf. [Sei94, DV02]), in which t is traversed by starting from its leaves and proceeding towards the root, and top-down finite state weighted tree automata (td-w-fta, cf. [Kui97]), in which t is traversed by starting from its root and proceeding towards the leaves. Orthogonal to the distinction between bottom-up and top-down finite state weighted tree automata can be nondeterministic or deterministic. These two distinction criteria give rise to four classes of tree languages:

- the class $A^{n,bu}\langle\langle T_\Sigma\rangle\rangle$ of tree series accepted by nondeterministic bu-w-fta,
- the class $A^{d,bu}\langle\langle T_\Sigma\rangle\rangle$ of tree series accepted by deterministic bu-w-fta,
- the class $A^{n,td}\langle\langle T_\Sigma\rangle\rangle$ of tree series accepted by nondeterministic td-w-fta, and
- the class $A^{d,td}\langle\langle T_\Sigma\rangle\rangle$ of tree series accepted by deterministic td-w-fta.

We prove that if the underlying semiring is a commutative and locally finite semifield, then

$$A^{d,td}\langle\langle T_\Sigma\rangle\rangle \subsetneq A^{n,bu}\langle\langle T_\Sigma\rangle\rangle = A^{d,bu}\langle\langle T_\Sigma\rangle\rangle = A^{n,td}\langle\langle T_\Sigma\rangle\rangle.$$

[0] The first author was supported by German Research Council (DFG, GRK 433/2).

Note that this is a generalization of well known theorems on tree languages (cf. e.g. [GS84] Chapter II, Theorems 2.6 and 2.10, Example 2.11). In particular, the inclusion $A^{n,bu}(T_\Sigma) \subseteq A^{d,bu}(T_\Sigma)$ is based on the determinization of bu-w-fta.

2. Finite state weighted tree automata

Definition 2.1 *A* bottom-up tree representation (over Q, Σ, and A) *is a family* $\mu = (\mu_k \mid k \geq 0)$ *of mappings*

$$\mu_k : \Sigma^{(k)} \longrightarrow A^{Q^k \times Q},$$

where Q is an arbitrary, not necessarily finite set, Σ is a ranked alphabet, and A is a semiring. μ is deterministic *if, for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and k-tuple $(q_1, \ldots, q_k) \in Q^k$, there is at most one $q \in Q$ such that $\mu_k(\sigma)_{(q_1,\ldots,q_k),q} \neq 0$.* $\diamond$

Every tree representation μ over a finite set Q of states induces a family $(\overline{\mu_k(\sigma)} \mid k \geq 0, \sigma \in \Sigma^{(k)})$ of mappings in the following way: for every $k \geq 0$ and $\sigma \in \Sigma^{(k)}$ we define a mapping

$$\overline{\mu_k(\sigma)} : A^Q \times \ldots \times A^Q \longrightarrow A^Q$$

with k arguments such that for every $V_1, \ldots, V_k \in A^Q$ and $q \in Q$,

$$\overline{\mu_k(\sigma)}(V_1, \ldots, V_k)_q = \sum_{(q_1,\ldots,q_k) \in Q^k} (V_1)_{q_1} \odot \ldots \odot (V_k)_{q_k} \odot \mu_k(\sigma)_{(q_1,\ldots,q_k),q}.$$

Note that, since we have defined $\overline{\mu_k(\sigma)}(V_1, \ldots, V_k)$ only for a tree representation over a finite set Q, we do not require A to be a complete semiring.

Observation 2.2 If Q is a finite set, then the algebraic structure $\left(A^Q, \left(\overline{\mu_k(\sigma)} \mid k \geq 0, \sigma \in \Sigma^{(k)}\right)\right)$ is a Σ-algebra. Its unique homomorphism $h_\mu : T_\Sigma \to A^Q$ is defined for every $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots t_k \in T_\Sigma$ by

$$h_\mu(\sigma(t_1, \ldots t_k)) = \overline{\mu_k(\sigma)}(h_\mu(t_1), \ldots, h_\mu(t_k)). \quad \diamond$$

Because of later convenience we first define a tree automaton with an arbitrary, not necessarily finite set Q of states.

Definition 2.3 *A* bottom-up weighted tree automaton over A (for short bu-w-ta) *is a 5-tuple $M = (Q, \Sigma, Q_d, A, \mu)$, where Q is a set, Σ is a ranked alphabet, $Q_d \subseteq Q$, A is a semiring, and μ is a bottom-up tree representation. A bu-w-ta is termed* deterministic *if its tree representation is deterministic.*

The bu-w-ta $M = (Q, \Sigma, Q_d, A, \mu)$ is a bottom-up finite state weighted tree automaton over A (for short bu-w-fta) *if Q is finite. In this case the tree series $S_M \in A\langle\langle T_\Sigma \rangle\rangle$ accepted by M is defined for every $t \in T_\Sigma$ by*

$$(S_M, t) = \sum_{q \in Q_d} h_\mu(t)_q.$$

We also call S_M the semantics of M. Moreover, two bu-w-fta M and N are called equivalent *if $S_M = S_N$.* $\diamond$

Note that we have defined the semantics only in the finite case. Hence again no completeness property is required.

The concept of td-w-fta is very similar. The one and only difference is the form of the transition matrix which is $\mu_k : \Sigma^{(k)} \to A^{Q \times Q^k}$.

3. Determinization

In this section we compare the classes $A^{n,bu}\langle\langle T_\Sigma\rangle\rangle$ and $A^{d,bu}\langle\langle T_\Sigma\rangle\rangle$ for classes of semifields. More precisely, for a given nondeterministic bu-w-fta M we construct a deterministic bu-w-ta M' which, if it is finite, is equivalent to M. To do so we extend the power set construction known from the theory of finite state weighted automata (cf. [BGW00, Moh97]). Throughout this section let $M = (Q, \Sigma, Q_d, A, \mu)$ denote an arbitrary bu-w-fta over an arbitrary semifield A, if not stated otherwise.

In order to define the determinizaton of M we have to define a binary relation ρ_M on T_Σ as well as a tree series $S_M \in A\langle\langle T_\Sigma\rangle\rangle$. For every $t \in T_\Sigma$ let

$$\rho_M := \{(s,t) \in T_\Sigma \times T_\Sigma \mid h_\mu(s) = h_\mu(t)\}$$

$$(S'_M, t) := \begin{cases} (S_M, t) & \text{if } t \in supp(S_M), \\ 1 & \text{otherwise.} \end{cases}$$

Now we can define the determinization of M. Note that the constructed automaton might have an infinite set of states, i.e. we generate only a bu-w-ta (cf. Definition 2.3). Later on we will give two conditions, which ensure the finiteness of the set of states of $det(M)$. This implies that $det(M)$ is a finite automaton.

