GhK

Universität
Gesamthochschule
Kassel

Mathematische Schriften Kassel
Preprint-Reihe
Fachbereich Mathematik/Informatik

9. Theorietag der GI-Fachgruppe 0.1.5 „Automaten und Formale Sprachen“

Schauenburg-Elmshagen bei Kassel

27.-29.9.1999

Friedrich Otto, Gundula Niemann (Hrsg.)

Preprint No. 12/99
September 1999

Vorwort

Seit 1991 veranstaltet die GI-Fachgruppe 0.1.5 *Automaten und Formale Sprachen* jährlich einen Theorietag, der eineinhalb Tage dauert, und auf dem auch die jährliche Fachgruppensitzung stattfindet. Nach Magdeburg (1991), Kiel (1992), Dagstuhl (1993), Herrsching (1994), Schloss Rauischholzhausen (1995), Cunnersdorf (1996), Barnstorf (1997) und Riveris (1998) findet der 9. Theorietag in Schauenburg-Elmshagen statt. Wie seit 1996 üblich, geht dem eigentlichen Theorietag ein eintägiger Workshop voraus, der in diesem Jahr unter dem Thema *Formale Sprachen, Automaten und Ersetzungssysteme* steht. Als Vortragende für diesen Workshop haben

- Jean Berstel (Marne-la-Vallée, Frankreich),
- Yuji Kobayashi (Funabashi, Japan),
- Gundula Niemann (Kassel),
- Birgit Reinert (Kaiserslautern) und
- Andrea Sattler-Klein (Kaiserslautern)

zugesagt.

Als Besonderheit wird am Vorabend des eigentlichen Theorietages eine Podiumsdiskussion zum Thema *Theoretische Informatik in der Lehre* veranstaltet, ein Vorhaben, dass von Teilnehmern des Theorietages in Riveris angeregt worden ist. Die Diskusionsleitung hat freundlicherweise Annegret Habel (Oldenburg) übernommen, und als Diskussionsteilnehmer auf dem Podium haben Jean Berstel (Marne-la-Vallée), Norbert Blum (Bonn), Hans-Jörg Kreowski (Bremen), Klaus Madlener (Kaiserslautern), Wolfgang Thomas (Aachen) und Heiko Vogler (Dresden) zugesagt.

Außer den oben genannten Vortragenden und den Teilnehmern der Podiumsdiskussion haben sich 28 weitere Teilnehmer für den Theorietag angemeldet, dessen Programm aus 21 Vorträgen besteht. Die Zusammenfassungen sowohl dieser Vorträge als auch der Vorträge des Workshops sind in dem vorliegenden technischen Bericht enthalten. Ferner finden sich hier die Programme des Workshops und des Theorietages sowie eine Liste aller Teilnehmer samt ihrer postalischen und elektronischen Adressen.

Der GI sowie dem Fachbereich Mathematik/Informatik der Universität Kassel möchten wir für die finanzielle Unterstützung dieses Theorietages danken. Wir wünschen allen Teilnehmern einen anregenden und interessanten Theorietag sowie einen angenehmen Aufenthalt in Elmshagen.

Kassel, den 9.9.1999

Gundula Niemann
Friedrich Otto

Inhaltsverzeichnis

Anhang

Workshop

„Automaten, Formale Sprachen und Ersetzungssysteme“

Schauenburg-Elmshagen, Montag, den 27.9.1999

Programm

$9.^{15}$	Begrüßung
$9.^{20}$ – $10.^{25}$	*Jean Berstel* Sturmian Words, and Beyond
$10.^{25}$ – $10.^{55}$	Kaffeepause
$10.^{55}$ – $12.^{00}$	*Gundula Niemann* Über Klassen der Church-Rosser Sprachen und ihr Verhältnis zu anderen Sprachklassen
$12.^{00}$ – $13.^{45}$	Mittagspause
$13.^{45}$ – $14.^{50}$	*Andrea Sattler-Klein* Unentscheidbarkeitsresultate für endlich dargestellte Monoide
$14.^{50}$ – $15.^{55}$	*Birgit Reinert* Gröbnerbasen in Monoid- und Gruppenringen
$15.^{55}$ – $16.^{25}$	Kaffeepause
$16.^{25}$ – $17.^{30}$	*Yuji Kobayashi* On the termination of one-rule string-rewriting systems
$17.^{30}$ – $19.^{30}$	Abendessen
$19.^{30}$ – $21.^{00}$	Podiumsdiskussion zum Thema: „Theoretische Informatik in der Lehre“

9. Theorietag

„Automaten und Formale Sprachen"

Schauenburg-Elmshagen, 28.–29.9.1999

Programm

Dienstag, den 28.9.99

$8.^{55}$	Eröffnung des Theorietages
$9.^{00}$ – $9.^{25}$	*Bianca Truthe*: Zur Endlichkeit von Bildsprachen synchroner deterministischer Ketten-Code-Bild-Systeme
$9.^{25}$ – $9.^{50}$	*Ludwig Staiger*: Finite automata encoding geometric figures
$9.^{50}$ – $10.^{15}$	*Frank Drewes*: Collagengrammatiken und iterierte Funktionensysteme mit Farboperationen
$10.^{15}$ – $10.^{45}$	Kaffeepause
$10.^{45}$ – $11.^{10}$	*Annegret Habel*: On the generative power of graph grammars with injective matching
$11.^{10}$ – $11.^{35}$	*Detlef Plump*: Church-Rosser hypergraph languages
$11.^{40}$ – $12.^{05}$	*Jens Woinowski*: Eine unter Bildung der Präfixsprache abgeschlossene Teilmenge der Church-Rosser-Sprachen
$12.^{05}$ – $12.^{30}$	*Markus Lohrey*: Zur Präperfektheit von Semi-Thue-Systemen
$12.^{30}$ – $14.^{15}$	Mittagspause
$14.^{15}$ – $14.^{40}$	*Martin Leucker*: Erfüllbarkeit globaler Spurlogiken mittels alternierender Automaten
$14.^{40}$ – $15.^{05}$	*Klaus Wich*: Zerlegung polynomiell mehrdeutiger kontextfreier Grammatiken
$15.^{05}$ – $15.^{30}$	*Johannes Waldmann*: Boolesche Operationen auf Wort- und Baum-Automaten mit kurzen Schleifen
$15.^{30}$ – $16.^{00}$	Kaffeepause
$16.^{00}$ – $16.^{25}$	*Heiko Vogler*: Using recursion in MSO tree transductions
$16.^{25}$ – $16.^{50}$	*Thomas Noll*: Kohärenzeigenschaften in termersetzungsbasierten Modellen für verteilte Systeme
$16.^{55}$ – $17.^{20}$	*Gerhard Buntrock*: XML, eine Verwandte der Dyck-Sprachen
$17.^{20}$ – $17.^{45}$	*Dieter Hofbauer*: XSSR: An Experimental System for String-Rewriting — Decision Problems, Algorithms, and Implementation
$18.^{00}$ – $18.^{45}$	Fachgruppensitzung
$19.^{00}$	Abendessen

9. Theorietag

„Automaten und Formale Sprachen"

Schauenburg-Elmshagen, 28.–29.9.1999

Programm

Mittwoch, den 29.9.99

$9.^{00}$ – $9.^{25}$	*Andreas Klein*: Zeithierarchiesätze für iterative Arrays
$9.^{25}$ – $9.^{50}$	*Martin Kutrib*: Kontextfreie Sprachen und Polyautomaten
$9.^{50}$ – $10.^{15}$	*Henning Bordihn*: H-Ausdrücke - eine Erweiterung regulärer Ausdrücke
$10.^{15}$ – $10.^{45}$	Kaffeepause
$10.^{45}$ – $11.^{10}$	*Rudolf Freund*: Algebraische Repräsentation regulärer Array-Sprachen
$11.^{10}$ – $11.^{35}$	*Dietrich Kuske*: Rationale Teilbarkeitsmonoide
$11.^{40}$ – $12.^{05}$	*Holger Petersen*: Beschreibungsgröße zyklischer Sprachen
$12.^{05}$ – $12.^{30}$	*Christof Löding*: Die Theorie der ω-regulären Sprachen aus der Sicht der Alternierenden Schwachen Automaten
$12.^{30}$	Mittagessen und Ende des Theorietages

Workshop

„Automaten, Formale Sprachen und Ersetzungssysteme“

Sturmian Words, and Beyond
(Extended Abstract)

Jean Berstel

Institut Gaspard Monge
Université de Marne-la-Vallée

Abstract

This survey is on combinatorial and arithmetic properties of a special class of infinite words called *Sturmian* words. It also addresses extensions of this family to words on more than two letters and to planes.

We start with the following question: "What is a discretized straight line?" Consider the converse problem: Given a straight line $y = ax + b$, with a, b reals and w.l.o.g. $0 < a < 1$, how is it plotted on a raster display? Every algorithm (and Bresenham's algorithm does it in a clever way) computes pairs of integers $(k, \lfloor ak + b \rfloor)$ (or closely related pairs) for a given interval $\ell \leq k \leq r$. The problem stated initially is a basic question in pattern recognition. It is to "recognize" the parameters of the equation, that is to compute a, b or reasonable approximations of it, from the set $(k, \lfloor ak + b \rfloor)$ of points. More generally, a set of points may possibly be decomposed into several contiguous discretized line segments, giving thus a discretized polygonal line. We present several older and newer algorithms developed to handle these and related problems.

Next, we consider discretized line segments for their own. The usual encoding (Freeman encoding) associates to a line segment from point $(0,0)$ to point (p, q) with $0 < q < p$ and p, q relatively prime, a word over a binary alphabet. This word has p occurrences of one letter and $p - q$ occurrences of the other. One of the first authors to consider these words was Christoffel, at the end of the XIXth century. Several combinatorial properties of these words will be given. In particular, when the first and the last letters are dropped, the resulting word is always a primitive palindrom. Moreover, it is a product of two palindroms. These are characterized as being extremal words with respect to the famous Fine and Wilf periodicity theorem: If a word x has two periods p and q which are relatively prime, and if the length of x is at least $p + q - 1$, then x is a unary word (that is all its letters are the same). The discretized line segments are examples showing that the bound in Fine and Wilf's theorem is optimal. More precisely, they constitue *all* examples for the optimality of the bound. This property has been extended recently to a three period version of Fine and Wilf's theorem.

A slight modification of the initial object of investigation leads to a large amount of new insights, in connection with combinatorics, number theory and dynamical systems. The modification consists in considering lines $y = ax + b$, where the slope a is assumed to be irrational $(0 < a < 1)$. The discretized straight lines obtained were called *Sturmian* words by Morse and Hedlund in 1940. They are infinite words. The most famous of these is the Fibonacci

word. Sturmian words of a given slope are minimal symbolic dynamical systems. Thus, the set of factors depends only of the slope in the equation, not of the intercept b. We will give an arithmetic proof of this. Sturmian words have a dozen of different characterizations, some of which we will discuss. One of the most interesting characterizations is in connection with combinatorics on words, and is known since 1940: Sturmian words are of lowest nontrivial complexity. To be more precise, define the *complexity* function p_x of given word x by setting $p_x(n)$ to be the number of factors (blocs) of x of length n. Clearly, p_x is an increasing function. It is not difficult to show that if $p_x(n) \leq n$ for some n, then x is eventually periodic. Thus, an aperiodic word x satisfies the condition $p_x(n) \geq n + 1$ for all $n \geq 0$. Sturmian words are exactly those words for which equality holds at each stage.

The structure of a Sturmian word is closely related to its slope, and in fact to the continued fraction expansion of its slope. The very simple recurrence formula for the construction of the Fibonacci word stems from the very simple continued fraction expansion of the golden ratio. More precise information, e.g. on the powers of words appearing in a Sturmian word can be derived from the partial quotients of the expansion. In particular, powers appearing in a Sturmian word have bounded exponent if and only if the partial quotients are bounded. This striking result of Mignosi has been revisited very recently, in view of giving a quantitative formulation.

The investigation of discretized planes is not yet so developed as it is the case for straight lines. Algorithms to recognize digitized planes are merely brute force. Characterizations similar to those of Sturmian words are sometimes available, sometimes open, sometimes false. The last part of this survey will give a short account of the state of the art.

On the termination problem of one-rule string-rewriting systems

Yuji Kobayashi

Faculty of Science
Toho University, Funabashi, Japan

String-rewriting systems (semi-Thue systems) are special term-rewriting systems where all the function symbols are of arity one. One-rule string-rewriting systems are thus considered to be the simplest examples of rewriting systems. Even for these simplest systems, two fundamental problems still remain unsolved, the word problem and the termination problem.

It is undecidable whether a finite string-rewriting system is terminating ([1]). Actually, termination is undecidable for three-rule string-rewriting systems ([3]). However, it is not known whether the termination problem is decidable for one-rule string-rewriting systems, though the confluence of such systems is decidable ([2], [6]).

The first systematical study on the termination problem of one-rule string rewriting systems was attempted by Kurth [2]. He gave several criterions for termination. McNaughton [4] introduced the notion of 'well-behaved' derivation and showed that termination of well-behaved rewriting is decidable. Zantema and Geser [7] gave a characterization for termination of the special systems $\{a^p b^q \to b^n a^m\}$.

For termination of a confluent one-rule system $\{s \to t\}$, it suffices to treat the case where s is self-overlap-free [5]. Since the termination problem for general one-rule systems seems to be very difficult, the confluent case is a good intermediate goal for study.

Let $\{s \to t\}$ be a one-rule system over an alphabet Σ such that s is self-overlap-free. In derivation steps we look at positions (cuts) where the rule is applied. A position of a word x is a pair (x', x'') of a prefix x' and a suffix x'' of x such that $x = xx''$. If $x = x'sx''$ is rewritten to $y = x'tx''$ by the rule $s \to t$, the two positions (x', tx'') and $(x't, x'')$ of y are the seams of y created by the step $x'sx'' \to_R x'tx''$, and all the seams of x inside the subword s are destroyed or patched. For each seam σ of x we give a label $\phi(\sigma)$ which is a letter in a certain alphabet different from Σ. The trace $\phi(x)$ of x is a word $\phi(\sigma_1)\cdots\phi(\sigma_k)$ spelling the labels of seams of x from left to right. If by a step $x \to_R y$, seams $\sigma_i, \cdots, \sigma_j$ are patched and a pair of seams τ_1 and τ_2 are created, then the trace $\phi(x) = \phi(\sigma_1)\cdots\phi(\sigma_k)$ of x is transformed (rewritten) to the trace $\phi(\sigma_1)\cdots\phi(\sigma_{i-1})\phi(\tau_1)\phi(\tau_2)\phi(\sigma_{j+1})\cdots\phi(\sigma_k)$ of y.

Through this mechanism we associate another rewriting system over another alphabet with the original system R. The new system simulates the original one. It is not any more a one-rule system and looks complex, but an analysis of termination for it is sometimes easier than for the original system.

References

[1] G. Huet and D. Lankfort, On the uniform halting problem for term rewriting systems, Technical Report 283, INRIA, 1978.

[2] W. Kurth, Termination und Konfluenz von Semi-Thue-systemen mit nur einer Regel, Dissertation, Technischen Univ. Clausthal, 1990.

[3] Y. Matiyasevich and G. Sénizergues, Decision problems for semi-Thue systems with a few rules, Proc. LICS'96 IEEE (1996), 523-531.