Definition 3.1 *The* determinization of M *is the bu-w-ta* $det(M) = (Q', \Sigma, Q'_d, A, \mu')$ *given by*

$$Q' = \{[t]_\rho \mid t \in T_\Sigma, h_\mu(t) \neq \underline{0}\}, \qquad Q'_d = \{[t]_\rho \mid t \in supp(S_M)\},$$

and for every $k \geq 0$, $q'_1, \ldots, q'_k, q' \in Q'$, *and* $\sigma \in \Sigma^{(k)}$,

$$\mu'_k(\sigma)_{(q'_1,\ldots,q'_k),q'} = \begin{cases} (S_M^{(\varphi)}, t_k)^{-1} \odot \cdots \odot (S_M^{(\varphi)}, t_1)^{-1} \odot (S_M^{(\varphi)}, \sigma(t_1,\ldots,t_k)) \\ \quad \textit{if } \exists t_1,\ldots,t_k \in T_\Sigma, q'_1 = [t_1]_\rho, \ldots, q'_k = [t_k]_\rho, q' = [\sigma(t_1,\ldots,t_k)]_\rho \\ 0 \quad \textit{otherwise} \end{cases}$$

◇

Theorem 3.2 *If the set of states of $det(M)$ is finite, then $det(M)$ is a well defined, finite, and deterministic bu-w-fta being equivalent to M.* □

Corollary 3.3 *If $\{h_\mu(t) \subseteq A^Q \mid t \in T_\Sigma\}$ is a finite set, then $det(M)$ is a finite and deterministic bu-w-fta being equivalent to M.* □

We now present three phenomena of our determinization.

(i) There is a bu-w-fta such that there does not exist an equivalent deterministic automaton.

(ii) There is a bu-w-fta M such that there exists an equivalent deterministic bu-w-fta, but $det(M)$ is not a finite automaton.

(iii) There is a determinstic bu-w-fta M such that $det(M)$ is not a finite automaton.

Note that (iii) can be avoided by considering bu-w-fta over a commutative semifield and by choosing slightly more sophisticated ρ_M and S'_M.

Now let us turn to the second condition, which ensures that $det(M)$ is a finite automaton.

Theorem 3.4 *If A is a locally finite semifield, then $det(M)$ is a finite and deterministic bu-w-fta being equivalent to M.* □

4. Comparison of classes of recognizable tree series

Theorem 4.1 *Let A be a commutative and locally finite semifield and Σ a ranked alphabet. Then*

$$A^{n,bu}\langle\langle T_\Sigma\rangle\rangle = A^{d,bu}\langle\langle T_\Sigma\rangle\rangle = A^{n,td}\langle\langle T_\Sigma\rangle\rangle.$$

Furthermore, if Σ satisfies $card\left(\Sigma^{(0)}\right) \geq 2$ and $\bigcup_{k\geq 2} \Sigma^{(k)} \neq \emptyset$, then

$$A^{d,td}\langle\langle T_\Sigma\rangle\rangle \subsetneq A^{n,td}\langle\langle T_\Sigma\rangle\rangle. \qquad \square$$

Note that the Boolean semiring is a commutative and locally finite semifield. Moreover, one can prove that there is a one-to-one correspondence between bu-w-fta over the Boolean semiring and bottom-up finite state tree automata. Hence we have obtained a generalization of the well known theorem on tree languages (cf. [GS84] Chapter II, Theorems 2.6 and 2.10, Example 2.11).

References

[BGW00] A.L. Buchsbaum, R. Giancarlo, and J.R. Wetbrook. On the determinization of weighted finite automata. *SIAM Journal of Computing*, to appear, 2000.

[BR88] J. Berstel and Ch. Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS-Monographs*. Springer Verlag, 1988.

[Don65] J.E. Doner. Decidability of the weak second-order theory of two successors. *Notices Amer. Math. Soc.*, 12:Abstract No. 65T 648, 819, 1965.

[DV02] M. Droste and H. Vogler. A Kleene theorem for weighted tree automata. Technical Report TUD-FI-02-04, Dresden University of Technology, Faculty of Computer Science, June 2002.

[EFV01] J. Engelfriet, Z. Fülöp, and H. Vogler. Bottom-up and top-down tree series transformations. *J. Automata, Languages and Combinatorics*, 2001. accepted.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines, Vol.A.* Academic Press, 1974.

[Eng75] J. Engelfriet. Bottom–up and top–down tree transformations — a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[GS84] F. Gécseg and M. Steinby. *Tree Automata.* Akadémiai Kiadó, Budapest, 1984.

[GS97] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer-Verlag, 1997.

[KS86] W. Kuich and A. Salomaa. *Semirings, Automata, Languages.* EATCS Monographs on Theoretical Computer Science, Springer Verlag, 1986.

[Kui97] W. Kuich. Formal power series over trees. In *Proc. of the 3rd International Conference Developments in Language Theory*, pages 60–101. Aristotle University of Thessaloniki, 1997.

[Moh97] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311, 1997.

[Sei94] H. Seidl. Finite tree automata with cost functions. *Theoret. Comput. Sci.*, 126:113–142, 1994.

[SS78] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series.* Texts and Monographs in Computer Science, Springer Verlag, 1978.

[TW68] J.W. Thatcher and J.B. Wright. Generalized finite automata theory with application to a decision problem of second-order logic. *Math. Systems Theory*, 2(1):57–81, 1968.

P-Systeme mit Energiekontrolle

Rudolf Freund
Institut für Computersprachen, Technische Universität Wien
Favoritenstraße 9, A-1040 Wien, Österreich
e-mail: `rudi@emcc.at`

KURZFASSUNG

P-Systeme mit Energiekontrolle sind Membransysteme, bei denen die Evolutionsregeln Energie erzeugen oder verbrauchen, dabei aber in Summe innerhalb eines bestimmten Bereichs bleiben. P-Systeme mit Energiekontrolle mit nur einer einzigen Membran und einem Energiebereich von $\{0, 1\}$ für die Summe der Energien können ohne Verwendung weiterer für P-Systeme sonst üblicher Ingredienzien (Katalysatoren, Prioritäten etc.) jede beliebige rekursiv aufzählbare Menge von Vektoren natürlicher Zahlen erzeugen; mit dem minimalen Energiebereich von $\{0\}$ können zumindest die von Matrixgrammatiken (ohne Vorkommenstest) erzeugbaren Mengen generiert werden.

Schlüsselwörter: Matrixgrammatiken, P-Systeme, Universalität

Membransysteme wurden im Jahre 1998 (s. [5]) durch Gheorghe Păun eingeführt (und deshalb danach auch P-Systeme genannt); sie stellen eine Klasse verteilter paralleler Berechnungsmodelle dar, die von biologischen Vorgängen in der Natur inspiriert sind. Die Membranstruktur selbst und besondere Eigenschaften der Membranen, insbesondere für den Transport von Objekten, sind die wichtigsten Charakteristika, welche in den verschiedensten Modellen von P-Systemen untersucht werden (z.B., s. [1], [2], [6], [7]). Eine Membranstruktur besteht aus Membranen, welche hierarchisch in die äußerste Hautmembran eingebettet sind; jede Membran umschließt eine Region, welche auch andere Membranen enthalten kann. In einem Kompartment k (jenem Teil, welcher durch eine Membran k und durch ihre inneren Membranen begrenzt wird) entwickeln sich Multimengen von Objekten auf Grund sogenannter Evolutionsregeln. Eine Berechnung in einem P-System besteht aus einer endlichen Folge von aus der Startkonfiguration durch maximal parallele Anwendung der Evolutionsregeln ableitbaren Konfigurationen. Die Anzahl terminaler Objekte in (einer bestimmten Membran einer) haltenden Berechnung (auf die Endkonfiguration kann keine Evolutionsregel mehr angewendet werden) stellt das Ergebnis einer Berechnung in einem P-System dar. Für viele Varianten von P-Systemen konnte gezeigt werden, dass sie die rekursiv aufzählbaren Mengen von Vektoren natürlicher Zahlen bzw. die rekursiv aufzählbaren Sprachen charakterisieren (für eine jeweils aktuelle Literaturübersicht s. [4]).

In [8] wurde erstmals die Energiebilanz der biologischen Prozesse in einer Zelle als Grundlage für eine Variante von P-Systemen betrachtet: Die von allen in einem Ableitungsschritt verwendeten Regeln erzeugte Energie wird aufsummiert; nur wenn diese positiv ist, kann diese Multimenge von Regeln angewendet werden, allerdings nur, wenn diese Multimenge auch eine maximale mit dieser Eigenschaft ist. Überschreitet die über mehrere Schritte akkumulierte Energie in einem Kompartment einen Grenzwert, so führt dies zur Auflösung der umgebenden Membran.

In den nun hier betrachteten P-Systemen mit Energiekontrolle werden in jedem einzelnen Kompartment die für eine Multimenge von Regeln gebrauchten Energien aufsummiert; ist die

Gesamtenergie in einem Kompartment innerhalb eines vorgegebenen Bereiches und ist diese Multimenge von Regeln auch maximal bezüglich dieser Eigenschaft, so kann sie auf die Objekte in diesem Kompartment angewendet werden.

Bereits mit einer minimalen Membranstruktur (also mit nur einer Membran) und einem Energiebereich von $\{0, 1\}$ können P-Systeme mit Energiekontrolle jede beliebige rekursiv aufzählbare Menge von Vektoren natürlicher Zahlen erzeugen; mit dem minimalen Energiebereich von $\{0\}$ können zumindest die von Matrixgrammatiken (ohne Vorkommenstest) erzeugbaren Mengen generiert werden. Bei Verwendung kontextfreier Produktionen in P-Systemen mit Energiekontrolle können überdies ebenfalls mit nur einer Membran und einem Energiebereich von $\{0, 1\}$ alle rekursiv aufzählbaren Sprachen generiert werden.

Eine ausführliche Darstellung der oben beschriebenen P-Systeme mit Energiekontrolle und vollständige Beweise der oben angeführten Resultate sind in [3] zu finden.

Literatur

[1] C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London (2000) (Chapter 3: "Computing with Membranes").

[2] J. Dassow, Gh. Păun, On the power of membrane computing, *J. of Universal Computer Sci.* **5**, 2 (1999), 33-49 (www.iicm.edu/jucs).

[3] R. Freund, Energy-controlled P systems, *Proceedings WMC 2002* (2002).

[4] The P Systems Web Page: http://dna.bio.disco.unimib.it/psystems/

[5] Gh. Păun, Computing with membranes, *Turku Center for Computer Science-TUCS Report* No 208 (1998) (s. www.tucs.fi).

[6] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS* **67** (1999), 139–152.

[7] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae* **41**, 3 (2000), 259-266, and *Turku Center for Computer Science-TUCS Report* No 218 (1998) (s. www.tucs.fi).

[8] Gh. Păun, Y. Suzuki, H. Tanaka, P Systems with energy accounting, *Int. J. Computer Math.* **78**, 3 (2001), 343–364.

On Deterministic Finite Automata and Syntactic Monoid Size

Markus Holzer and Barbara König
Institut für Informatik, Technische Universität München,
Arcisstraße 21, D-80290 München, Germany
e-mail: {holzer,koenigb}@informatik.tu-muenchen.de

Regular languages and their implementations have received more and more attention in recent years due to the many new applications of finite automata and regular expressions in object-oriented modeling, programming languages and other practical areas of computer science. In recent years, quite a few software systems for manipulating formal language objects, with an emphasis on regular-language objects, have been developed. These applications and implementations of regular languages motivate the study of descriptive complexity of regular languages. A very well accepted and studied measure of descriptional complexity for regular languages is the size, i.e., number of states, of deterministic finite automata.

Besides machine oriented characterization of regular languages, they also obey several algebraic characterizations. It is a consequence of Kleene's theorem [1], that a language $L \subseteq \Sigma^*$ is regular if and only if there exists a finite monoid M, a morphism $\varphi : \Sigma^* \to M$, and a finite subset $N \subseteq M$ such that $L = \varphi^{-1}(N)$. The monoid M is said to recognize L. The syntactic monoid of L is the smallest monoid recognizing the language under consideration. It is uniquely defined up to isomorphism and is induced by the syntactic congruence $\sim_L$ defined over Σ^* by $v_1 \sim_L v_2$ if and only if for every $u, w \in \Sigma^*$ we have $uv_1w \in L \iff uv_2w \in L$. The syntactic monoid of L is the quotient monoid $M(L) = \Sigma^*/\sim_L$. It is well known, that the syntactic monoid $M(L)$ is isomorphic to the transition monoid of the minimal deterministic finite automaton accepting L (see, e.g., Pin [4]). In this paper we propose the size of the syntactic monoid as a natural measure of descriptive complexity for regular languages and study the relationship between automata and monoid size in more detail.

In most cases, we show tight upper bounds on the syntactic monoid, proving that there are languages accepted by n-state deterministic finite automata whose syntactic monoid has a certain size. It is easy to see that for unary regular languages the size is linear, while that for regular languages over an input alphabet with at least three letters is maximal, i.e., n^n. The challenging part is to determine the size of the syntactic monoid for regular languages over a binary alphabet. The trivial lower and upper bounds are $n!$—induced by the two generators of S_n—and $n^n - n! + g(n)$, respectively, where $g(n)$ denotes Landau's function [2], which equals the maximal order of all permutations in S_n. With a more sophisticated argument, we are able to design a semigroup $U_{k,\ell}$ on two generators, which is a syntactic monoid, and prove the following result:

Theorem *Assume $n \geq 3$ and let A be a n-state deterministic finite automata. Then a monoid of size $n^n - n! + g(n)$ is sufficient to recognize the language $L(A)$ and*

$$n^n - \binom{n}{\ell}\ell! n^k - \binom{n}{\ell} k^k \ell^\ell,$$

where $n = k + \ell$, $\ell - k \leq 4$, and $\gcd\{k, \ell\} = 1$ for some natural numbers k and ℓ, are necessary in the worst case.

Compared to the trivial lower bound, where $\lim_{n\to\infty} \frac{n!}{n^n} = 0$, the lower bound on the syntactic monoid size given above is much better, since the fraction of $n^n - \binom{n}{\ell}\ell! n^k - \binom{n}{\ell} k^k \ell^\ell$ (for appropriate k and ℓ) and n^n tends to 1 as n goes to infinity. Thus, we obtain the following corollary:

Corollary *There is a sequence $L_1, L_2, \ldots$ of binary regular languages such that*

$$\lim_{n\to\infty} \frac{M(L_i)}{n^n} = 1,$$

and each L_i is accepted by a minimal deterministic finite automaton with exactly n states.