[4] R. McNaughton, Well behaved derivations in one-rule Semi-Thue systems, Rensselaer Polytechnic Institute Report 95-15, 1995.

[5] K. Shikishima-Tsuji, M. Katsura, and Y. Kobayashi, On termination of confluent one-rule string-rewriting systems, Inform. Process. Let. **61** (1997), 91-96.

[6] C. Wrathall, Confluence of one-rule Thue systems, in: Word Equations and Related Topics, Lect. Not. Comp. Sci. **572** (1992), 237-246.

[7] H. Zantema and A. Geser, A complete characterization of termination of $0^p1^q \to 1^r0^s$, Lect. Not. Comp. Sci. **914**, (1995), 41-55.

Über die Klasse der Church-Rosser Sprachen und ihr Verhältnis zu anderen Sprachklassen

Gundula Niemann

Universität Kassel

Verkürzende konfluente Wortersetzungssysteme bieten bequeme Möglichkeiten, Wörter zu manipulieren. So kann man in linearer Zeit zu jedem Wort einen irreduziblen Nachfolger bestimmen. Überdies ist der irreduzible Nachfolger eindeutig bestimmt. Das heißt, zwei Wörter sind genau dann äquivalent, wenn ihre irreduziblen Nachfolger identisch sind. Also ist das Wortproblem für ein endliches verkürzendes CR-System in linearer Zeit lösbar.

Hierdurch motiviert haben McNaughton, Narendran und Otto die folgenden Sprachklassen eingeführt [MNO88].

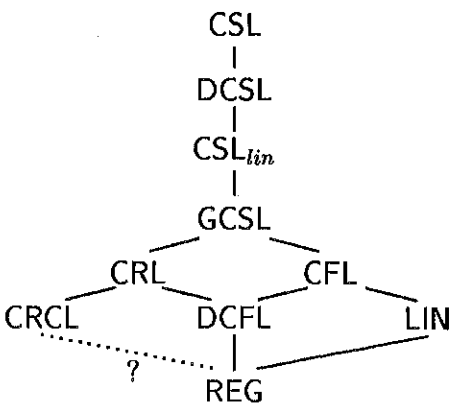
Definition 1. *Eine Sprache $L \in \Sigma^*$ ist eine Church-Rosser-Sprache, abgekürzt* CRL, *genau dann, wenn es ein endliches verkürzendes konfluentes Wortersetzungssystem R über einem Alphabet $\Gamma \supset \Sigma$ gibt und dazu irreduzible Wörter $t_1, t_2 \in (\Gamma \setminus \Sigma)^*$ und einen irreduziblen Buchstaben $Y \in \Gamma \setminus \Sigma$, so daß für jedes Wort $w \in \Sigma^*$ gilt:*

$$w \in L \Leftrightarrow t_1 w t_2 \to_R^* Y$$

Eine Sprache L ist eine Church-Rosser-Kongruenz-Sprache, abgekürzt CRCL, *genau dann, wenn sie als Vereinigung endlich vieler Kongruenzklassen eines endlichen verkürzenden konfluenten Wortersetzungssystems darstellbar ist.*

Das Elementproblem für CRL ist also in linearer Zeit lösbar, und CRL liegt zwischen DCFL und CSL [MNO88].

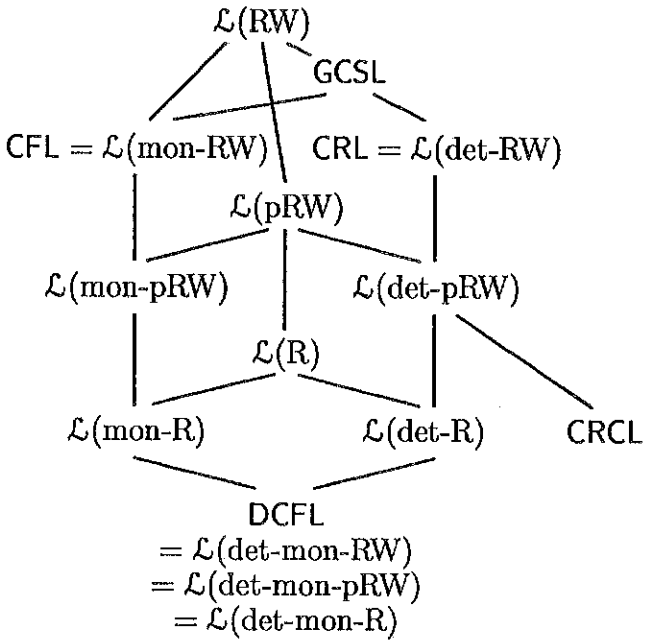
Die Klasse der wachsend kontextsensitiven Sprachen (GCSL) umfaßt genau die Sprachen, die von Grammatiken mit ausschließlich verlängernden Regeln erzeugt werden. Sie liegt zwischen CFL und CSL. Hier läßt sich das Elementproblem deterministisch in polynomieller Zeit entscheiden [DW86]. Buntrock und Otto charakterisierten GCSL durch sogenannte schrumpfende Zweikeller-Automaten [BO98]. Deren deterministische Version charakterisiert eine verallgemeinerte Form der Church-Rosser Sprachen (genannt GCRL), in der statt verkürzenden gewichtsreduzierende Wortersetzungssysteme zugrundegelegt werden [BO98].



Die beiden Sprachklassen CRL und GCRL fallen zusammen [NO98], und so ergibt sich das nebenstehende Bild. Mit dieser Gleichheit lassen

sich unter anderem die Kenntnisse über Abschlußeigenschaften zusammenfügen. Außerdem ist CRL eine Basis für die rekursiv aufzählbaren Sprachen [Bun96, OKK97]. Darunter verstehen wir, daß der Abschluß von CRL unter allgemeinen Homomorphismen gleich der Menge der rekursiv aufzählbaren Sprachen ist.

Man erhält eine weitere Charakterisierung von CRL durch die von Jančar, Mráz, Plátek, and Vogel eingeführten Restarting Automaten mit Rewriting, kurz RW-Automaten [JMPV97b]. Durch Einschränkungen des allgemeinen Modells wie deterministisch (det), pur (abgekürzt p), ohne Überschreiben von Buchstaben (R-Automaten), oder monoton (mon), entsteht eine Reihe von Automatentypen. Die Beziehungen zwischen ihnen wurden in [JMPV97b, JMPV97a] untersucht. Niemann und Otto fügten CRL in diese Landschaft ein, indem sie diese Klasse durch die deterministischen RW-Automaten charakterisierten [NO99]. Damit ergibt sich das nebenstehende Bild.



Das Verhältnis von CRL zu anderen Sprachklassen wird unter anderem durch möglichst typische Sprachen deutlich.

	Sprache	markante Eigenschaften
1	$\{a^n b^n c^n : n \geq 0\}$	$\notin$ CFL, $\in$ CRL
2	$\{a^{2^n} : n \geq 0\}$	$\notin$ CFL, unär, $\in$ CRL
3	$\{ww : w \in \{a,b\}\}$	$\notin$ CFL, $\notin$ CRL, $\in$ indexed languages
4	$\{(a^n b)^n : n \geq 0\}$	$\in$ CRL, $\notin$ indexed languages
5	$\{ww^R : w \in \{a,b\}\}$	$\notin$ DCFL, $\in$ GCSL

Aus Zeile 2 folgt, daß die programmierten kontextfreien Grammatiken ohne appearance checking (kurz rC) nicht jede CRL erzeugen können, da jede unäre Sprache aus $\mathcal{L}$(rC) regulär ist (siehe [DPS96]). Aus Zeilen 3 und 4 folgt die Unvergleichbarkeit der indexed languages mit CRL.

Offene Fragen

- Ist GCSL = $\mathcal{L}$(RW) ?
- Ist $\{ww^R : w \in \{a,b\}\} \in$ CRL ? Wie beweist man das Gegenteil?
- Ist REG $\subset$ CRCL ?

Weiterführende Fragen

- Kann jede GCSL (CRL) von einer programmierten kontextfreien Grammatik mit appearance checking (kurz $\mathrm{rC}_{\mathrm{ac}}$) erzeugt werden? Welche Sprachen liegen in CSL $\setminus \mathcal{L}(\mathrm{rC}_{\mathrm{ac}})$?
- Wie liegen CRL und GCSL im Verhältnis zu durch L-Systeme definierten Sprachklassen?

Literatur

[BO98] G. Buntrock and F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Information and Computation*, 141:1–36, 1998.

[Bun96] G. Buntrock. *Wachsende kontextsensitive Sprachen.* Habilitationsschrift, Fakultät für Mathematik und Informatik, Universität Würzburg, July 1996.

[DPS96] J. Dassow, G. Păun, and A. Salomaa. Grammars with controlled derivations. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, vol. 2, pp. 101–154. Springer, Berlin/New York, 1996.

[DW86] E. Dahlhaus and M. K. Warmuth. Membership for growing context–sensitive grammars is polynomial. *Journal of Computer and System Sciences*, 33:456–472, 1986.

[JMPV97a] P. Jančar, F. Mráz, M. Plátek, and J. Vogel. Monotonic rewriting automata with a restart operation. In F. Plášil and K. G. Jeffery, editors, *SOFSEM'97: Theory and Practise of Informatics*, LNCS no. 1338, pp. 505–512, Springer, Berlin/New York, 1997.

[JMPV97b] P. Jančar, F. Mráz, M. Plátek, and J. Vogel. On restarting automata with rewriting. In G. Paun and A. Salomaa, editors, *New Trends in Formal Languages*, LNCS no. 1218, pp. 119–136, Springer, Berlin/New York, 1997.

[MNO88] R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association Computing Machinery*, 35:324–344, 1988.

[NO98] G. Niemann and F. Otto. The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. In M. Nivat, editor, *Foundations of Software Science and Computation Structures, Proceedings FoSSaCS'98*, LNCS no. 1378, pp. 243–257, Springer, Berlin/New York, 1998.

[NO99] G. Niemann and F. Otto. Restarting automata, Church-Rosser languages, and representations of r.e. langugaes. In W. Thomas, editor, *Develpments in Language Theory, 4th International Conference, Preproceedings*, Aachener Informatik-Berichte no. 99-5, pp. 49–62. RWTH Aachen, Fachgruppe Informatik, 1999.

[OKK97] F. Otto, M. Katsura, and Y. Kobayashi. Cross-sections for finitely presented monoids with decidable word problems. In H. Comon, editor, *Rewriting Techniques and Applications, Proceedings RTA'97*, LNCS no. 1232, pp. 53–67, Springer, Berlin/New York, 1997.

Gröbnerbasen in Monoid- und Gruppenringen

Birgit Reinert

Fachbereich Informatik
Universität Kaiserslautern

1965 wurde von Bruno Buchberger im Rahmen seiner Dissertation [2] das Konzept der *Gröbnerbasen* als algebraische Simplifikationsmethode für Polynomringe über Körpern entwickelt. Gröbnerbasen sind endliche Idealbasen, die einen kanonischen Simplifikationsprozeß ermöglichen. Dabei wird ein Polynom p mit einem anderen Polynom f reduziert, falls ein geeignetes Monomvielfaches von f benutzt werden kann, um p durch Subtraktion zu verkleinern (bezüglich einer sogenannten zulässigen Ordnung). Benutzt man in diesem Simplifikationsprozeß nur Elemente einer Gröbnerbasis eines Ideals, so erhält man eindeutige Repräsentanten bezüglich der Idealkongruenz (Repräsentant der Idealelemente ist die Null). Somit erlauben Gröbnerbasen die effektive Lösung vieler idealtheoretischer Probleme (vgl. [3]). Insbesondere sind Gröbnerbasen von Idealen auch berechenbar, denn Buchberger gab in seiner Arbeit ein effektives Verfahren, den sogenannten *Buchberger Algorithmus*, an, um eine beliebige endliche Idealbasis in eine endliche Gröbnerbasis zu transformieren.

Auch bei der Untersuchung anderer algebraischer Strukturen interessiert man sich für sogenannte Moduln oder Ideale und sucht nach Wegen, algebraische Simplifikationsmethoden zur effektiven Lösung von Problemen einzusetzen. Mein Vortrag beschäftigt sich in diesem Zusammenhang hauptsächlich mit *Monoid- und Gruppenringen.* Hier werden algebraische Simplifikationsmethoden zweifach eingesetzt: einerseits zur Realisierung des effektiven Rechnens in einem Monoid oder einer Gruppe und andererseits zur Beschreibung von Idealkongruenzen. Da im Allgemeinen Monoide und insbesondere Gruppen keine zulässigen, d.h. mit der entsprechenden Multiplikation verträglichen, Ordnungen erlauben, treten bei der Definition einer geeigneten Reduktionsrelation (Simplifikation) wesentliche Probleme auf: Zum einen ist es schwierig, die Terminierung einer Reduktionsrelation zu garantieren, zum anderen braucht man zusätzliche Techniken, um die Idealkongruenz mit einer Reduktionsrelation zu erfassen. Beides hat natürlich Auswirkungen auf das Ziel, eindeutige Repräsentanten von äquivalenten Elementen im Monoid- oder Gruppenring auszuzeichnen. In [9] habe ich alternative Möglichkeiten, Reduktionsrelationen zur Beschreibung von Rechtsidealkongruenzen in Monid- und Gruppenringen zu definieren, aufgezeigt. In die untersuchten Definitionen gehen sowohl syntaktische Ersetzungskriterien als auch semantisches Wissen über die speziell behandelten Klassen von Monoiden und Gruppen ein. Positive Ergebnisse wurden sowohl für die Klasse der endlichen Monoide und die Klasse der endlich dargestellten freien Monoide als auch für die Klasse der endlich dargestellten freien Gruppen, die Klasse der endlich dargestellten ebenen Gruppen, die Klasse der endlich dargestellten kontext-freien Gruppen und die Klasse der endlich dargestellten nilpotenten Gruppen erzielt. Diese wurden in [7] für die Klasse der polyzyklischen Gruppen erweitert. Bei einer näheren Analyse der Ergebnisse hat sich insbesondere gezeigt, daß bekannte Methoden zur Behandlung des Untergruppenproblems

mit algebraischen Simplifikationsmethoden (vergleiche z.B. [1, 4, 5, 6, 11]) sich als Spezialfall der in diesem Zusammenhang entwickelten Gröbnerbasen auffassen lassen ([9]). Auch das bekannte Verfahren von Todd und Coxeter zur Aufzählung von Nebenklassen von Untergruppen läßt sich als Simplifikationsverfahren interpretieren und mit Gröbnerbasen formulieren [10, 8].

Literatur

[1] J. Avenhaus and K. Madlener. The Nielsen reduction and p-complete problems in free groups. *Theoretical Computer Science*, 32:61–76, 1984.

[2] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal.* PhD thesis, Universität Innsbruck, 1965.

[3] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer, 1992.

[4] R. Cremanns and F. Otto. Constructing canonical presentations for subgroups of context-free groups in polynomial time. In J. von zur Gathen and M. Giesbrecht, editors, *Proc. ISSAC'94*, pages 147–153. ACM, 1994.

[5] N. Kuhn. *Zur Entscheidbarkeit des Untergruppenproblems für Gruppen mit kanonischen Darstellungen.* PhD thesis, Universität Kaiserslautern, 1991.