We summarize some computed values on the size of some semigroups (monoids) in Table 1. There, the number $max(n)$ denotes the size of the maximal transformation semigroup (monoid)

n	$\|S_n\| = n!$	$\|U_{k,\ell}\|$			$max(n)$	$n^n - n! + g(n)$	$\|T_n\| = n^n$
		ℓ	k				
3	6	2	1	13	24	24	27
4	24	3	1	133	176	236	256
5	120	3	2	1857	2110	3011	3125
		4	1	1753			
6	720	5	1	27311	32262 (?)	45942	46656
7	5040	4	3	607285	610871 (?)	818515	823543
		5	2	610871			
		6	1	492637			
8	40320	5	3	13492007	13492007 (?)	16736911	16777216
		7	1	10153599			
9	362880	5	4	323534045	323534045 (?)	387057629	387420489
		7	2	306605039			
		8	1	236102993			
10	3628800	7	3	8678434171	8678434171 (?)	9996371230	10000000000
		9	1	6122529199			
11	39916800	6	5	256163207631	258206892349 (?)	285271753841	285311670611
		7	4	258206892349			
		8	3	251856907425			
		9	2	231326367879			
		10	1	175275382621			

Table 1: Sizes of some investigated semigroups.

with two generators, which might not coincide with the size of some $U_{k,\ell}$. A table entry with a question mark indicates our that the precise value is not known and thus is a conjecture. The generators for the groups with 24, 176, 2110, and 32262 elements all contain a single cycle permutation $(1\,2\,\ldots\,n)$. However, already for $n = 7$, the case where one of the generators is the cycle is beat by our semigroup.

It remains to tighten the bound on the syntactic monoid size on two generators in future research. To understand the very nature of this question it seems to be very important, to precisely characterize the maximal size transformation semigroup on two generators, in a similar way as the generators for T_n and S_n are characterized in the theorems of Salomaa [5] and Piccard [3]. We conjecture, that for every $n \geq 7$, there exists natural numbers k and ℓ with $n = k + \ell$ such that the semigroup $U_{k,\ell}$ is maximal under all two generator transformation semigroups (monoids).

References

[1] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata studies*, volume 34 of *Annals of mathematics studies*, pages 2–42. Princeton University Press, 1956.

[2] E. Landau. Über die Maximalordnung der Permutationen gegebenen Grades. *Archiv der Mathematik und Physik*, 3:92–103, 1903.

[3] S. Piccard. *Sur les bases du groupe symétrique et les couples de substitutions qui engendrent un groupe régulier*. Librairie Vuibert, Paris, 1946.

[4] J.-E. Pin. *Varieties of formal languages*. North Oxford, 1986.

[5] A. Salomaa. On the composition of functions of several variables ranging over a finite set. *Annales Universitatis Turkuensis*, 41, 1960. Series AI.

Verallgemeinerte kontextfreie Grammatiken

Andreas Klein
Institut für Mathematik und Informatik
Universität Kassel
e-mail: `klein@mathematik.uni-kassel.de`

und

Martin Kutrib
Institut für Informatik
Universität Giessen
e-mail: `kutrib@informatik.uni-giessen.de`

KURZFASSUNG

Kontextfreie Grammatiken spielen in vielen Anwendungen eine wichtige Rolle. In diesem Vortrag stellen wir eine Verallgemeinerung kontextfreier Grammatiken vor, die auf gekoppelten Nichtterminalsymbolen basiert.

Schlüsselwörter: kontextfreie Grammatiken, Pumping Lemma

Kontextfreie Sprachen haben viele angenehme Eigenschaften, es existieren zum Beispiel effiziente Parsing-Algorithmen. Allerdings gibt es viele Sprachen, die nicht kontextfrei sind. Es liegt daher nahe, kontextfreie Grammatiken zu verallgemeinern, so daß man eine größere Klasse von Sprachen erhält, ohne daß die guten Eigenschaften kontextfreier Sprachen verloren gehen.

Wir führen zu diesem Zweck *gekoppelte* Nichtterminalsymbole ein. Untereinander gekoppelte Nichtterminalsymbole dürfen immer nur gemeinsam ersetzt werden. (Die auf diese Weise definierten Grammatiken ähneln indizierten Grammatiken, haben aber andere Eigenschaften.)

Beispiel Die Grammatik mit den Regeln

$$S \to AA$$

$$A\ A \to ab\ cd$$

$$A\ A \to aAb\ cAd$$

erzeugt die Menge $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$.

Eine typische Ableitung in dieser Grammatik ist:

$$S \to AA \to aAbcAd \to aaAbbccAdd \to aaabbbcccddd$$

Wir nennen Grammatiken, in denen höchstens k Nichtterminalsymbole miteinander gekoppelt sind, eine verallgemeinerte Grammatik vom Typ $2k$. (Man kann als Verallgemeinerung der regulären Sprachen auch Grammatiken mit ungeradem Grad definieren. Dies soll jedoch im Folgenden

nicht weiter betrachtet werden.)

Die Analogie zu kontextfreien Grammatiken zeigt sich in der Verallgemeinerung des Pumping Lemmas:

Lemma (schwaches Pumping Lemma) *Sei G eine verallgemeinerte kontextfreie Grammatik vom Grad k. Dann existiert eine Konstante $n \in \mathbb{N}$, so daß jedes Wort $w \in L(G)$ mit $|w| > n$ sich als $x_0y_1x_1y_2 \dots y_nx_n$ schreiben läßt, wobei $1 \leq |y_1y_2 \dots y_n| \leq n$ gilt und für jedes $i \in \mathbb{N}_0$ existiert ein Wort $w_i \in L(G)$, das in irgendeiner Reihenfolge aus den Teilwörtern $x_0, \dots x_n$ und i mal y_j für $1 \leq j \leq k$ zusammengesetzt ist.*

Im Gegensatz zu dem normalen Pumping Lemma für kontextfreie und reguläre Sprachen wird in dieser schwachen Version des Pumping Lemmas keine Aussage über die Reihenfolge der Teilwörter getroffen. Trotzdem reicht die schwache Version des Pumping Lemmas aus, um die meisten Aussagen über kontextfreie Grammatiken auf verallgemeinerte kontextfreie Grammatiken zu übertragen. Z.B. erhalten wir:

Theorem *Jede verallgemeinerte kontextfreie Sprache über einem unären Alphabet ist regulär.*

Für kleine Grade können wir das schwache Pumping Lemma zu dem starken Pumping Lemma verbessern.

Lemma (starkes Pumping Lemma) *Für $k \leq 6$ gilt:*

Für jede verallgemeinerte kontextfreie Grammatik G existiert eine Konstante $n \in \mathbb{N}$, so daß für alle Wörter $w \in L(G)$ mit $|w| > n$ das Wort w in $x_0x_1 \dots x_k$ zerlegt werden kann. Außerdem existieren $y_1 \dots y_k$ mit $1 \leq |y_1 \dots y_k|$, so daß für alle $i \in \mathbb{N}_0$ das Wort $w_i = x_0y_1^ix_1y_2^ix_2 \dots y_k^ix_k$ in $L(G)$ liegt.

Es bleibt eine offene Frage, ob das starke Pumping Lemma auch für beliebige Grade gilt.