[6] N. Kuhn and K. Madlener. A method for enumerating cosets of a group presented by a canonical system. In G. Gonnet, editor, *Proc. ISSAC'89*, pages 338–350. ACM, 1989.

[7] K. Madlener and B. Reinert. Gröbner bases in non-commutative reduction rings. In B. Buchberger and F. Winkler, editors, *Gröbner Bases and Applications (Proc. of the Conference 33 Years of Gröbner Bases)*, volume 251 of *London Mathematical Society Lecture Notes Series*, pages 408–420. Cambridge University Press, 1998.

[8] B. Reinert. Coset enumeration – a comparison of methods. Technical report, Universität Kaiserslautern.

[9] B. Reinert. *On Gröbner Bases in Monoid and Group Rings.* PhD thesis, Universität Kaiserslautern, 1995.

[10] B. Reinert, T. Mora, and K. Madlener. A note on nielsen reduction and coset enumeration. In O. Gloor, editor, *Proc. ISSAC'98*, pages 171–178. ACM, 1998.

[11] D. Wißmann. *Anwendung von Rewriting-Techniken in polyzyklischen Gruppen.* PhD thesis, Universität Kaiserslautern, 1989.

Unentscheidbarkeitsresultate für endlich dargestellte Monoide

Andrea Sattler-Klein

Fachbereich Informatik
Universität Kaiserslautern

Es ist bekannt, dass aus einer endlichen Darstellung eines Monoids, d.h. aus einem Paar $(\Sigma, \mathcal{R})$ bestehend aus einem endlichen Alphabet Σ und einem endlichen Wortersetzungssystem $\mathcal{R}$, nahezu keine Information über die algebraischen Eigenschaften des dargestellten Monoids hergeleitet werden kann. Insbesondere sind - wie ein klassisches Resultat von Markov [Ma51] zeigt - die folgenden Probleme im allgemeinen unentscheidbar:

- das *Endlichkeitsproblem*:
  EINGABE: Ein endliches WES $\mathcal{R}$ über einem endlichen Alphabet Σ.
  FRAGE: Ist das durch $(\Sigma, \mathcal{R})$ dargestellte Monoid $\mathcal{M}_{\mathcal{R}}$ endlich?

- das *Freie-Monoid-Problem*:
  EINGABE: Ein endliches WES $\mathcal{R}$ über einem endlichen Alphabet Σ.
  FRAGE: Ist das durch $(\Sigma, \mathcal{R})$ dargestellte Monoid $\mathcal{M}_{\mathcal{R}}$ frei, d.h., ist $\mathcal{M}_{\mathcal{R}}$ isomorph zu Γ^* für ein endliches Alphabet Γ?

- das *Triviale-Monoid-Problem*:
  EINGABE: Ein endliches WES $\mathcal{R}$ über einem endlichen Alphabet Σ.
  FRAGE: Ist das durch $(\Sigma, \mathcal{R})$ dargestellte Monoid $\mathcal{M}_{\mathcal{R}}$ trivial, d.h., ist $\mathcal{M}_{\mathcal{R}}$ isomorph zu $\{\varepsilon\}$?

- das *Gruppe-Problem*:
  EINGABE: Ein endliches WES $\mathcal{R}$ über einem endlichen Alphabet Σ.
  FRAGE: Ist das durch $(\Sigma, \mathcal{R})$ dargestellte Monoid $\mathcal{M}_{\mathcal{R}}$ eine Gruppe?

- das *Kommutativitätsproblem*:
  EINGABE: Ein endliches WES $\mathcal{R}$ über einem endlichen Alphabet Σ.
  FRAGE: Ist das durch $(\Sigma, \mathcal{R})$ dargestellte Monoid $\mathcal{M}_{\mathcal{R}}$ kommutativ?

Von Interesse sind daher Teilklassen von Monoiddarstellungen, für die zumindestens einige dieser Probleme entscheidbar sind. Eine solche interessante Klasse bildet die Klasse der endlichen vollständigen (konvergenten) Darstellungen: Für diese Klasse ist sogar jedes der oben aufgelisteten Probleme entscheidbar.

Da Monoide die eine endliche vollständige Darstellung erlauben entscheidbares Wortproblem haben, andererseits der Beweis von Markovs Unentscheidbarkeitsresultaten auf der Unentscheidbarkeit des Wortproblems beruht, stellt die Klasse der endlichen Darstellungen mit

entscheidbarem Wortproblem eine weitere wichtige und interessante Teilklasse der endlichen Monoiddarstellungen dar.

In diesem Vortrag wird gezeigt, dass jedoch für die Klasse der endlich dargestellten Monoide mit entscheidbarem Wortproblem jedes der zuvor aufgelisteten Probleme unentscheidbar ist (s. [Sa96, Sa97]). Insbesondere wird bewiesen, dass all diese Probleme selbst für gewisse echte Teilklassen der Klasse der endlich dargestellten Monoide mit entscheidbarem Wortproblem - wie beispielsweise für die Klasse der endlich dargestellten Monoide, deren Wortprobleme in polynomialer Zeit entscheidbar sind - unentscheidbar sind.

Von besonderer Bedeutung ist dabei die zum Nachweis dieser Verschärfung des Resultats von Markov entwickelte Beweistechnik, da mit Hilfe dieser Technik weitere interessante Unentscheidbarkeitsresultate für endlich dargestellte Monoide nachgewiesen werden konnten [Sa96, Sa97, OS97, OS99].

Literatur

[Ma51] A.A. Markov. The impossibility of algorithms for recognizing some properties of associative systems. *Doklady Akademii Nauk SSSR* 77 (1951), 953–956.

[OS97] F. Otto, A. Sattler-Klein. FDT is undecidable for finitely presented monoids with solvable word problems. In T.B.S. Chlebus, L. Czaja, editors, *Fundamentals of Computation Theory, Proceedings FCT'97*, 388–399. Springer-Verlag, Berlin, 1997. Lecture Notes in Computer Science 1279.

[OS99] F. Otto, A. Sattler-Klein. The property FDT is undecidable for finitely presented monoids that have polynomial-time decidable word problems. *Int. J. of Algebra and Computation*, to appear.

[Sa96] A. Sattler-Klein. A systematic study of infinite canonical systems generated by Knuth-Bendix completion and related problems. Dissertation, Fachbereich Informatik, Universität Kaiserslautern, Februar 1996.

[Sa97] A. Sattler-Klein. New undecidability results for finitely presented monoids. In H. Comon, editor, *Rewriting Techniques and Applications, Proceedings RTA'97*, 68–82. Springer-Verlag, Berlin, 1997. Lecture Notes in Computer Science 1232.

Beiträge zum 9. Theorietag

H-Ausdrücke – eine Erweiterung regulärer Ausdrücke

Henning Bordihn[1] und Jürgen Dassow[1] und Markus Holzer[2]

[1] Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
Postfach 4120, D-39016 Magdeburg, Germany
email: {bordihn,dassow}@iws.cs.uni-magdeburg.de
[2] Département d'I.R.O., Université de Montréal, C.P. 6128,
succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada
email: holzer@iro.umontreal.ca

H-Ausdrücke und *erweiterte H-Ausdrücke* sind regulärartige Ausdrücke, d.h. reguläre Ausdrücke erweitert um die Operationen der homomorphen Ersetzung bzw. der homomorphen Ersetzung und der iterierten homomorphen Ersetzung. Somit sind (erweiterte) H-Ausdrücke eine natürliche Variante der regulärartigen Ausdrücke mit (iterierter) Substitution, die von Gruska eingeführt wurden, um die kontextfreien Sprachen zu charakterisieren [2].

Die Operation der homomorphen Ersetzung wurde von Albert und Wegner untersucht [1] und taucht in der Literatur unter verschiedenen Namen auf, z.B. als „konsistente Substitution" in van Wijngaarden Grammatiken, als „inside-out (IO) substitution" in Makro-Grammatiken oder als „call by value substitution" in algebraischen Ansätzen in der Theorie formaler Sprachen.

Wir untersuchen die Beschreibungskapazität der H-Ausdrücke und der erweiterten H-Ausdrücke, vergleichen sie mit den Klassen der Chomsky-Hierarchie und insbesondere mit der Kapaziät der von Albert und Wegner eingeführten homomorphen Ersetzungssysteme [1], die eine Verallgemeinerung der Pattern- und Multipattern-Sprachen darstellen. Ferner charakterisieren wir die durch H-Ausdrücke beschriebenen Sprachen durch eine iterierte Variante der homomorphen Ersetzungssysteme. Schließlich untersuchen wir die Entscheidbarkeit und Komplexität der folgenden Probleme für H-Ausdrücke und erweiterte H-Ausdrücke: fixed membership, general membership, Äquivalenz und Leerheit. In vielen Fällen lässt sich Vollständigkeit für bekannte Komplexitätsklassen feststellen.

Literatur

[1] J. Albert and L. Wegner. Languages with homomorphic replacements. In *Proceedings of the 7th International Colloquium on Automata Languages and Programming*, number 85 in LNCS, pages 19–29. Springer, July 1980.

[2] J. Gruska. A characterization of context-free languages. *Journal of Computer and System Sciences*, 5:353–364, 1971.

XML, eine Verwandte der Dyck-Sprachen

Ausführliche Zusammenfassung

Ralf Behrens[1] und Gerhard Buntrock[2]

[1] Institut für Informationssysteme
Medizinische Universität zu Lübeck
[2] Institut für Informatik
Universität Gießen
und
Institut für Theoretische Informatik
Medizinische Universität zu Lübeck

1 Einführung

Das W3C-Konsortium legt mit der Extensible Markup Language (XML) eine Sprachdefinition zur Dokumentenbeschreibung vor [BPSM98]. Es ist eine auf Anwendungsfälle zugeschnittene Einschränkung von SGML, die sich zunehmender Beliebtheit erfreut. XML Dokumente bestehen aus einem grammatikalischen Teil — der sogenannten *Dokument Type Definition* (DTD), auf deutsch „Dokumententypdefinition" — und einer Dokumenteninstanz.

In der Regel werden heute Texte und andere Informationen mit verschiedenen Zusatzinformationen gespickt: dem „Markup". Hierfür hat sich die eingedeutschte Sprechweise „taggen" eingebürgert. Das heißt, Informationen der Art, „hier steht eine Überschrift" und „hier steht ein Absatz" und „hier steht ein Name" wird im Text mit Klammern der Typen Überschrift, Absatz und Name dargestellt, die den Anfang und das Ende dieser Passagen jeweils anzeigen.

Eine schon seit der Nutzung des Internets aufgekommene mittlerweile sehr bekannte Markup-Sprache ist HTML: *Hypertext Markup Language* [RHB98]. Auch Textsysteme wie LaTeX enthalten in gewisser Weise ein Markup. Allerdings wird hier in der Regel nur an eine einzige Anwendung, nämlich das Anzeigen bzw. Drucken gedacht. Für XML ist eine Formatierungssprache in der Entwicklung: *Extensible Stylesheet Language* (XSL) [ABC+99]. Diese gestattet dann je nach Anwendungsfall gewisse Informationen auszublenden oder nur auf konkrete explizite Nachfrage anzuzeigen oder auch nur die Betonung (etwa Größe, Farbe oder Form) der einzelnen Passgagen den verschiedenen Typen und Anwendungsfällen anzupassen.

Auf diese Weise können wir z. B. als konkrete Anwendung aus einer Personaldatei einerseits die postalischen Anschriften andererseits aber auch Telefonnummern oder andere Informationen mit entsprechenden XSL-Tools herauslesen.

2 Dyck-Sprachen und wohlgeformte Dokumente

XML-Dokumente bestehen stets aus einer DTD und einer Dokumenteninstanz. Passt die Dokumenteninstanz zur DTD, so heißt die Dokumenteninstanz *gültig für die* DTD. Das W3C-

Konsortium hebt aus beliebigen Dokumentinstanzen die *wohlgeformten Dokumentinstanzen* (wellformed documents) heraus, das sind korrekt geklammerte Dokumentinstanzen ohne Betrachtung von grammatikalischen Einschränkungen bezüglich einer DTD.

Wir betrachten zunächst solche wohlgeformten Dokumente. Wir bezeichnen mit $\Sigma_k =_{\text{def}} \{(_1,)_1,(_2,)_2,\dots(_k,)_k\}$, die Menge der Klammersymbole, auch *Tags* genannt und mit Δ die Menge der Datensymbole. Hierbei ist $\Sigma_k \cap \Delta = \emptyset$ festgelegt. Genaugenommen besteht Σ aus Wörtern, die in „<“ und „>“ eingeschlossen werden bzw. in „</“ und „>“. Somit hat beispielsweise ein Klammerpaar die Gestalt „<Überschrift>“, „</Überschrift>“. Damit ist gewährleistet, dass wir eine unbeschränkte Anzahl von Klammerpaaren zur Verfügung haben.

Die wohlgeformten Dokumente definieren wir auf folgende Weise für jedes $k \in \mathbb{N}$:

$$\begin{array}{lll} \mathsf{W}_k & =_{\text{def}} & \{w \in (\Sigma_k \cup \Delta)^* \mid \exists j \in \mathbb{N} : \exists v \in \mathsf{W}' : \ 1 \le j \le k, w = (_j v)_j \ \} \\ v \in \mathsf{W}' & \Longleftrightarrow_{\text{def}} & v \in \Delta^* \\ & & \vee \ \exists v_1 \in \mathsf{W}', \exists i \in \mathbb{N} : \ v = (_i v_1)_i \\ & & \vee \ \exists v_1, v_2 \in \mathsf{W}' : \ v = v_1 v_2 \end{array}$$

Aus der Theorie der Formalen Sprachen kennen wir die Familie der Dycksprachen (siehe z. B. [Gru97]): Dazu betrachten wir zunächst die folgenden Regeln

$$R_{\text{DYCK}} =_{\text{def}} \{(S \to \varepsilon), (s \to S[_i S]_i) \mid 1 \le i \le k\}$$

Die Dycksprachen bestehen dann aus den aus R_{DYCK} ableitbaren Wörtern:

$$\text{DYCK}_k =_{\text{def}} \{w \in \{[_1,]_1,[_2,]_2,\dots,[_k,]_k\}^* \mid S \underset{R}{\Longrightarrow} w\}$$

Somit lassen sich die wohlgeformten Dokumente mit Dycksprachen darstellen, indem wir an beliebigen Stellen Wörter aus Δ^* zulassen und stets mit einer öffnenden Klammer beginnen, die zu der das Dokument schließenden Klammer gehört. Umgekehrt ist der Fall, dass jede Dycksprache mit einem zusätzlichen Klammerpaar zu einem wohlgeformten Dokument werden kann. So können wir folgendes feststellen:

> Den löschenden Homomorphismus $h_1 : \Delta \to \{\varepsilon\}$ und den Isomorphismus $h_2 : \Sigma \to \{[_1,]_1,[_2,]_2,\dots,[_k,]_k\}$ mit $h_2((_i) = [_i$ und $h_2()_i) =]_i$ fassen wir mit dem Homomorphismus $h : \Delta \cup \Sigma \to \{[_1,]_1,[_2,]_2,\dots,[_k,]_k\}$ in offensichtlicher Weise zusammen. Dann gilt: Für jedes $k \in \mathbb{N}$: $\mathsf{W}_k = h_2^{-1}([_1)h^{-1}(\text{DYCK}_k)h_2^{-1}(]_1)$.

3 Dokumententypdefinitionen und deterministisch kontextfreie Sprachen

Eine DTD legt eine Menge von Dokumenten fest, die von passender Struktur sind. Ein Parser für XML-Dokumente prüft daher, ob die DTD die Struktur der Dokumenteninstanz beschreibt. Mit obigem Satz und der Tatsache, dass jede DTD eine kontextfreie Grammatik repräsentiert, ergibt sich, dass die Menge der Dokumentinstanzen zu einer DTD eine deterministisch kontextfreie Sprache ist. Damit ist auch deren Parsierbarkeit in linearer Zeit garantiert. Andererseits können wir jede kontextfreie Sprache mit einer DTD beschreiben: Wir zeigen dazu, der Abschluss unter löschenden Homomorphismen der Menge der mit DTDs beschreibbaren Dokumentinstanzen ergibt die Menge der kontextfreien Sprachen. Dazu nehme man einen Homomorphismus, der alle Zeichen aus Δ und alle schließenden Klammern löscht.

In der Praxis sind wir an verschiedenen Abschlusseigenschaften interessiert. Wir untersuchen daher, ob die Klasse der Sprachen, für die es jeweils eine DTD gibt, für die sie gültig ist, unter verschiedenen klassischen Abschlusseigenschaften abgeschlossen ist. Oft sind für solche Eigenschaften kleine definitorische Schwierigkeiten zu überwinden. Siehe dazu auch den Report [BBLL99].

Literatur

[ABC+99] Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Alex Milowski, and Steve Zilles. Extesible Stylesheet Language (XSL) Specification. W3C recommendation, `http://www.w3.org/TR/WD-xsl`, April 1999.

[BBLL99] Ralf Behrens, Gerhard Buntrock, Carsten Lecon, and Volker Linnemann. Is XML really enough? Technical report, Medizinische Universität zu Lübeck und Universität Gießen, July 1999.

[BPSM98] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C recommendation, `http://www.w3.org/TR/REC-xml`, February 1998.

[Gru97] Jozef Gruska. *Foundations of Computing*. Thomson Computer Press, 1997.

[RHB98] Dave Ragget, Arnaud Le Hors, and Bert Bos. HTML 4.0 Specification. W3C recommendation, `http://www.w3.org/TR/REC-html40`, April 1998.

Collagengrammatiken und iterierte Funktionensysteme mit Farboperationen

Frank Drewes

Fachbereich 3 – Mathematik/Informatik
Universität Bremen

Die Art und Weise, in der z.B. iterierte Funktionensysteme (IFS) und Collagengrammatiken Bilder erzeugen, kann durch Systeme beschrieben werden, die Bäume (= Terme) erzeugen, welche dann als Ausdrücke interpretiert werden, die Bilder liefern. Etwas formaler heißt das, es wird eine Σ-Algebra $\mathcal{P}$ betrachtet, deren Elemente Bilder sind (wobei es zunächst einmal nicht nötig ist, den Begriff „Bild" näher zu definieren), und eine Grammatik G beliebiger Art, die eine Sprache $L(G) \subseteq T_\Sigma$ von Bäumen über Σ erzeugt. Da der Wert $val_\mathcal{P}(t)$ eines Baumes in $\mathcal{P}$ ein Bild ist, definiert G eine Sprache $L_\mathcal{P}(G) = \{val_\mathcal{P}(t) \mid t \in L(G)\}$ von Bildern.

Üblicherweise ist ein Bild eine Teilmenge von $\mathbb{R}^d$, meistens $\mathbb{R}^2$. Um zu IFS oder Collagengrammatiken äquivalente Systeme zu bekommen, betrachtet man Algebren $\mathcal{P}$, deren Operationen $\langle f_1 \cdots f_n \rangle$ aus n affinen Transformationen $f_1, \ldots, f_n$ bestehen. Die n-stellige Operation $\langle f_1 \cdots f_n \rangle$ ist dann definiert durch

$$\langle f_1 \cdots f_n \rangle(p_1, \ldots, p_n) = f_1(p_1) \cup \cdots \cup f_n(p_n).$$

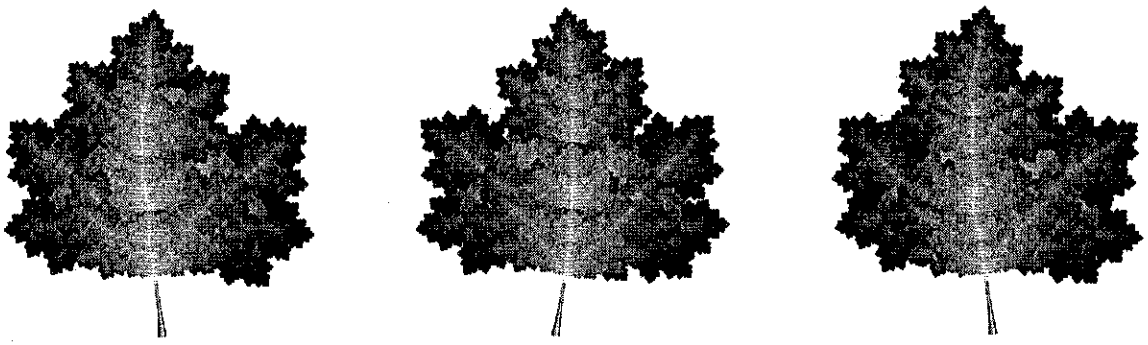
(Auf die Sorte von Grammatiken, die benötigt wird, um auf Basis dieser Operationen zu IFS bzw. Collagengrammatiken zu gelangen, soll in dieser Zusammenfassung nicht eingegangen werden.)

Diese Darstellung bildgenerierender Systeme ist gut für Variationen und Verallgemeinerungen geeignet, wie das Beispiel des Generierens farbiger Bilder zeigt. Hierzu ist grundsätzlich nichts weiter zu tun, als zu $\mathcal{P}$ Operationen hinzuzufügen, die die Farbe eines Bildes zu beeinflussen erlauben. Allerdings muss dazu natürlich der verwendete Begriff des Bildes um Farbe erweitert werden: Farbige Bilder sind Abbildungen $p\colon \overline{p} \to F$, wobei $\overline{p} \subseteq \mathbb{R}^d$ das unterliegende Bild und F die Menge der Farben ist. Zum Beispiel kann $F = [0,1]^3$ gewählt werden, wobei $p(x) = (r, g, b)$ als RGB-Wert des Punktes $x \in \overline{p}$ interpretiert wird. Meist reicht es aber aus, die Diskussion anhand von Grauwerten zu führen, d.h. $F = [0,1]$ zu betrachten.

Eine verbleibende Schwierigkeit ist, die Operationen der Form $\langle f_1 \cdots f_n \rangle$, die ja weiterhin benötigt werden, dem neuen Begriff von Bildern anzupassen. Das Problem ist dabei, dass die Farbe von Punkten im Schnitt mehrerer der Teilbilder $f_i(\overline{p_i})$ eindeutig definiert werden muss. Dies kann man tun, indem man eine Mischfunktion $\mu\colon F^+ \to F$ betrachtet, die $n \geq 1$ Farben $c_1, \ldots, c_n$ zu einer Farbe $\mu(c_1 \cdots c_n)$ „vermischt".

Besonders dann, wenn man an der Erzeugung farbiger *Fraktale* interessiert ist, stellen sich interessante Fragen: Welche Bedingungen müssen erfüllt sein, damit unendliche Bäume (denn diese beschreiben Fraktale) einen wohldefinierten Wert liefern? Approximieren die durch endliche Bäume beschriebenen Bilder die Fraktale? Wann ist die Färbung eines Fraktals stetig? Gibt es dafür Entscheidungsalgorithmen?

Erste Ergebnisse bezüglich dieser Fragen sollen in dem Vortrag dargestellt werden.



Algebraische Repräsentation regulärer Array-Sprachen

Rudolf Freund

Institut für Computersprachen
Technische Universität Wien

Die Verallgemeinerung eindimensionaler Wörter führt zu d-dimensionalen Arrays, d.h. Funktionen von Z^d in ein endliches Alphabet. In den letzten beiden Jahrzehnten wurden neue theoretische Modelle von generierenden und akzeptierenden Mechanismen entwickelt, welche nicht nur interessante theoretische Resultate (s. [3], [8]) erlaubten, sondern auch die theoretische Grundlage für bestimmte Anwendungen, z.B. in der Mustererkennung (s. [2], [6]), darstellten.

Endliche Folgen von d-dimensionalen Vektoren und Buchstaben (im Folgenden als d-dimensionale formale Arrays bezeichnet) können als d-dimensionale Arrays interpretiert werden. Im Gegensatz zu d-dimensionalen Arrays selbst können für diese d-dimensionalen formalen Arrays die Operationen Konkatenation und iterierte Konkatenation (Stern) definiert werden, da jeder formale Array eine wohldefinierte Anfangsposition und eine wohldefinierte Endposition besitzt. Zusammen mit der Operation Vereinigung (von Mengen formaler Arrays) gelingt nun die algebraische Repräsentation bestimmter regulärer Array-Sprachen, also die Charakterisierung von Array-Sprachen, die von spezifischen regulären d-dimensionalen Array-Grammatiken erzeugt werden, durch entsprechende reguläre (d-dimensionale Array-)Ausdrücke. Überdies kann jeder dieser spezifischen d-dimensionalen Array-Grammatiken ein rationales Gleichungssystem (vgl. [5]) derart zugeordnet werden, dass die erzeugte reguläre d-dimensionale Array-Sprache eine Komponente der minimalen Fixpunktlösung dieses Gleichungssystems ist.

Diese in [4] bewiesene Charakterisierung durch reguläre d-dimensionale Array-Ausdrücke gelingt nur für reguläre d-dimensionale Array-Sprachen, die von sogenannten maximalen regulären d-dimensionalen Array-Grammatiken erzeugt werden, bei denen zu jeder regulären d-dimensionalen Array-Produktion in der Array-Grammatik entsprechende reguläre d-dimensionale Array-Produktionen in alle in der Array-Grammatik vorkommenden Richtungen enthalten sind. Im eindimensionalen Fall erhält man mit maximalen regulären Array-Grammatiken die auch sonst in der Literatur behandelten regulären eindimensionalen Array-Sprachen, welche überdies den regulären (Wort-)Sprachen entsprechen. Für $d \geq 2$ hingegen können maximale reguläre d-dimensionale Array-Grammatiken, die eine algebraische Repräsentation der erzeugten Array-Sprachen erlauben, nur einen Bruchteil aller regulären d-dimensionalen Array-Sprachen darstellen. Reguläre d-dimensionale Array-Grammatiken besitzen nämlich für $d \geq 2$ eine relativ große Erzeugungskraft (s. [7]); überdies sind Leerheit und Endlichkeit der von einer regulären d-dimensionalen Array-Grammatik erzeugten Array-Sprache im Allgemeinen unentscheidbar, für maximale reguläre d-dimensionale Array-Grammatiken aber entscheidbar.

Das Modell d-dimensionaler formaler Arrays und der entsprechenden Konkatenationsoperationen kann auf allgemeinere Strukturen als Z^d erweitert werden, beispielweise kann man Cayley-Graphen endlich erzeugter Gruppen als zugrundeliegende Strukturen für die Arrays betrachten (s. [1]).

Literatur

[1] E. Csuhaj-Varjú, R. Freund, V. Mitrana, *Arrays on Cayley grids, TUCS Report,* 1998.

[2] H. Fernau, R Freund, M. Holzer, *Character recognition with k-head finite array automata,* in: A. Amin, D. Dori, P. Pudil, H. Freeman (eds.), *Advances in Pattern Recognition,* Springer 1998, pp. 282–291.

[3] R. Freund, *Array grammar systems,* in: A. Kelemenova (ed.), *Proceedings of the MFCS'98 Satellite Workshop on Grammar Systems,* Silesian University of Opava, 1998, pp. 91–116.

[4] R. Freund, A. Mateescu, A. Salomaa, *Algebraic representation of regular array languages, to appear.*

[5] W. Kuich, A. Salomaa, *Semirings, Automata, Languages* (Springer-Verlag, Berlin, 1986).

[6] A. Rosenfeld, *Picture Languages* (Academic Press, Reading, MA, 1979).

[7] Y. Yamamoto, K. Morita, K. Sugata, *Context-sensitivity of two-dimensional regular array grammars,* in: [8].

[8] P.S.-P. Wang (ed.), *Array Grammars, Patterns and Recognizers,* WSP Series in Computer Science, Vol. 18 (World Scientific Publ., Singapore, 1989), pp. 17–41.

On the Generative Power of Graph Grammars with Injective Matching

Annegret Habel

Fachbereich Informatik
Universität Oldenburg

1 Introduction

Graph grammars (see [Ehr79]) may be seen a generalization of string grammars. A graph replacement rule consists of two graphs, the left-hand side L and the ride-hand side R, together with an interface graph K and morphisms $L \leftarrow K$ and $K \rightarrow R$. Such a rule can be applied to a graph G by searching for an occurrence of L in G, checking a suitable application condition, and replacing the occurrence of L by R. The occurrence of L in G is fixed by a graph morphism $g : L \rightarrow G$. If the graph morphism is required to be injective, we speak on graph grammars with injective matching.
In this talk, we investigate and compare four variants of graph grammars. Besides graph grammars with arbitrary matching and injective right-hand morphisms, we consider three variations by employing injective matching and/or arbitrary right-hand morphisms in rules. We show that injective matching provides additional expressiveness in one respect: for generating graph languages by grammars without nonterminals. For more details see [HMP99a].

2 Preliminaries

A *graph replacement rule* $p = \langle L \leftarrow K \rightarrow R \rangle$ consists of two graph morphisms with a common domain. L is the *left-hand side*, R the *right-hand side*, and K the *interface* of p. We throughout assume that $K \rightarrow L$ is an inclusion. Such a rule is *injective* if $K \rightarrow R$ is injective. Given two graphs G and H, G *directly derives* H *through* p, denoted by $G \Rightarrow_p H$, if the diagrams (1) and (2) below are graph pushouts.

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ \downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \longleftarrow & D & \longrightarrow & H \end{array}$$

The notation $G \Rightarrow_{p,g} H$ is used when $g: L \rightarrow G$ shall be made explicit.
A direct derivation $G \Rightarrow_{p,g} H$ is said to be in
- $\mathrm{DPO}^{\mathrm{i/a}}$ if p is injective and g is arbitrary (the "traditional approach"),
- $\mathrm{DPO}^{\mathrm{a/a}}$ if p and g are arbitrary,
- $\mathrm{DPO}^{\mathrm{i/i}}$ if p and g are injective,

– $\mathrm{DPO}^{a/i}$ if p is arbitrary and g is injective.
A *graph grammar in* $\mathrm{DPO}^{x/y}$ $(x, y \in \{a, i\})$ is a system $\mathcal{G} = \langle \mathcal{C}, \mathcal{R}, S, \mathcal{T} \rangle$, where $\mathcal{C}$ is a finite label alphabet, $\mathcal{T}$ is an alphabet of *terminal* labels with $\mathcal{T}_V \subseteq \mathcal{C}_V$ and $\mathcal{T}_E \subseteq \mathcal{C}_E$, S is a graph over $\mathcal{C}$ called the *start graph*, and $\mathcal{R}$ is a finite set of rules over $\mathcal{C}$ such that if $x = i$, all rules in $\mathcal{R}$ are injective. The *graph language generated by* $\mathcal{G}$ is the set $L(\mathcal{G})$ consisting of all graphs G over $\mathcal{T}$ such that there is a derivation $S \Rightarrow_{\mathcal{R}}^* G$ in $\mathrm{DPO}^{x/y}$. We denote by $\mathcal{L}^{x/y}$ the class of all graph languages generated by graph grammars in $\mathrm{DPO}^{x/y}$.