Self-Assembling Finite Automata

Andreas Klein
Institut für Mathematik und Informatik
Universität Kassel
e-mail: `klein@mathematik.uni-kassel.de`

and

Martin Kutrib
Institut für Informatik
Universität Giessen
e-mail: `kutrib@informatik.uni-giessen.de`

ABSTRACT

We investigate a model of self-assembling finite automata. An automaton is assembled on demand during its computation from copies out of a finite set of items. The items are pieces of a finite automaton which are connected to the already existing automaton by overlaying states. Depending on the allowed number of such interface states, the degree, infinite hierarchies of properly included language families are shown. The presented model is a natural and unified generalization of regular and context-free languages since degrees one and two are characterizing the finite and pushdown automata, respectively. Moreover, by means of different closure properties nondeterministic and deterministic language families are separated.

Introduction

Self-assembly appears in nature in several ways. One of the simplest mechanisms is the merging of drops of water when placed close together. The process is directed by minimization of potential energy and, thus, an example for uncoded self-assembly. On the other extreme in complexity protein molecules inside biological cells self-assemble to reproduce cells each time they divide. In this example the assembly instructions are built in the components and, therefore, it is coded self-assembly [4]. Originally, the study of self-assembly was motivated by biologists. A well-studied example is the assembly of bacteriophages, a type of virus which infects bacterial cells [1]. Formal investigations in this field are accompanied by the development of corresponding computational models which are also of great interest from an engineering point of view. An introduction can be found in [8] where an automaton model of self-assembling systems is presented.

Here, in some sense, we adapt self-assembly to the theory of automata and formal languages. Basically, the idea is to assemble an automaton during its computation. Therefore, we provide a finite set of items, the so-called *modules*. The automata are assembled from module copies on demand. The assembling rules are encoded by the state transition function. Starting with one piece of a finite automaton during the computation so-called *assembling transitions* are traversed that direct the assembling of another copy of some item in a well-specified manner.

Each of the modules has a set of entry and a set of return states which together are called *interface states*. An assembling transition specifies how the new copy of the module fits to the already existing part of the automaton. The connection is made by overlaying the interface states by existing states. So the result of the self-assembly is a finite automaton, but the number of its states may depend on the input. It will turn out that the generative capacity of such models depend on their degree, i.e. the number of interface states of the modules.

Related to the work of the present paper are the so-called self-modifying finite automata [5, 6, 7, 9]. In this model modifications of the automaton are allowed during transitions. The modifications include adding and deleting states and transitions. A weak form of self-modifying automata has been shown to accept the metalinear languages as well as some other families of context-free and non-context-free languages. Less restricted variants can accept arbitrarily hard languages, even non-recursive ones.

Since assembling modules sounds like calling subroutines another related paper is [3], where finite automata are considered that have a stack for storing return addresses (states). Every time a final state is entered the computation continues in the state at the top of the stack. Depending on the number of states which may be stored during one transition an infinite hierarchy in between the regular and context-free languages is shown.

Here by means of self-assembling finite automata of degree k we obtain a natural and unified generalization of finite automata and pushdown automata. In particular, infinite hierarchies depending on the degree are shown. For degree one and two the regular and context-free languages are characterized, respectively. Moreover, some closure properties are proved which lead to a separation result between nondeterministic and deterministic computations.

References

[1] Casjens, S. and King, J. *Virus assembly*. Annual Review of Biochemistry 44 (1975), 555–604.

[2] Klein, A. and Kutrib, M. *Self-assembling finite automata*. IFIG Research Report 0201, Institute of Informatics, University of Giessen, 2002.
`http://www.informatik.uni-giessen.de/staff/kutrib/papers.html`

[3] Nebel, M. E. *On the power of subroutines for finite state machines*. J. Aut., Lang. and Comb. 6 (2001), 51–74.

[4] Penrose, L. S. *Self-reproducing machines*. Scientific American 200 (1959), 105–113.

[5] Rubinstein, R. S. and Shutt, J. N. *Self-modifying finite automata*. Proc. of the IFIP 13th World Computer Congress. Vol. 1 : Technology and Foundations, 1994, pp. 493–498.

[6] Rubinstein, R. S. and Shutt, J. N. *Self-modifying finite automata – power and limitations*. Technical Report WPI-CS-TR-95-4, Computer Science Department, Worcester Polytechnic Institute, 1995.

[7] Rubinstein, R. S. and Shutt, J. N. *Self-modifying finite automata: An introduction*. Inform. Process. Lett. 56 (1995), 185–190.

[8] Saitou, K. and Jakiela, M. J. *On classes of one-dimensional self-assembling automata*. Complex Systems 10 (1996), 391–416.

[9] Wang, Y., Inoue, K., Ito, A., and Okazaki, T. *A note on self-modifying finite automata*. Inform. Process. Lett. 72 (1999), 19–24.

Berechnung von Implikationen

Roman König
Lehrstuhl für Informatik II, Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
e-mail: `Roman.Koenig@informatik.uni-erlangen.de`

Ein (formaler) Kontext $K = (G, M, I)$ ist eine Relation I zwischen einer Menge G von Gegenständen und einer Menge M von Merkmalen. Für Teilmengen U und V von M sagt man, dass die Implikation $U \longrightarrow V$ im Kontext K erfüllt ist, wenn jeder Gegenstand von K, der alle Merkmale aus U besitzt, auch alle Merkmale aus V besitzt. Das Auffinden von Implikationen für einen gegebenen Kontext ist eine zentrale Aufgabe für viele Anwendungen. Durch induktive Definition eines geeigneten Kontextes gelingt es, einen einfachen Algorithmus anzugeben, der alle Implikationen von K auffindet.

Mustererzeugung mit Zellularautomaten

Jan-Thomas Löwe
Institut für Informatik
Universität Giessen
Arndtstr. 2, D-35392 Giessen
e-mail: Jan-Thomas.Loewe@informatik.uni-giessen.de

Zur Speicherung von Bildern in Computersystemen werden diese normalerweise durch eine Sequenz von Helligkeitswerten angegeben. Die Ausgabe auf einem Bildschirm erfolgt Punkt- und Zeilenweise. Culik und Kari [CulikKari:1993, CulikKari:1994] verwenden (gewichtete) endliche Automaten zur Beschreibung und Kompression von Mustern. Betrachtet man einen LCD-Flachbildschirm, so kann man die einzelnen Pixel als Zellen eines Zellularautomaten auffassen. Damit lassen sich Muster ebenfalls in Form von Überführungsregeln des Automaten beschreiben.

In diesem Vortrag zeigen wir verschiedene Formen der Mustererzeugung in Zellularautomaten. Während bei Culik und Kari die Rekonstruktion eines Musters Koordinaten-Eingaben von aussen benötig, kann der Zellularautomat diese Informationen selbst erzeugen. Ein interessanter Aspekt ist die Erzeugung von Mustern als Raum-Zeit-Diagramm des Automaten. Dabei berechnet der Automat aus einer vorgegebenen Zeile die jeweils nachfolgende Musterzeile (im zweidimensionalen) oder eine Musterfläche (im mehrdimensionalen). D.h. das Muster wird aus lokalen Informationen und Berechnung konstruiert.