3 Generative Power

We investigate the expressiveness of graph grammars in $\mathrm{DPO}^{x/y}$, for $x, y \in \{a, i\}$. It turns out that all four variants have the same generative power for unrestricted grammars, but $\mathrm{DPO}^{x/i}$ is more powerful than $\mathrm{DPO}^{x/a}$ if we consider grammars without nonterminal labels.

Lemma 3.1. *Let* $x \in \{a, i\}$. *1. For every graph grammar* $\mathcal{G}$ *in* $\mathrm{DPO}^{x/a}$ *there exists a graph grammar* $\mathcal{G}'$ *in* $\mathrm{DPO}^{x/i}$ *such that* $L(\mathcal{G}) = L(\mathcal{G}')$. *2. For every graph grammar* $\mathcal{G}$ *in* $\mathrm{DPO}^{x/i}$ *there exists a graph grammar* $\mathcal{G}'$ *in* $\mathrm{DPO}^{x/a}$ *such that* $L(\mathcal{G}) = L(\mathcal{G}')$.

By Lemma 3.1 we know that injective matching does not increase the generative power of unrestricted graph grammars. The next lemma shows that the same applies to non-injective rules.

Lemma 3.2. *For every graph grammar* $\mathcal{G}$ *in* $\mathrm{DPO}^{a/i}$ *there exists a graph grammar* $\mathcal{G}'$ *in* $\mathrm{DPO}^{i/i}$ *such that* $L(\mathcal{G}) = L(\mathcal{G}')$.

Proof. Let $\mathcal{G}$ be a graph grammar in $\mathrm{DPO}^{a/i}$. We construct a grammar $\mathcal{G}'$ in $\mathrm{DPO}^{i/i}$ that simulates the identification of two edges by deleting one of the edges, and the identification of two nodes by redirecting all edges incident with one node to the other one and by deleting the first one. □

An alternative proof of Lemma 3.2 can be obtained from Uesu's result that $\mathcal{L}^{i/i}$ is the class of all recursively enumerable sets of graphs [Ues78]. (A set of graphs over some label alphabet is recursively enumerable if its image under a Gödel numbering is a recursively enumerable set of natural numbers.) This is because each graph language in $\mathcal{L}^{a/i}$ is clearly recursively enumerable and hence $\mathcal{L}^{a/i} \subseteq \mathcal{L}^{i/i}$. However, our proof shows in addition how to transform an arbitrary grammar in $\mathrm{DPO}^{a/i}$ into an equivalent grammar in $\mathrm{DPO}^{i/i}$.

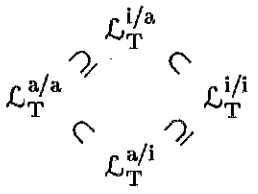
Theorem 3.3 (Generative power I). $\mathcal{L}^{i/a} = \mathcal{L}^{i/i} = \mathcal{L}^{a/a} = \mathcal{L}^{a/i}$.

Injective matching provides more generative power if we restrict ourselves to grammars without nonterminal labels. To this end, we denote by $\mathcal{L}_T^{x/y}$ the class of all graph languages generated by a grammar $\langle \mathcal{C}, \mathcal{R}, S, \mathcal{T} \rangle$ in $\mathrm{DPO}^{x/y}$ with $\mathcal{T} = \mathcal{C}$. Note that for such a grammar, every graph derivable from S belongs to the generated language.

Theorem 3.4 (Generative power II). *For* $x \in \{a, i\}$, $\mathcal{L}_T^{x/a} \subset \mathcal{L}_T^{x/i}$.

Proof. By the proof of Lemma 3.1, for every grammar $\mathcal{G}$ in $\mathrm{DPO}^{x/a}$ without nonterminal labels there is a grammar $\mathcal{G}'$ in $\mathrm{DPO}^{x/i}$ without nonterminal labels such that $L(\mathcal{G}) = L(\mathcal{G}')$. Hence $\mathcal{L}_T^{x/a} \subseteq \mathcal{L}_T^{x/i}$. To show that the inclusion is strict, let $\mathcal{G}$ be the grammar in $\mathrm{DPO}^{i/i}$ that generates the set of all loop-free graphs over $\mathcal{C}$. It turns out that there is no grammar in $\mathrm{DPO}^{a/a}$ without nonterminal labels can generate $L(\mathcal{G})$. □

By Theorem 3.4 and the fact that $\mathrm{DPO}^{i/y}$ is contained in $\mathrm{DPO}^{a/y}$, for $y \in \{a, i\}$, we have the following relations between the four classes of graph languages generated by grammars without nonterminal labels:



It is open whether or not the inclusions $\mathcal{L}^{i/a} \subseteq \mathcal{L}_T^{a/a}$ and $\mathcal{L}_T^{i/i} \subseteq \mathcal{L}_T^{a/i}$ are also strict.

References

[Ehr79] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In Proc. Graph-Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science 73, 1–69, 1979.

[HMP99a] Annegret Habel, Jürgen Müller, Detlef Plump. Double-pushout approach with injective matching. In Proc. Theory and Application of Graph Transformation, Lecture Notes in Computer Science, 1999. To appear.

[Ues78] Tadahiro Uesu. A system of graph grammars which generates all recursively enumerable sets of labelled graphs. Tsukuba J. Math. 2, 11–26, 1978.

XSSR: An Experimental System for String-Rewriting — Decision Problems, Algorithms, and Implementation

D. Hofbauer[1], C. Kögl[2], K. Madlener[2], F. Otto[1], and B. Reinert[2]

[1] Universität Kassel, Fachbereich Mathematik/Informatik
[2] Universität Kaiserslautern, Fachbereich Informatik

1 Intended functionality

XSSR is an experimental system for string-rewriting[1]. It is intended for computations with monoids and groups that are given in terms of finite presentations. It incorporates various algorithms for deciding properties of monoids, submonoids, and elements of monoids and groups that are based on rewriting methods. In [3] a detailed description is given of the algorithms that are to be implemented in XSSR.

A monoid or a group is described internally as a collection of finite presentations. These presentations are either entered by the user, or they are generated from given presentations through the use of the operations provided by the system. These operations include the Tietze transformations and various Knuth-Bendix style procedures for completion, λ-completion and weak completion. Also some algebraic operations like free or direct products will be available to construct new monoids or groups from given ones. However, in order to ensure that the database of presentations is free from contradictions, only presentations generated by the system can be added to the collection of presentations describing a certain monoid. If the user wants to add another presentation of the same monoid or group, then this must be done in such a way that the system can verify that the new presentation does indeed present the same object. In particular, the system determines and stores the morphisms that realize the isomorphisms between the various presentations of the same monoid or group.

Investigating a specific monoid or group the user can ask questions about properties of elements, subsets, or the monoid or group as such. For example, he can ask whether the monoid considered is actually a group. If the information is already available, or if an algorithm for answering this question is known for one of the presentations listed for the monoid or group considered, the system will provide an answer immediately; otherwise, it will state that the answer is not yet known, and it will suggest ways of trying to find the answer. The different classes of presentations are organized as a hierarchy. However, since certain classes are incomparable under set inclusion, this hierarchy is not a linear one.

Using XSSR it will be possible to run various rewrite-based algorithms described in the literature for non-trivial examples, thus obtaining data on their behaviour in practice. This

[1]Supported as part of the DFG project *Konvergente Reduktionssysteme für Monoide und Gruppen: Algebraische Charakterisierungen und Vervollständigungsverfahren.*

information will help to devise strategies for improving these algorithms. Also XSSR will provide an opportunity to try out new ideas for attacking open problems, for example concerning strategies of how to introduce new generators during the completion process.

There are various options for extending the functionality of XSSR. A presentation of a group G together with a finite set of strings in the generators of this presentation describes a subgroup H of G. Since H is a group in itself, one may be interested in obtaining a (group) presentation of H that is from the same syntactic class of presentations as the one for G [5]. Another option is the computation of Gröbner bases in monoid and group rings [6]. Finally, XSSR will provide interfaces to other systems, for example to HOMER, which computes homology groups of monoids that are given through finite convergent presentations [2].

2 Object-oriented design

The design of a software system capable of providing the described kind of assistance to the monoid and group theory practitioner is a complex task for several reasons:

- The application area is reasonably well understood in terms of the theoretical underpinnings, but it is much less clear what the most common or typical 'business cases' are that have to be supported by the software.

- It was clear right from the beginning of the project that the resulting software system would be rather large. In order not to fall prey to common project risks like lack of an architecture, difficult maintainability and extensibility, and low performance, we tried to apply recent software design techniques that facilitate the design and implementation of more robust, extensible, and maintainable systems [4, 1].

- In order to be able to solve non-trivial problems in monoid and group theory the underlying algorithms should be implemented to work as efficiently (in terms of time but also space complexity) as possible. This requirement is usually in conflict with the wish for easy extensibility. Knuth-Bendix-type completion procedures, which are needed as a crucial preprocessing step for virtually all of the algorithms of [3], are the most prominent example in our system. There is a host of strategies available for actually implementing the completion procedure, and even for a fixed implementation there is usually a set of client-configurable parameters that influence the performance and termination behaviour of a completion run profoundly. The user would also like the software to take advantage of additional knowledge that may speed up the completion process. Examples are the existence of inverse letters or the fact that we deal with a special string-rewriting system. We attempt to reconcile the seemingly conflicting requirements of high efficiency and easy extensibility by formulating a generic completion procedure (or a small set of such procedures) that provides several plug-in points with clearly documented interfaces that correspond to 'natural' subtasks of the completion procedure. Developers can then implement components that conform to the interfaces and easily integrate them into the system (the normalization facility or the facility for dealing with critical pairs are subtasks that come to mind).

For the design and the implementation we employ object-oriented techniques. This approach seems to be well-suited for computer algebra systems in general, and the abstractions of our application area readily map onto classes and objects. As implementation language we

chose C++ because it supports the implementation of almost all of the object-oriented modeling techniques that are used today and (if employed with care and experience) allows the production of efficient code. C++ also provides powerful facilities for generic programming (in the form of class and function templates). There is one caveat, though: C++ is a very complex language and it takes dedication and experience (and the right literature) to profit rather than to suffer from its use.

Currently, the areas that are most actively worked on are the various completion procedures; the component for handling a monoid's or group's collection of presentations and for selecting the 'best' algorithm for a client request (if there is more than one algorithm available); the automata and language package, which is central to most of the non-obvious algorithms and some of the specialized completion procedures; and the design of user interface components that are to be kept entirely separate from the application kernel.

To our dismay the freely available automata packages do neither match our functional requirements nor do they satisfy our need for clearly documented and readily usable plug-in components. So we are forced to design a package from scratch, but we hope to at least profit from the implementation experiences that the authors of available packages have documented, see [7, 8, 9], among others.

References

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-Oriented Software Architecture.* Wiley, 1996.

[2] M. Buchheit. *Algorithmen zur Berechnung von Invarianten für konvergente Wortersetzungs-Systeme.* Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, 1991.

[3] R. Cremanns. *Decision problems for string-rewriting systems.* Mathematische Schriften Kassel 10/97, Universität Kassel, 1997.
`http://www.db.informatik.uni-kassel.de/FG_TH/projekte/proj10.html`

[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns.* Addison-Wesley, 1995.

[5] K. Madlener and F. Otto. *Some Applications Of Prefix-Rewriting In Monoids, Groups, And Rings.* Reports On Computer Algebra, No. 22, Centre for Computer Algebra, University of Kaiserslautern, 1998. `http://www.mathematik.uni-kl.de/~zca`.

[6] K. Madlener and R. Reinert. String rewriting and Gröbner bases – a general approach to monoid and group rings. In *Proc. Workshop on Symbolic Rewriting Techniques, Monte Verita, 1995*, pp. 127–180. Birkhäuser, 1998.

[7] O. Matz, A. Miller, A. Potthoff, W. Thomas, E. Valkema. Report on the Program AMoRE. *Bericht Nr. 9507*, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, 1995.

[8] D. R. Raymond and D. Wood. Grail: A C++ Library for Automata and Expressions. *J. Symb. Computation*, 17:341–350, 1994. See `http://www.csd.uwo.ca/research/grail/`

[9] B.W. Watson. FIRE Lite: FAs and REs in C++. In D. Raymond, D. Wood, and S. Yu, eds., *Proc. 1st Workshop on Implementing Automata, (London, Ontario, Canada, 1996)*, LNCS Vol. 1260, Springer, 1997. See `http://www.ribbitsoft.com/`

Zeithierarchiesätze für iterative Arrays

Andreas Klein

Institut für Informatik
Universität Gießen

Ein iteratives Array (IA) ist ein Modell für massiv parallele Rechner mit sequentieller Eingabe [2]. Dabei werden endliche Automaten in einer Reihe miteinander verbunden, so daß die Zustandsüberführung jedes Automaten von dem eigenen Zustand und den Zuständen seiner beiden Nachbarn abhängt. Ein Automat wird als Ursprung ausgezeichnet und kann zusätzlich die Eingabe lesen. Die Automaten arbeiten synchron zu diskreten Zeitpunkten, d.h. es wird ein globaler Taktgeber vorausgesetzt.

Ein interessantes Problem der Komplexitätstheorie ist der Beweis von Hierarchiesätzen. Mit Hilfe universeller Automaten lassen sich Hierarchiesätze der folgenden Form beweisen.

Satz 1. *Seien $t, t' : \mathbb{N} \to \mathbb{N}$ Funktionen mit $t(n) \geq n$ und $t'(n) \geq n$. Ist t zeitberechenbar und $t'(n) = o(t(n))$, so folgt:*
$DTime_k(t'(n)) \subset DTime_k(t(n))$ *für* $k \geq 2$ *(Turing–Maschinen)*
$\mathscr{L}_{t'(n)}(CS) \subset \mathscr{L}_{t(n)}(CS)$ *(Zellularräume)*
$\mathscr{L}_{t'(n)}(IA) \subset \mathscr{L}_{t(n)}(IA)$ *(iterative Arrays)*

Außerdem lassen sich leicht Beschleunigungssätze der folgenden Form zeigen.

Satz 2. *Für alle Funktionen $t : \mathbb{N} \to \mathbb{N}$ und Konstanten $k \in \mathbb{N}$ gilt*
$\mathscr{L}_{n+k(t(n))}(CS) = \mathscr{L}_{n+t(n)}(CS)$ *und* $\mathscr{L}_{n+k(t(n))}(IA) = \mathscr{L}_{n+t(n)}(IA)$.

Auf diese Weise bleibt der Fall von Zeitbeschränkungen zwischen $id(n) = n$ und $2id$ ungelöst. Bisher sind auch keine allgemeinen Methoden bekannt, um den Fall kleiner Komplexitätsschranken zu behandeln. Zum Beispiel erhält man bei Turing–Maschinen mit Hilfe von Crossing–Sequenzen

$$DTime_1(LIN) = DTime_1(o(n \log(n))) = \mathcal{REG},$$

während das Problem, ob

$$\mathscr{L}_{rt}(CA) = \mathscr{L}_{lt}(CA)$$

gilt, eines der hartnäckigsten ungelösten Probleme in der Theorie der parallelen Automaten ist.

Für iterative Arrays können wir eine dichte Zeithierarchie beweisen.

Satz 3. *Seien $t(n) : \mathbb{N} \to \mathbb{N}$ und $t'(n) : \mathbb{N} \to \mathbb{N}$ zwei Funktionen, die den folgenden Bedingungen genügen:*

1. $t'(n) \leq t(n)$ *für alle* $n \in \mathbb{N}$.

2. *Es gibt eine Funktion* $g : \mathbb{N} \to \mathbb{N}$ *mit* $g(n) \geq (n+1)^2$, *so daß* $t(g(n)) = \Omega(n^2)$ *und* $t'(g(n)) = o(n^2)$ *gilt.*

3. *Außerdem gebe es einen* ϵ*-freien Homomorphismus* h *und eine Sprache* $L \in \mathscr{L}_{rt}(IA)$, *so daß* $h(L) = \{a^{g(n)-n}b^n | n \in \mathbb{N}\}$ *gilt.*

Dann gilt

$$\mathscr{L}_{n+t'(n)}(IA) \subset \mathscr{L}_{n+t(n)}(IA).$$

(Bemerkung: Die Voraussetzungen 1 bis 3 lassen sich leicht erfüllen. Z.B. wird Voraussetzung 3 von $g(n) = 2^n$ und $g(n) = n^2$ erfüllt. Außerdem ist die Klasse aller Funktionen, die Voraussetzung 3 erfüllen, unter Verkettung, Addition und Multiplikation abgeschlossen.

Ein Beispiel für Funktionen, die 1 bis 3 erfüllen, ist $t(n) = \log(n)$, $t'(n) = \log\log(n)$ und $g(n) = 2^{n^2}$.)

Dieser Satz behandelt Zeitbeschränkungen $t(n) = O(n)$, während der Fall $t(n) = \omega(n)$ bereits durch den oben zitierten Satz abgedeckt ist.

Literatur

[1] T. Buchholz, A. Klein und M. Kutrib. Iterative arrays with a wee bit alternation. *FCT '99*, LNCS 1684, Springer, Berlin, 1999.

[2] S. N. Cole. Real-Time Computation by n-Dimensional Iterative Arrays of Finite-State Machines. *IEEE Transactions on Computers*, C-18, (1969), 349–365.

[3] F. C. Hennie. One-Tape, Off-Line Turing Machine Computations. *Information and Control*, 8, (1965), 553–578.

[4] B. Martin. Efficient Unidimensional Universal Cellular Automaton. *MFCS '92*, LNCS 629, Springer, Berlin, 1992, 374–382.

Rationale Teilbarkeitsmonoide

Dietrich Kuske

Institut für Algebra
TU Dresden

In the literature, Kleene's theorem on recognizable languages of finite words has been generalized in several directions, e.g. to formal power series by Schützenberger, to infinite words by Büchi, and to infinite trees by Rabin. More recently, Sakarovitch [Sak87] investigated rational monoids, in which the recognizable languages coincide with the rational ones. Also, several authors investigated recognizable languages in trace monoids (free partially commutative monoids) which generalize free monoids. It is known that here the recognizable languages only form a proper subclass of the rational languages, but a precise description of them using c-rational expressions could be given by Ochmański [Och85]. A further generalization of Kleene's and Ochmański's results to concurrency monoids was given by Droste. In [DK99], we consider divisibility monoids and extended Ochmański's result to these monoids, i.e. we show that a set in a "labeled divisibility monoid with finite commutation behavior" is recognizable iff it is c-rational. This result was presented at the last Theorietag in Riveris.

Thus, two main directions of generalization of Kleene's Theorem in monoids are represented by Sakarovitch's rational monoids and by trace monoids. Since the only trace monoids that satisfy Kleene's Theorem are free monoids, these two directions are "orthogonal", i.e. the intersection of the classes of monoids in consideration is the set of free monoids. In [DK99] we already remarked that there are rational divisibility monoids which are not free. Thus, our further extension of Ochmański's result to divisibility monoids is not "orthogonal" any more. Our main result concerning the relation between rational and divisibility monoids is the following (for precise definitions of undefined terms see [DK99] or [Kus99]):

Theorem 1 *Let $(M, \cdot, 1)$ be a divisibility monoid with finite commutation behavior. Then M is width-bounded $\iff$ M is rational $\iff$ M satisfies Kleene's Theorem.*

The proof uses a Foata Normal Form for the elements of a divisibility monoid, order theoretic considerations and Ramsey's Theorem.

As remarked earlier, divisibility monoids generalize trace and concurrency monoids. These two classes of monoids are defined via a finite presentation (using a set of letters Σ together with a dependence relation on Σ in the case of trace monoids and using an automaton with concurrency relations in the case of concurrency monoids). Later, algebraic properties where discovered that characterize trace monoids (up to isomorphism) [Dub86]. Differently, divisibility monoids are defined in the language of monoids, i.e. via their algebraic properties. We explicitly describe a type of finite presentation that gives rise precisely to divisibility monoids:

Theorem 2 *Let T be a finite set and E a set of equations of the form $ab = cd$ with $a, b, c, d \in T$. Let $\sim$ be the least congruence on T^* containing E. Then $M := T^*/\sim$ is a*

divisibility monoid if and only if (i)-(iii) hold for any $a, b, c, b', c' \in T$*:*

(i) $(\downarrow(a \cdot b \cdot c), \leq)$ *is a distributive lattice,*

(ii) $a \cdot b \cdot c = a \cdot b' \cdot c'$ *or* $b \cdot c \cdot a = b' \cdot c' \cdot a$ *implies* $b \cdot c = b' \cdot c'$*, and*

(iii) $a \cdot b = a' \cdot b'$, $a \cdot c = a' \cdot c'$ *and* $a \neq a'$ *imply* $b = c$.

Furthermore, each divisibility monoid arises this way.

The complete paper [Kus99], containing full proofs, can be obtained via the URL `http://www.math.tu-dresden.de/~kuske`.

References

[DK99] M. Droste and D. Kuske. On recognizable languages in divisibility monoids. In G. Ciobanu and Gh. Paun, editors, *FCT99*, Lecture Notes in Comp. Science vol. 1684, page ?? Springer, 1999.

[Dub86] C. Duboc. Commutations dans les monoïdes libres: un cadre théorique pour l'étude du parallelisme. Thèse, Faculté des Sciences de l'Université de Rouen, 1986.

[Kus99] D. Kuske. On rational and on left divisibility monoids. Technical Report MATH-AL-3-1999, TU Dresden, 1999.

[Och85] E. Ochmański. Regular behaviour of concurrent systems. *Bull. Europ. Assoc. for Theor. Comp. Science*, 27:56–67, 1985.

[Sak87] J. Sakarovitch. Easy multiplications. I. The realm of Kleene's Theorem. *Information and Computation*, 74:173–197, 1987.

Kontextfreie Sprachen und Polyautomaten

Martin Kutrib

Institut für Informatik
Universität Gießen

Der von Smith 1976 [5] geprägte Begriff „Polyautomaten" faßt unter anderem die Klassen der Zellularautomaten und der iterativen Arrays zusammen. Durch Präzisierung der definierenden Parameter lassen sich daraus weitere Unterklassen ableiten. So kann etwa zwischen Systemen mit uni- bzw. bidirektionalem Informationsfluß, mit paralleler bzw. sequentieller Eingabe oder mit unbeschränktem bzw. beschränktem Raum unterschieden werden.

Ausgehend von den bidirektionalen, real-raumbeschränkten Zellularautomaten mit paralleler Eingabe (CA) werden durch Einschränkung auf unidirektionalen Informationsfluß bzw. auf sequentielle Eingabe die sogenannten OCAs bzw. iterativen Arrays (IA) gewonnen. Beide Modelle sind im allgemeinen schwächer als CAs und untereinander nicht vergleichbar. Dieser (grobe) Zusammenhang spiegelt sich auch in der Lage der von ihnen erzeugten Sprachfamilien (L_t(CA), L_t(OCA), L_t(IA)) in der Chomsky-Hierarchie wider. Während alle Familien die regulären Sprachen enthalten und andererseits in den kontextsensitiven Sprachen enthalten sind, gibt es Unterschiede und offene Fragen bzgl. der kontextfreien Sprachen (L_2) bzw. ihrer diversen Teilfamilien.

Alle parallelen Familien enthalten auch kontextsensitive Sprachen. Es läßt sich aber zeigen, daß die Realzeit-IA-Sprachen nicht die deterministischen linearen kontextfreien Sprachen enthalten. Mittels einer algebraischen Methode kann nachgewiesen werden, daß die Realzeit-OCA-Sprachen ebenfalls nicht mit L_2 vergleichbar sind. Als Zeuge dient hier z.B. die Konkatenation einer linearen Sprache mit sich selbst.

Die Frage nach der Vergleichbarkeit von Linearzeit-CA-Sprachen und L_2 ist offen. Bekannte Ergebnisse hinsichtlich der Familien, die $\mathrm{L}_{rt}(\mathrm{CA}) \cup \mathrm{L}_2$ enthalten, können aber verbessert werden. Darüber hinaus werden die Beziehungen zwischen verschiedenen Teilfamilien von L_2 und den parallelen Sprachfamilien nachgewiesen.

Literatur

[1] Buchholz, Th., Klein, A., and Kutrib, M. *On interacting automata with limited nondeterminism.* IFIG Report 9806, Institut für Informatik, Universität Gießen, Gießen, 1998.

[2] Chang, J. H., Ibarra, O. H., and Vergis, A. *On the power of one-way communication.* Journal of the ACM 35 (1988), 697–726.

[3] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines.* IEEE Transactions on Computers C-18 (1969), 349–365.

[4] Kasami, T. and Fuji, M. *Some results on capabilities of one-dimensional iterative logical networks.* Electronics and Communications in Japan 51-C (1968), 167–176.

[5] Smith III, A. R. *Introduction to and survey of polyautomata theory.* In Lindenmayer, A. and Rozenberg, G. (eds.), *Automata, Languages, Development*. North-Holland, Amsterdam, 1976, pp. 405–422.

[6] Terrier, V. *On real time one-way cellular array.* Theoretical Computer Science 141 (1995), 331–335.

Erfüllbarkeit globaler Spurlogiken mittels alternierender Automaten

Jesper Gulmann Henriksen[1] und Martin Leucker[2]

[1] **BRICS**, Dept. of Computer Science
University of Aarhus, Denmark
[2] Lehrstuhl für Informatik II
RWTH Aachen

Ein üblicher Ansatz zur automatischen Verifikation von Programmen ist *Model Checking*, insbesondere zur Verifikation von Spezifikationen in linearer temporaler Logik. Automatenbasierte Ansätze haben sich hierfür als besonders hilfreich herausgestellt. Zu einer als Formel ϕ gegebenen Spezifikation wird ein Automat $\mathcal{A}_\phi$ konstruiert, der alle Modelle der Formel akzeptiert. Leerheit der Sprache dieses Automaten ist auf diese Weise äquivalent zur Unerfüllbarkeit der Spezifikation. Hat man ferner eine Implementierung der Spezifikation als Automat $\mathcal{B}$ gegeben, so liefert der Schnittautomat $\mathcal{B} \cap \mathcal{A}_{\neg\phi}$ die Menge aller Läufe, die die Spezifikation verletzen. Leerheit der Sprache des Schnittautomaten ist auf diese Weise äquivalent zum Model Checking Problem.

Üblicherweise sind Implementierungen verteilter Systeme in Form einzelner Komponenten gegeben, die teilweise Abhängigkeiten aufweisen, teilweise unabhängig voneinander arbeiten können. Daher wurden in den letzten Jahren temporale Logiken definiert und untersucht, die direkt über Bereichen interpretiert werden, die eine feste Unabhängigkeitsrelation implizieren. Insbesondere haben sich hier Logiken über *Mazurkiewicz Spuren* als hilfreich erwiesen. Derartige Logiken können unterschieden werden in *lokale* und *globale* Logiken. Während die ersten über den einzelnen Ereignissen innerhalb einer Spur interpretiert werden, ist die Semantik globaler Logiken definiert mit Hilfe von Mengen dieser Ereignisse, auch *Konfigurationen* genannt. Bekannte Vertreter lokaler Spurlogiken sind TrPTL und TLC, während LTrL und ISTL als globale Spurlogiken angesehen werden.

Während Logiken mit Unabhängigkeit eine klare Spezifikation eines Systems zulassen, ist die Komplexität für das Erfüllbarkeitsproblem dieser Logiken im allgemeinen sehr hoch. Auch ist die Konstruktion eines korrespondierenden Automaten kombinatorisch aufwendig. Insbesondere wurde von Walukiewicz gezeigt, daß LTrL nicht–elementar ist und somit kein effizienter automatentheoretischer Ansatz existieren kann. In [1] wurde ferner ein Fragment von LTrL betrachtet, bezeichnet als LTrL^-. Für diese Logik wurde ein doppelt–exponentielles Entscheidungsverfahren vorgestellt, welches nicht automatenbasiert und sehr technisch ist. Schließlich wurde gezeigt, daß dieses Verfahren optimal ist, da eine untere EXPSPACE Schranke angegeben wurde.

Wir zeigen, daß auch für globale Spurlogiken ein automatenbasierter Ansatz gewählt werden kann. Wir geben ein (optimales) Entscheidungsverfahren für LTrL^- an, welches auf alternierenden Automaten basiert. Die Konstruktion verwendet neben der Formel selbst auch bestimmte „unabhängige Ersetzungen“ von Teilen der Formel. Wir erhalten damit eine Form

des Fischer–Ladner–Abschlusses der Formel, deren Größe exponetiell in der Länge der Formel ist. Dieser wird für die Konstruktion eines alternierenden Automaten verwendet. Das Leerheitsproblem dieses Automaten kann in exponentieller Zeit entschieden werden, so daß sich die Gesamtkomplexität als doppelt exponentiell ergibt. Auf diese Weise erhalten wir die unserer Kenntnis nach erste automatenbasierte Konstruktion eines Erfüllbarkeitstests für eine globale Spurlogik. Neben dem Resultat selbst halten wir aber insbesondere die Methodik unseres Ansatzes für interessant. Die Komplexität, die durch die Unabhängigkeit einzelner Ereignisse hervorgerufen wird, kann durch die Modifikation der zugrundeliegenden Formel aufgefangen werden. Auf diese Weise kann die grundlegende Idee für die Konstruktion eines Automaten für eine Formel (mit sehr leichten Abänderungen) übernommen werden.