Wir betrachten sowohl *direkte* als auch *indirekte* Zustandsinterpretation, d.h. die Fälle in denen der Zustand direkt als Farbwert aufgefasst werden kann bzw. mehr Zustände pro Farbwert existieren. Bei direkter Zustandsinterpretation sind nicht alle Muster erzeugbar. In manchen Fällen lassen sich minimale „Fehler“ einbauen, die nur geringfügig sichtbar sind, siehe z.B. Abbildung 2.

Zu einem gegebenen Muster konstruieren wir die zugehörige Überführungsfunktion eines Zellularautomaten. Durch wiederholte Anwendung auf verschiedene Muster erhalten wir eine Überführungsfunktion, die alle diese Muster erzeugen kann. Bei entsprechender Ähnlichkeit der Muster zueinander, wächst die Zustands- und Überführungszahl langsamer als die Größe der Musterpunkte. Für Videosequenzen ist diese Methode zur Kompression der Informationen geeignet einsetzbar.

Weitere Anwendungen bestehen in der Erkennung von Veränderung innerhalb von Mustern oder Korrektur von auftretenden Fehler bzw. Störungen mit Hilfe von bereits bekannten Informationen, siehe z.B. Abbildung 3.

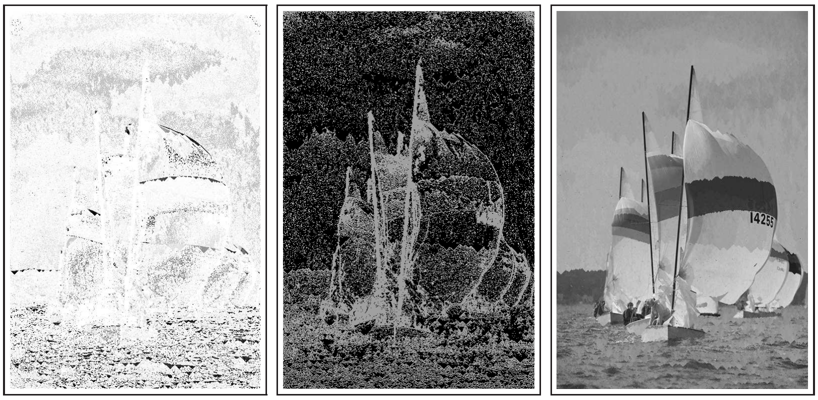

Abbildung 2: Abweichungen und eingebaute Fehler im Muster

Abbildung 3: Korrektur von Fehlern

Literatur

[1] K. Culik II, J. Kari *Image Compression Using Weighted Finite Automata.* Comput. Graphics 17 (1993), 305-313.

[2] K. Culik II, J. Kari *Inference Algorithms for WFA and Image Compression.* In Fractal Image Compression, ed. Y. Fisher, Springer, 1994, 245-262.

Berechnung der Hausdorff-Dimension regulärer ω-Sprachen

René Mazala
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1, 06099 Halle
e-mail: mazala@informatik.uni-halle.de

Die Hausdorff-Dimension ist ein vielseitig eingesetzter Komplexitätsbegriff in topologischen Räumen. Im Bereich der ω-Sprachen über einem r-elementigen Alphabet X verwendet man meistens die Standard-Topologie, die von den offenen Kugeln $\{w \cdot X^\omega \;:\; w \in X^*\}$ erzeugt wird. Für $\alpha \in \mathbb{R},\ \alpha \geq 0$ wird das Hausdorff-Maß $L_\alpha(F)$ einer ω-Sprache $F \subseteq X^\omega$ wie folgt erklärt:

$$L_\alpha(F) := \lim_{n\to\infty} \left(\inf \left\{ \sum_{v \in V} r^{-\alpha\cdot|v|} \;:\; F \subseteq V \cdot X^\omega \;\wedge\; \inf\{|v| \;:\; v \in V\} \geq n \right\} \right)$$

Dann definiert man die Hausdorff-Dimension $\dim(F)$ als

$$\dim(F) := \inf\{\alpha \geq 0 \;:\; L_\alpha(F) = 0\}$$

Die effektive Berechnung der Hausdorff-Dimension stellt allerdings eine nicht sehr leicht zu lösende Aufgabe dar. Wir wollen hier mit Hilfe von unendlichen Spielen die Hausdorff-Dimension regulärer ω-Sprachen berechnen. Der erste Spieler setzt fortlaufend Anteile seines gegenwärtigen Kapitals auf die Buchstaben des Alphabets, während der zweite Spieler nach jedem Setzen einen Buchstaben auswählt und das Kapital je nach Einsatz neu berechnet. Der zweite Spieler muss dabei nur beachten, dass das entstehende ω-Wort aus der vorgegebenen Sprache F ist. Gesucht ist dann eine Strategie für den ersten Spieler, bei der auf allen Wörtern aus F der Gewinn möglichst schnell wächst.

Es ist bekannt, dass der Gewinnexponent einer Strategie durch $1 - \dim(F)$ beschränkt ist. Wir zeigen nun, dass man diese obere Schranke durch Approximation erhalten kann. Die von uns verwendeten Strategien lassen sich von endlichen Automaten realisieren, setzen rationale Anteile und folgen einer gewissen Gleichverteilung auf den Präfixen der ω-Sprache F.

Wir führen unsere Untersuchungen zunächst an abgeschlossenen regulären ω-Sprachen F durch, deren Automat eine starke Zusammenhangskomponente ist. Anschließend benutzen wir bekannte Darstellungsformen regulärer ω-Sprachen, um auch allgemeine Resultate zu erhalten.

Verallgemeinerte P-Systeme mit verbotenem Kontext

Marion Oswald
Institut für Computersprachen, Technische Universität Wien
Favoritenstraße 9, A-1040 Wien, Österreich
e-mail: marion@emcc.at

KURZFASSUNG

Wir betrachten erweiterte Varianten von verallgemeinerten P-Systemen, d.h. Membransystemen, in welchen die Evolutionsregeln sequentiell angewendet werden. Wir untersuchen hauptsächlich Bedingungen der Anwendbarkeit auf einzelne Objekte sowie auf den restlichen Inhalt des zugrundeliegenden Kompartments. Wir zeigen, dass man schon für eine sehr eingeschränkte Variante (verallgemeinerte P-Systeme mit verbotenem Kontext) die Mächtigkeit universeller Berechenbarkeit erreichen kann.

Schlüsselwörter: Membranstruktur, P-Systeme, Registermaschinen

Seit der Einführung von Membransystemen durch Gheorghe Păun (s. [6]) im Jahre 1998 wurden bereits viele Varianten, welche durch Merkmale biologischer Systeme inspiriert sind, untersucht. Die Membranstruktur selbst und besondere Eigenschaften der Membranen, insbesondere für den Transport von Objekten, sind die wichtigsten Charakteristika, welche in den verschiedensten Modellen von P-Systemen untersucht werden (z.B., s. [1], [7]).