Literatur

[1] Walukiewicz, I.: Difficult configurations — on the complexity of LTrL (extended abstract). Proceedings of the 25th ICALP. Lecture Notes in Computer Science **1443**, Springer-Verlag (1998) 140–151

Die Theorie der ω-regulären Sprachen aus der Sicht der Alternierenden Schwachen Automaten

Christof Löding

Lehrstuhl für Informatik VII
RWTH Aachen

Alternierende schwache Automaten auf Bäumen wurden in [2] eingeführt, um eine automatentheoretische Charakterisierung der schwachen monadischen Logik zweiter Stufe (WMSO) über Bäumen zu geben. In alternierenden Automaten gibt es nicht nur die Möglichkeit der existentiellen Verzweigung, wie bei nichtdeterministischen Automaten, sondern zusätzlich sind auch universelle Verzweigungen zugelassen. Dies führt dazu, daß ein Lauf eines alternierenden Automaten statt eines Wortes ein azyklischer gerichteter Graph ist, dessen Knoten mit Zuständen beschriftet sind. In dem Spezialfall der nichtdeterministischen Automaten hat in diesem Graph jeder Knoten genau einen Nachfolger und jeder Knoten bis auf den Anfangsknoten hat genau einen Vorgänger. Also ist für nichtdeterministische Automaten ein Lauf wie üblich ein ω-Wort über der Zustandsmenge. Ein Lauf eines alternierenden Automaten ist akzeptierend, wenn jeder Pfad durch den Lauf die Akzeptierbedingung erfüllt. Ein Wort wird akzeptiert, wenn es einen akzeptierenden Lauf auf diesem Wort gibt.

In schwachen Automaten gibt es eine Abbildung, die jedem Zustand eine natürliche Zahl (seine sogenannte Farbe) zuordnet. Von einem Zustand aus dürfen Transitionen nur zu Zuständen gehen, deren Farbe kleiner oder gleich der Farbe des aktuellen Zustands ist. Ein Pfad durch den Lauf erfüllt die Akzeptierbedingung, wenn die kleinste Farbe der unendlich oft besuchten Zustände auf diesem Pfad gerade ist. Wegen der Einschränkung der erlaubten Transitionen in schwachen Automaten besucht jeder Pfad irgendwann nur noch Zustände mit der gleichen Farbe. Dies ist gleichzeitig auch die kleinste überhaupt besuchte Farbe. Also kann man die Akzeptierbedingung statt in der Menge der unendlich oft besuchten Zustände auch in der Menge der überhaupt besuchten Zustände interpretieren.

Über Wörtern hat WMSO die gleiche Ausdrucksstärke wie MSO, also haben die alternierenden schwachen Automaten über Wörtern die gleiche Ausdruckstärke wie nichtdeterministische Büchi-Automaten oder deterministische Paritätsautomaten. Allein deshalb erscheint es sinnvoll, diese Automaten eingehender zu studieren. Es gibt allerdings noch weitere Eigenschaften der alternierenden schwachen Automaten, die zeigen, daß diese Automaten interessant sind. Alternierende schwache Automaten lassen sich leicht durch das Dualisieren der Transitionsfunktion und der Akzeptierbedingung komplementieren [3]. Formeln der linearen temporalen Logik lassen sich in alternierende schwache Automaten übersetzen, deren Zustandszahl linear in der Länge der Formel ist (siehe z.B. [4]). Nichtdeterministische Büchi-Automaten lassen sich in alternierende schwache Automaten übersetzen, deren Zustandszahl quadratisch in der Zustandszahl der Büchi-Automaten ist [1].

In diesem Vortrag werden weitere Aspekte der Theorie der ω-regulären Sprachen aus der Sicht der alternierenden schwachen Automaten untersucht. Landweber hat drei echte

Unterklassen der ω-regulären Sprachen mit Hilfe von deterministischen Büchi- und co-Büchi-Automaten charakterisiert. Wir geben zwei unterschiedliche Charakterisierungen dieser Klassen durch alternierende schwache Automaten an.

Ein weiteres Fragment der ω-regulären Sprachen besteht aus den Sprachen, die sich in linearer temporaler Logik (oder auch in der Logik erster Stufe) definieren lassen. Auch für dieses Fragment gibt es eine Charakterisierung durch eine Teilklasse der alternierenden schwachen Automaten. Eine ω-regulären Sprache ist genau dann in linearer temporaler Logik definierbar, wenn es einen alternierenden schwachen Automaten gibt, der diese Sprache erkennt und bei dem alle Zustände unterschiedliche Farben haben. Das bedeutet, daß in dem Transitionsgraphen alle starken Zusammenhangskomponenten maximal einen Zustand enthalten.

Ein letzter Aspekt ist die Beschreibung von ω-Automaten in MSO. Büchi hat bewiesen, daß man jede durch einen nichtdeterministischen Büchi Automaten erkennbare Sprache auch durch eine existentielle MSO-Formel definieren kann, indem man die akzeptierenden Läufe des Automaten beschreibt. Für alternierende schwache Automaten geben wir eine Δ_2-Charakterisierung in MSO an. Dies spiegelt auch die Selbstdualität dieser Automaten auf der logischen Seite wider.

Die erzielten Ergebnisse sind in Zusammenarbeit mit Wolfgang Thomas entstanden.

Literatur

[1] O. Kupferman and M.Y. Vardi, *Weak Alternating Automata are not that Weak*, in Proc. 5th Israeli Symposium on Theory of Computing and Systems, IEEE Computer Society Press, 1997, pp. 147-158.

[2] D.E. Muller, A. Saoudi and P.E. Schupp, *Alternating Automata, the Weak Monadic Theory of the Tree and its Complexity*, in Proc. 13th ICALP, LNCS 226, 1986, pp. 275-283.

[3] D.E. Muller and P.E. Schupp, *Alternating Automata on Infinite Trees*, TCS 54, 1987, pp. 267-276.

[4] M.Y. Vardi, *An Automata-Theoretic Approach to Linear Temporal Logic*, in Logics for Concurrency: Structure versus Automata, LNCS 1043, Springer, 1996, pp. 238-266.

Zur Präperfektheit von Semi-Thue Systemen

Markus Lohrey

Institut für Informatik
Universität Stuttgart

In dem Vortrag werden einige neue Resultate zur Präperfektheit von Semi-Thue Systemen vorgestellt. Im folgenden werden kurz die zum Verständnis des Vortrages notwendigen Definitionen eingeführt sowie die oben erwähnten Resultate präsentiert.

Im folgenden sei Σ ein endliches Alphabet. Die Länge eines Wortes $x \in \Sigma^*$ wird mit $|x|$ bezeichnet. Sei $\mathcal{R} \subseteq \Sigma^* \times \Sigma^*$ ein endliches Semi-Thue System über Σ, wobei alle Regeln aus $\mathcal{R}$ nicht verlängernd sein sollen, d.h. $\forall(\ell, r) \in \mathcal{R} : |\ell| \geq |r|$. Die Regeln in $\mathcal{R}$ können dann in einen längenreduzierenden Teil $\mathcal{R}_> = \{(\ell, r) \in \mathcal{R} \mid |\ell| > |r|\}$ und einen längenerhaltenden Teil $\mathcal{R}_= = \mathcal{R} \setminus \mathcal{R}_>$ aufgeteilt werden. Mit dem Semi-Thue System $\mathcal{R}$ werden die folgenden binären Relationen auf Σ^* assoziiert [1]:

- $\leftrightarrow = \{(xuy, xvy) \mid x, y \in \Sigma^*, (u, v) \in \mathcal{R} \text{ oder } (v, u) \in \mathcal{R}\}$
- $\vdash = \{(xuy, xvy) \mid x, y \in \Sigma^*, (u, v) \in \mathcal{R}_= \text{ oder } (v, u) \in \mathcal{R}_=\}$
- $\mapsto = \vdash \cup \{(xuy, xvy) \mid x, y \in \Sigma^*, (u, v) \in \mathcal{R}_>\}$

Das Semi-Thue System $\mathcal{R}$ wird schließlich als *präperfekt* bezeichnet, falls für alle $x, y \in \Sigma^*$ gilt:

$$x \overset{*}{\leftrightarrow} y \quad \text{genau dann, wenn} \quad \exists z \in \Sigma^* : x \overset{*}{\mapsto} z \text{ und } y \overset{*}{\mapsto} z$$

Es ist einfach zu sehen, daß sich für ein präperfektes Semi-Thue System das Wortproblem in PSPACE befindet. Dies motiviert die Frage nach der Entscheidbarkeit der Präperfektheit von Semi-Thue Systemen. In [NM84] wurde gezeigt, daß Präperfektheit unentscheidbar für Semi-Thue Systeme ist. Dieses Resultat wurde in [NO88] auf die folgende Art verschärft:

Eine Unabhängigkeitsrelation $I \subseteq \Sigma \times \Sigma$ ist eine irreflexive und symmetrische Relation. Mit I wird die Relation $\vdash_I = \{(ab, ba) \mid (a, b) \in I\}$ assoziiert. Der reflexive und transitive Abschluß von $\vdash_I$ wird mit $\equiv_I$ bezeichnet. Da dieser eine Kongruenzrelation auf Σ^* ist, kann das Quotientenmonoid $\mathbb{M}(\Sigma, I) = \Sigma^*/_{\equiv_I}$ gebildet werden, welches als das von I erzeugte *Spurmonoid* bezeichnet wird. In [NO88] wurde nun gezeigt, daß Präperfektheit auch noch unentscheidbar für Semi-Thue Systeme von der Form $\mathcal{R} \cup \vdash_I$ ist, wobei $\mathcal{R}$ nur längenreduzierende Regeln enthält, und I eine Unabhängigkeitsrelation ist. Dieses Resultat ist insbesondere interessant, da leicht zu sehen ist, daß $\mathcal{R} \cup \vdash_I$ genau dann präperfekt ist, wenn $\mathcal{R}$ in dem Spurmonoid $\mathbb{M}(\Sigma, I)$ ein konfluentes Ersetzungssystem darstellt.

Dieses Unentscheidbarkeitsresultat motiviert nun die Suche nach eingeschränkteren Teilklassen von Semi-Thue Systemen, für die Präperfektheit entscheidbar ist. In dem Vortrag wird gezeigt, daß Präperfektheit in Polynomialzeit entscheidbar ist für Semi-Thue Systeme

[1] Wir verzichten hier auf den Index $\mathcal{R}$, da dies zu keinen Mehrdeutigkeiten führen wird.

der Form $\mathcal{R} \cup \vdash_I$, wobei $\mathcal{R}$ nur löschende Regeln enthält, d.h. $\forall(\ell, r) \in \mathcal{R} : |r| = 0$, und I eine Unabhängigkeitsrelation ist [Loh99]. Dieses Resultat beantwortet insbesondere die Frage nach der Entscheidbarkeit der Konfluenz von löschenden Spurersetzungssystemen positiv [Die90]. Andererseits wird gezeigt, das Präperfektheit für Semi-Thue Systeme auch dann noch unentscheidbar ist, falls der längenreduzierende Teil nur löschende Regeln enthält. Dabei kann entweder der längenreduzierende Teil oder der längenerhaltende Teil fest gewählt werden.

Literatur

[Die90] V. Diekert. *Combinatorics on Traces.* Number 454 in Lecture Notes in Computer Science. Springer, Berlin-Heidelberg-New York, 1990.

[Loh99] M. Lohrey. Complexity results for confluence problems. Technical Report 1999/05, Universität Stuttgart, 1999.
ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/TR-1999-05/.

[NM84] P. Narendran and R. McNaughton. The undecidability of the preperfectness of Thue systems. *Theoretical Computer Science*, 31:165–174, 1984.

[NO88] P. Narendran and F. Otto. Preperfectness is undecidable for Thue systems containing only length-reducing rules and a single commutation rule. *Information Processing Letters*, 29:125–130, 1988.

Kohärenzeigenschaften in termersetzungsbasierten Modellen für verteilte Systeme

Thomas Noll

Lehrstuhl für Informatik II
RWTH Aachen

Im Rahmen des Forschungsprojekts „Modellierung verteilter Systeme“ wird am Lehrstuhl für Informatik II der RWTH Aachen das Verifikationstool TRUTH entwickelt. Dieses unterstützt gegenwärtig tableaubasierte Modelchecking–Verfahren für den vollen μ–Kalkül sowie spielbasiertes Modelchecking für die alternierungsfreie Teillogik. Beide operieren auf dem semantischen Bereich der endlichen Transitionssysteme, welche durch CCS–Prozeßsysteme spezifiziert sind. Darüber hinaus bietet TRUTH interaktive Visualisierungs- und Simulationsmöglichkeiten.

Gegenwärtig wird an einer Erweiterung dieses Tools gearbeitet, welche die Verwendung anderer Spezifikationssprachen und semantischer Bereiche unterstützt. Dazu wird ein Compilergenerator entwickelt, welcher anhand der Beschreibung einer Spezifikationssprache wie CCS ein entsprechendes TRUTH–Frontend generiert. Die Syntax und Semantik einer solchen Sprache wird in einer Variante von Meseguer's *Rewriting Logic* notiert. Bei diesem Ansatz werden die Zustände eines verteilten Systems durch Terme (genauer: durch eine Gleichungsmenge E charakterisierte Äquivalenzklassen von Termen) und sein operationelles Verhalten durch (bedingte) Transitionsregeln beschrieben.

Als operationelles Modell ergibt sich Termersetzung modulo beliebiger Gleichungstheorien, welches im allgemeinen Fall als sehr aufwendig oder sogar unentscheidbar bekannt ist. Für die hier verfolgten Anwendungen im Bereich der Modellierung verteilter Systeme erweist sich jedoch die Beschränkung auf Termersetzung modulo Assoziativität und Kommutativität als ausreichend. Dazu wird die Gleichungsmenge E in zwei Teilmengen AC und ER zerlegt. Hierbei enthält AC alle Assoziativitäts- und Kommutativitätsgesetze, und ER besteht nur noch aus gerichteten Gleichungen, also Termersetzungsregeln, so daß sich ein terminierendes System (modulo AC) ergibt. In der Implementierung werden nun keine Termersetzungsschritte modulo E berechnet, sondern es wird zunächst die ER–Normalform des aktuellen Terms (wiederum modulo AC) gebildet, welche anschließend dem eigentlichen Transitionsschritt unterworfen wird.

In diesem Vortrag wird untersucht, unter welchen Bedingungen diese Implementierung vollständig ist, d.h. alle Berechnungsergebnisse liefert, die sich mit der vollen Gleichungsmenge bestimmen lassen. Dazu wird eine *Kohärenzeigenschaft* definiert, welche die Vollständigkeit gewährleistet. Weiter zeigen wir, daß sich diese Eigenschaft durch die Überprüfung endlich vieler *kritischer Paare* zwischen den Transitionsregeln und den orientierten Gleichungen aus ER überprüft werden kann.

Beschreibungsgröße zyklischer Sprachen

Holger Petersen

Institut für Informatik
Universität Stuttgart

Wir untersuchen die Zustandskomplexität regulärer unärer Sprachen über dem Alphabet $\{0\}$. Eine unäre Sprache L wird λ-zyklisch genannt, wenn es eine positive ganze Zahl λ gibt, für die

$$0^n \in L \Longleftrightarrow 0^{n+\lambda} \in L$$

für alle $n \geq 0$ gilt. Eine Sprache ist zyklisch, wenn ein λ mit dieser Eigenschaft existiert. Echt λ-zyklisch ist eine Sprache, wenn sie λ-zyklisch ist und λ minimal mit dieser Eigenschaft ist.

In [1] zeigen Mereghetti und Pighizzini als Theorem 3.3 das folgende Resultat.