Eine Membranstruktur besteht aus Membranen, welche hierarchisch in die äußerste Hautmembran eingebettet sind; jede Membran umschließt eine Region, welche auch andere Membranen enthalten kann; dabei bezeichnet Komparment k den Teil, welcher durch eine Membran k und durch ihre inneren Membranen begrenzt wird. Abgesehen von anderen Membranen kann eine Membran eines verallgemeinerten P-Systems (s. [2]) auch Multimengen von speziellen Objekten und Operatoren sowie Evolutionsregeln für die Operatoren enthalten. Im Gegensatz zur ursprünglichen Definition von P-Systemen müssen bei verallgemeinerten P-Systemen nicht alle Objekte, auf die eine Regel anwendbar ist, auch wirklich von der Anwendung einer Regel betroffen sein. Der Übergang von einer Konfiguration in eine andere Konfiguration eines verallgemeinerten P-Systems erfolgt durch die Anwendung einer Evolutionsregel für Operatoren in einem Kompartment sowie die Anwendung der ausgewählten Regeln auf die in diesem Kompartment vorhandenen Objekte. In einem verallgemeinerten P-System mit verbotenem Kontext können Operatoren nur dann angewendet werden, wenn bestimmte andere Operatoren auf die Objekte im zugrundeliegenden Kompartment nicht anwendbar sind (verbotener Kontext). Eine Berechnung in einem P-System besteht aus einer endlichen Folge von aus der Startkonfiguration sequentiell ableitbaren Konfigurationen. Die Anzahl terminaler Objekte in einer haltenden Berechnung (auf die Endkonfiguration kann keine Evolutionsregel mehr angewendet werden) stellt das Ergebnis einer Berechnung in einem verallgemeinerten P-System dar.

Verallgemeinerte P-Systeme mit verbotenem Kontext sind eine spezielle Variante, welche nicht nur auf sehr spezielle Formen von Produktionen, sondern auch auf nur drei Typen von Evolutionsregeln sowie auf die einfachste Membranstruktur eingeschränkt sind. Basierend auf einem

Ergebnis von [4] können wir dennoch zeigen, daß diese verallgemeinerten P-Systeme mit verbotenem Kontext n-Registermaschinen (s. [5]) recht einfach simulieren können, was ihre Universalität beweist. Betrachten wir verallgemeinerte P-Systeme mit verbotenem Kontext und externer Ausgabe (die nacheinander aus der Hautmembran in die Außenwelt geschickten Symbole werden in der Reihenfolge ihres Erscheinens als Wörter interpretiert), können analoge Ergebnisse erzielt werden.

Überdies können wir zeigen, dass die Charakterisierung von rekursiv aufzählbaren Sprachen über einem einelementigen Alphabet durch verallgemeinerte P-Systeme mit verbotenem Kontext und externer Ausgabe, welche nur zwei unterschiedliche Symbole verwenden, optimal ist.

Eine ausführliche Darstellung verschiedener Varianten der oben beschriebenen verallgemeinerten P-Systeme mit verbotenem Kontext sowie vollständige Beweise der oben angeführten Resultate sind in [3] zu finden.

Literatur

[1] Dassow, J., Păun, Gh.: On the power of membrane computing, *Journal of Universal Computer Science* **5**, 2 (1999), 33–49 (http://www.iicm.edu/jucs).

[2] Freund, R.: Generalized P-systems, *Fundamentals of Computation Theory, FCT'99,* Iaşi, 1999 (Ciobanu, G., Păun, Gh., eds.), LNCS 1684, Springer-Verlag, Berlin (1999), 281–292.

[3] Freund, R., Oswald, M.: GP systems with forbidding context, *Fundamenta Informaticae* **49**, 1-3 (2002), 81–102.

[4] Freund, R., Păun, Gh.: On the number of non-terminal symbols in graph-controlled, programmed and matrix grammars, *Proc. Conf. Universal Machines and Computations,* Chişinău, 2001 (Margenstern, M., Rogozhin, Y., eds.), Springer-Verlag, Berlin (2001).

[5] Minsky, M. L.: *Berechnung: Endliche und unendliche Maschinen,* Verlag Berliner Union GmbH. Stuttgart, Verlag W. Kohlhammer GmbH. Stuttgart (1971).

[6] Păun, Gh.: Computing with Membranes, *Journal of Computer and System Sciences* **61**, 1 (2000), 108–143.

[7] Păun, Gh., Rozenberg, G., Salomaa, A.: Membrane Computing with External Output, *Fundamenta Informaticae* **41**, 3 (2000), 259–266.

Möglichkeiten der Lösung des CLIQUE-Problems

Renate Winter
Martin-Luther-Universität Halle-Wittenberg
e-mail: winter@informatik.uni-halle.de

KURZFASSUNG

We consider possibilities to solve the CLIQUE problem in special cases of graphs and with parallelism.

Schlüsselwörter: NP-Vollständigkeit, Parallelisierung

1. Einleitung

$CLIQUE1$ ist ein stark NP-vollständiges Problem. Derzeit ist kein Polynomialzeitalgorithmus bekannt zur Entscheidung, ob ein beliebiger Graph G eine k-clique hat. Bei bekannten Lösungsmethoden werden im Wesentlichen die $\binom{n}{k}$ k-elementigen Knotenteilmengen auf Clique-Eigenschaft getestet.

2. Grundlegende Definitionen

Definition *CLIQUE 1 (Entscheidungsvariante): Gegeben sei ein Graph $G = [V, E]$ mit Knotenmenge V und Kantenmenge $E \subseteq V \times V$, $|V| = n, k \in N, k < n$. Hat G eine k-clique, d.h. existiert ein $V' \subseteq V$ mit $|V'| = k$, so dass für jedes Paar $v_1, v_2 \in V'$ $[v_1, v_2] \in E$ gilt ?*
CLIQUE 2 (Zahlvariante): Bestimme die maximale Cliquezahl k für $G = [V, E]$.
CLIQUE 3 (Optimierungsvariante): Gib die größte Clique V' für $G = [V, E]$ an.

Definition (1)–0–TM–SPACE(s) *ist die Familie der Probleme, die auf einer* 1*-bändrigen deterministischen Turingmaschine mit Speicherplatzkomplexität $s(\nu)$ für Eingabewortlänge ν entscheidbar sind.*
(1)–E–TM–SPACE(s) *ist die Familie der Probleme, die auf einer deterministischen Turingmaschine mit Eingabe- und Arbeitsband mit Speicherplatzkomplexität $s(\nu)$ für Eingabewortlänge ν entscheidbar sind. Dabei wird der Speicherbedarf nur auf dem Arbeitsband gemessen.*

Definition $PRAM$–TIME(t) *ist die Familie der Probleme, die auf einer PRAM mit Zeitkomplexität $t(\nu)$ für Eingabewortlänge ν entscheidbar sind.*

3. Nichtapproximierbarkeit des CLIQUE-Problems

Aus der PCP-Theorie wissen wir, dass das Clique-Problem nicht durch Approximationsalgorithmen mit akzeptablem Resultat gelöst werden kann.

Theorem *Für beliebiges $\varepsilon > 0$ gilt: Es gibt keinen Polynomialzeit-Approximationsalgorithmus für $CLIQUE2$ mit Güte $n^{1-\varepsilon}$, falls NP $\neq$ co–RP gilt. Es gibt keinen Poynomialzeit-Approximationsalgorithmus für $CLIQUE2$ mit Güte $n^{1/2-\varepsilon}$, falls NP $\neq$ P gilt* [2].

4. Erschöpfendes Suchen

In bekannten sequentiellen Algorithmen zur Entscheidung des $CLIQUE$-Problems werden im Prinzip $\binom{n}{k}$ Knotenteilmengen inspiziert. Das Lemma soll helfen zu entscheiden, für welche k Exponentialzeitalgorithmen noch praktikabel sind.

Lemma *Gegeben sei ein Graph $G = [V, E]$ mit $|V| = n$ und eine natürliche Zahl $k \leq n$. Die Zahl ζ aller möglichen Teilmengen $V' \subseteq V$ mit $|V'| = k$ lässt sich u.a. wie folgt abschätzen:*

1. $\frac{1}{\sqrt{2n+1}} 4^{n/2} < \zeta < \frac{1}{\sqrt{n+1}} 4^{n/2}$ *für* $k = n/2$
2. $\frac{8}{\sqrt{5n}} \left(\frac{256}{27}\right)^{n/4} < \zeta < 2\left(\frac{256}{27}\right)^{n/4}$ *für* $k = n/4$.

5. Behandlung spezieller Graphen, Greedy und Bron-Kerbosch-Algorithmus

Die Ausnutzung der speziellen Struktur eines Graphen (z.B. Planarität des Komplementgraphen) führt zu effizienten Algorithmen. Ein Greedy-Algorithmus bringt in vielen Fällen eine maximale Clique. Der Algorithmus von Bron/Kerbosch liefert gute Resultate bei Graphen, die keine Moon-Moser-Graphen sind oder Filze enthalten [1]. Parallelisierung bringt Laufzeitverbesserung.

6. Parallelisierung aus theoretischer Sicht

Aufgrund der parallelen Berechnungshypothese lassen sich Probleme, für deren Lösung es sequentielle Algorithmen mit polynomiell beschränktem Speicherplatz gibt, auf Parallelrechnern in Polynomialzeit lösen. Lässt sich ein Problem durch einen sequentiellen Algorithmus mit Speicherplatz $s(\nu)$ bei Eingabelänge ν lösen, so existiert ein äquivalenter paralleler Algorithmus, der mit Laufzeit $s(\nu)$ auskommt. Verwenden wir die gewöhnliche 1-bändrige Turingmaschine als sequentielles Rechnermodell, die PRAM als paralleles Rechnermodell, so ergibt sich (1)–0–TM–SPACE$(s) \subseteq PRAM$–TIME$(\Theta(s))$ für $s \geq id_N$ mit Identitätsfunktion $id(\nu) = \nu$ [4]. Wegen $CLIQUE1 \in$ (1)–0–TM–SPACE(ν) erhalten wir

Theorem $CLIQUE1 \in PRAM$–TIME(ν).

7. Ausblick auf weitere Verfahren

Theorem *Es sei $G = [V, E]$ ein Graph mit Knotenmenge $|V| = n$. Wenn es einen polynomiellen Algorithmus für CLIQUE1 (G, k) für $k < \log n$ gibt, dann gibt es auch einen Algorithmus für CLIQUE1 (G, k) mit Zeitkomplexität $T(n, k) = 2^{O(\sqrt{k} \log n)}$ für beliebiges $k \leq n$.*

Theorem *Für $G = [V, E]$ mit $|V| = n$ gilt:*
CLIQUE1 (G, k) für $k \geq n - O(\log n)$ ist in Polynomialzeit entscheidbar.

Literatur

[1] M. B. Eggers. *Ordering on Graphs and Computations of CLIQUE.* Technical Report No. 91-03, TU Berlin, FB 20, 1991

[2] E. W. Mayr, H. J. Prömel, A. Steger. *Lectures on Proof Verification and Approximation Algorithms.* Springer, München Berlin 1998

[3] R. Niedermeier, P. Rossmanith. *Upper Bounds for Vertex Cover Further Improved.* Proceedings of the 16th STACS, Springer LNCS 1999

[4] R. Vollmar, T. Worsch. *Modelle der Parallelverarbeitung.* B.G. Teubner, Stuttgart 1993

Autorenverzeichnis

Teilnehmerverzeichnis

JAN-HENRIK ALTENBERND
Lehrstuhl für Informatik VII, RWTH Aachen
Ahornstraße 55, 52056 Aachen
altenbernd@i7.informatik.rwth-aachen.de

BJÖRN BORCHARDT
Institut für Theoretische Informatik, Technische Universität Dresden
Cottaer Straße 8, 01159 Dresden
borchard@tcs.inf.tu-dresden.de

VASCO BRATTKA
Theoretische Informatik 1, Fern-Universität Hagen
58084 Hagen
Vasco.Brattka@FernUni-Hagen.de

JÜRGEN DASSOW
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Postfach 41 20, 39016 Magdeburg
dassow@iws.cs.uni-magdeburg.de

RUDOLF FREUND
Institut für Computersprachen, Technische Universität Wien
Favoritenstraße 9, A-1040 Wien, Austria
rudi@logik.at

PETER HERTLING
Lehrgebiet Berechenbarkeit und Logik, Fern-Universität Hagen
58084 Hagen
peter.hertling@fernuni-hagen.de

MARKUS HOLZER
Institut für Informatik, Technische Universität München
Arcisstraße 21, 80290 München
holzer@informatik.tu-muenchen.de

ANDREAS KLEIN
Fachbereich Mathematik/Informatik, Universität Gesamthochschule Kassel
Heinrich-Plett-Straße 40, 34132 Kassel
klein@mathematik.uni-kassel.de

ROMAN KÖNIG
Lehrstuhl für Informatik II, Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 3, 91058 Erlangen
Roman.Koenig@informatik.uni-erlangen.de

MARTIN KUTRIB
Institut für Informatik, Universität Giessen
Arndtstraße 2, 35392 Giessen
kutrib@informatik.uni-giessen.de

JAN-THOMAS LÖWE
Institut für Informatik, Universität Giessen
Arndtstraße 2, 35392 Giessen
Jan-Thomas.B.Loewe@informatik.uni-giessen.de

RENÉ MAZALA
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1, 06099 Halle (Saale)
mazala@informatik.uni-halle.de

NORBERT TH. MÜLLER
Abteilung Informatik, Universität Trier
54286 Trier
mueller@uni-trier.de

MARION OSWALD
Institut für Computersprachen, Technische Universität Wien
Favoritenstraße 9, A-1040 Wien, Austria
marion@emcc.at

FRIEDRICH OTTO
Fachbereich Mathematik/Informatik, Universität Gesamthochschule Kassel
Heinrich-Plett-Straße 40, 34132 Kassel
otto@flower.theorie.informatik.uni-kassel.de

BERND REICHEL
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Postfach 41 20, 39016 Magdeburg
reichel@iws.cs.uni-magdeburg.de

LUDWIG STAIGER
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1, 06099 Halle (Saale)
staiger@informatik.uni-halle.de

RALF STIEBE
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Postfach 41 20, 39016 Magdeburg
stiebe@iws.cs.uni-magdeburg.de

HEIKO VOGLER
Institut für Theoretische Informatik, Technische Universität Dresden
Mommsenstraße 13, 01062 Dresden
vogler@orchid.inf.tu-dresden.de

KLAUS WEIHRAUCH
Theoretische Informatik 1, Fern-Universität Hagen
58084 Hagen
Klaus.Weihrauch@FernUni-Hagen.de

RENATE WINTER
Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1, 06099 Halle (Saale)
winter@informatik.uni-halle.de