Satz 4 ([1]). *Sei L eine λ-zyklische Sprache mit $\lambda = p_1^{k_1} \cdot p_2^{k_2}$, wobei $p_1 \neq p_2$ Primzahlen sind und $k_1, k_2 \geq 0$ ganze Zahlen. Dann existiert ein nichtdeterministischer endlicher Zweiweg-Automat mit $p_1^{k_1} + p_2^{k_2}$ Zuständen, der L akzeptiert.*

Bei Mereghetti und Pighizzini bleibt offen, ob eine entsprechende Aussage für deterministische Automaten gilt. Wir zeigen, dass dies nicht der Fall ist. Dieser Unterschied zwischen der Beschreibungskomplexität durch deterministische und nichtdeterministische Automaten steht im Zusammenhang mit der Bemerkung in [2], dass die Anzahl der Zustände eines Automaten nicht in jedem Fall ein adäquates Maß seiner Komplexität ist. Ein nichtdeterministischer Automat kann eine in Bezug auf die Zustandsmenge quadratische Anzahl von Transitionen besitzen und auf diese Weise mehr Information als ein deterministischer Automat kodieren. Tatsächlich macht die Konstruktion aus [1] Gebrauch von einer quadratischen Anzahl an Transitionen.

Satz 5. *Für alle bis auf endlich viele $k \geq 0$ existiert eine λ-zyklische Sprache L mit $\lambda = 6^k = 2^k \cdot 3^k$, so dass kein deterministischer endlicher Zweiweg-Automat mit $2^k + 3^k$ Zuständen L akzeptiert.*

Ähnliche Überlegungen wie im Beweis von Satz 5 zeigen, dass die Simulation aus Theorem 3.3 in [1] sich nicht auf λ mit mehr als zwei verschiedenen Primfaktoren verallgemeinern lässt.

Die Folge a_n gibt die Anzahl verschiedener Halsketten an, die aus n Perlen gebildet werden können, wobei zwei Farben für die Perlen zur Verfügung stehen. Halsketten können gedreht, aber nicht umgeklappt werden.

Die Folge a_n erscheint in [3] als M0564 und wird in Figur M3860 besprochen. Eine geschlossene Formel ist

$$a_n = \frac{1}{n} \sum_{d|n} \phi(d) 2^{n/d},$$

wobei ϕ die Eulersche Totient-Funktion ist. Unter Verwendung des offensichtlichen Zusammenhangs zwischen Halsketten und Zyklen in endlichen Automaten, wobei die Färbung der Aufteilung in Endzustände und Nicht-Endzustände entspricht, können $n \cdot m$-zyklische Sprachen für teilerfremde n, m durch Automaten akzeptiert werden, die aus einem Zyklus von m Zuständen und a_n Zyklen von jeweils höchstens n Zuständen aufgebaut sind.

Satz 6. *Seien n und m teilerfremde positive Zahlen und $\lambda = n \cdot m$. Jede λ-zyklische Sprache kann von einem deterministischen endlichen Zweiweg-Automaten mit $2^n - 2 + m$ Zuständen akzeptiert werden.*

Mereghetti und Pighizzini formulieren als offenes Problem, ob die trivialen Simulationen von Einweg-Automaten durch Zweiweg-Automaten für zyklische Sprachen optimal sind. Ihr Theorem 3.1 zeigt die Optimalität für λ-zyklische Sprachen mit einer Primzahlpotenz λ. Die obige Simulation zeigt, dass eine Verbesserung für λ's erreicht werden kann, die sich in teilerfremde Faktoren genügend unterschiedlicher Größe (z.B. $n = 2, m = 3$) zerlegen lassen, womit die offene Frage negativ beantwortet ist.

Die verbesserte Simulation zeigt, dass für den Spezialfall einer Zerlegung in zwei Faktoren mit $n = 2$ eine Schranke wie in Satz 4 gilt.

Korollar 1. *Jede λ-zyklische Sprache mit $\lambda = 2 \cdot m$ für ungerades m kann von einem endlichen deterministischen Zweiweg-Automaten mit $2 + m$ Zuständen akzeptiert werden.*

Offen bleibt, ob sich für Faktorisierungen von λ in teilerfremde Faktoren ähnlicher Größe Verbesserungen bei der Simulation von Einweg-Automaten, die λ-zyklische Sprachen akzeptieren, durch deterministische Zweiweg-Automaten erzielen lassen. Kleinstes Beispiel, für das Satz 6 keine Verbesserung bringt, ist $\lambda = 35$.

Literatur

[1] C. Mereghetti and G. Pighizzini. Unary automata simulations and cyclic languages. In J. Dassow and D. Wotschke, editors, *Preproceedings of the International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, Preprint Nr. 17, Department of Computer Science, Otto-von-Guericke University, Magdeburg*, pages 145–153, 1999.

[2] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proceedings of the 12th Annual IEEE Symposium on Switching and Automata Theory, East Lansing (Michigan)*, pages 188–191, 1971.

[3] N. J. A. Sloane and S. Plouffe. *The Encyclopedia of Integer Sequences.* Academic Press, San Diego, 1995.

Church-Rosser Hypergraph Languages

Detlef Plump

Fachbereich Mathematik/Informatik
Universität Bremen

Church-Rosser hypergraph languages are defined similar to Church-Rosser string languages as introduced by McNaughton, Narendran and Otto. A hypergraph language L of (pointed) hypergraphs over some label alphabet Σ is a *Church-Rosser language* if there is a finite and confluent hypergraph rewriting system $\mathcal{R}$ over an extended label alphabet Γ, an irreducible context hypergraph C over Γ, and an irreducible hypergraph I over Γ such that a hypergraph G belongs to L if and only if $\mathcal{R}$ reduces $G^{\oplus}$ to I, where $G^{\oplus}$ denotes the gluing of G and C along their points. Moreover, $\mathcal{R}$ must only admit derivations of polynomial length (which includes the special case that all rules in $\mathcal{R}$ are size-reducing).

A key feature of Church-Rosser hypergraph languages is that their membership problem is solvable in polynomial time. Moreover, for terminating hypergraph rewriting systems an effective sufficient condition for confluence is given by the (strong) joinability of all *critical pairs*. This provides a criterion for testing whether a polynomially terminating hypergraph rewriting system defines a Church-Rosser language.

Several interesting (hyper-)graph languages can be shown to be Church-Rosser languages. For example, the "semi-structured flow graphs" of Farrow, Kennedy and Zucconi form a Church-Rosser language. Moreover, there are Church-Rosser languages that are not context-free in the sense that they cannot be generated by hyperedge replacement grammars. The class of all connected graphs and the class of all jungles (generalized term graphs) are two examples of this kind. On the other hand, it is known that there exist context-free hypergraph languages with an NP-complete membership problem and hence—provided P $\neq$ NP—not all context-free hypergraph languages are Church-Rosser languages.

Finite Automata Encoding Geometric Figures

Lugwig Staiger[1] and Helmut Jürgensen[2]

[1] Institut für Informatik
Universität Halle
[2] Institut für Informatik
Universität Potsdam

Finite automata are used for the encoding and compression of images. For black-and-white images, for instance, using the quad-tree representation, the black points correspond to ω-words defining the corresponding paths in the tree that lead to them. If the ω-language consisting of the set of all these words is accepted by a deterministic finite automaton then the image is said to be encodable as a finite automaton. For grey-level images and colour images similar representations by automata are in use.

In this paper we address the question of which images can be encoded as finite automata with full infinite precision. In applications, of course, the image would be given and rendered at some finite resolution – this amounts to considering a set of finite prefixes of the ω-language – and the features in the image would be approximations of the features in the infinite precision rendering.

We focus on the case of black-and-white images – geometrical figures, to be precise – but treat this case in a d-dimensional setting, where d is any positive integer. We show that among all polygons in d-dimensional space exactly those with rational corner points are encodable as finite automata.

In the course of proving this we show that the set of images encodable as finite automata is closed under rational affine transformations.

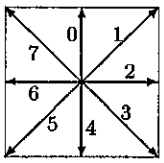
Several simple properties of images encodable as finite automata are consequences of this result. Finally we show that many simple geometric figures such as circles and parabolas are not encodable as finite automata.

Zur Endlichkeit von Bildsprachen synchroner deterministischer Ketten-Code-Bild-Systeme

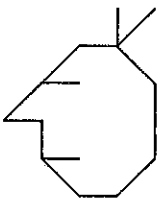
Bianca Truthe

Fakultät für Informatik
Universitat Magdeburg

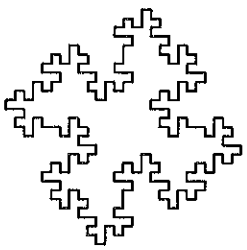
Zentrale Aufgaben der Bildverarbeitung liegen beim Beschreiben, Erzeugen, Speichern und Erkennen von Bildern. Mit Ketten-Codes lieferte FREEMAN in den 60er Jahren eine Beschreibungsmöglichkeit für Strichgraphiken. Dabei entsteht ein Bild durch eine Folge von Zeichenbewegungen, die durch Symbole, z. B. Buchstaben repräsentiert sind. Ein Wort beschreibt ein Bild, das durch Nacheinanderausführung der Zeichenschritte seiner Buchstaben entsteht. FREEMAN benutzt ein achtelementiges Alphabet $\{0,\ldots,7\}$, dessen Elemente entsprechend folgender Skizze interpretiert werden:

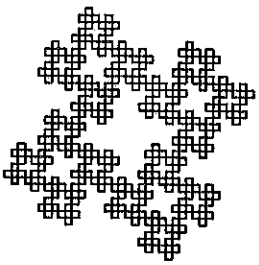


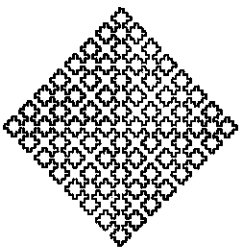
Zum Beispiel entsteht das nebenstehende Bild aus dem Wort 1261204153445672606:
(Beim Nachvollziehen beginne man an der Nasenspitze.)



Mittels Ketten-Codes lassen sich Muster – wie Fraktale, Kurven, Folkloremuster – beschreiben:







Der Interpretationszusammenhang von Wörtern und Bildern legt es nahe, Beziehungen zwischen formalen Sprachen und Bildmengen zu suchen. Für sprachentheoretische Betrachtungen genügen die vier Richtungen $\{0,2,4,6\}$. In Anlehnung an Plotter-Befehle schreibt man u,r,d,l für die Richtungen *up*, *right*, *down*, *left*.
Ketten-Code-Bild-Systeme sind LINDENMAYER-Systeme über einem speziellen Alphabet. Die erzeugten Wörter werden als Bilder interpretiert. Dies führt zu Ketten-Code-Bild-Sprachen. Die vorliegende Arbeit untersucht synchrone, deterministische Ketten-Code-Bild-Systeme über dem Alphabet $\{r,u,l,d\}$ hinsichtlich der Endlichkeit der von ihnen erzeugten Bildsprachen.
Zunächst wird eine Abstrahierungshierarchie entwickelt, in der die Interpretation eines Wortes als Bild einen mehrstufigen Prozeß durchläuft. Die unterste Schicht bilden Wörter einer

Sprache. Den Wörtern sind gerichtete Graphen (mit Mehrfachkanten) auf der nächsthöheren Ebene zugeordnet. Durch Abstrahieren von den Kantenwiederholungen (dritte Stufe) und Kantenrichtungen gelangt man schließlich auf die Ebene der Bilder. Anhand dieses Modells wird nachgewiesen, daß es entscheidbar ist, ob ein *sD0L*-System eine endliche oder unendliche Bildmenge erzeugt.
Die Menge aller *sD0L*-Systeme läßt sich nach dem Synchronisationsparameter k in folgende Teilmengen zerlegen:

- $\underline{\mathcal{S}} = \{ (\mathcal{A}, h, \omega, k, m) \mid k < 1 \}$,
- $\mathcal{S} = \{ (\mathcal{A}, h, \omega, k, m) \mid k = 1 \}$,
- $\overline{\mathcal{S}} = \{ (\mathcal{A}, h, \omega, k, m) \mid k > 1 \}$.

Im Falle $k < 1$ generiert jedes *sD0L*-System $G \in \underline{\mathcal{S}}$ eine endliche Bildmenge mit höchstens vier Elementen; im Falle $k > 1$ ist die erzeugte Bildsprache jedes Systems $G \in \overline{\mathcal{S}}$ unendlich. Für $k = 1$ gibt es sowohl *sD0L*-Systeme, die eine endliche Menge von Bildern liefern, als auch solche, die eine unendliche Bildmenge erzeugen. Mittels einer notwendingen und hinreichenden Bedingung kann jedoch nach dreimaligem Ableiten des Startworts die Endlichkeit bzw. Unendlichkeit entschieden werden. Falls die Bildmenge endlich ist, so enthält sie höchstens drei Elemente. Gilt für die Synchronisationsparameter $k = 1$ und $m = 0$, so ist die Bildsprache stets einelementig; ein dreimaliges Ableiten des Axioms ist daher in diesem Spezialfall nicht nötig. Die folgende Tabelle faßt die Ergebnisse zusammen:

$$
\begin{array}{lll}
G = (\mathcal{A}, h, \omega, k, m) \text{ mit } k < 1: & B(G) = \{ p(\omega), p(\omega'), p(\omega''), p(\omega''') \} \\
G = (\mathcal{A}, h, \omega, k, m) \text{ mit } k > 1: & |B(G)| = \infty \\
G = (\mathcal{A}, h, \omega, k, m) \text{ mit } k = 1: & \\
\qquad \text{allgemein:} & sg(\omega'') = sg(\omega''') \Longrightarrow B(G) = \{p(\omega), p(\omega'), p(\omega'')\} \\
& sg(\omega'') \neq sg(\omega''') \Longrightarrow |B(G)| = \infty \\
\qquad \text{mit } m = 0: & B(G) = \{ p(\omega) \}
\end{array}
$$

Aus dieser Aufspaltung folgt: Wenn die von einem *sD0L*-System erzeugte Bildsprache endlich ist, so enthält sie höchstens vier Elemente. Außerdem ist damit ein Algorithmus gegeben, der zu einem beliebigen *sD0L*-System G entscheidet, ob die erzeugte Bildsprache $B(G)$ endlich ist oder nicht.
Gilt bei einem gegebenen *sD0L*-System $G = (\mathcal{A}, h, \omega, k, m)$ für die Parameter $k < 1$, $k > 1$ oder $k = 1, m = 0$, so kann die Entscheidung, ob die erzeugte Bildmenge endlich ist oder nicht, sofort (ohne eine weitere Untersuchung des Systems), d. h. mit konstantem Zeitaufwand getroffen werden. Sonst ist das Startwort dreimal abzuleiten, und es sind die Kantenmengen der zweiten und dritten Ableitung auf Übereinstimmung zu prüfen. Damit ergibt sich ein Zeitaufwand, der kubisch in der Länge der ersetzenden Wörter ist.
Das bedeutet insgesamt: Für synchrone, deterministische, kontextfreie Ketten-Code-Bild-Systeme $G = (\mathcal{A}, h, \omega, k, m)$ sind Endlich- und Unendlichkeit in der Zeit $\mathcal{O}(pn^3)$ entscheidbar, wobei p die Länge des Startwortes und n die maximale Länge der ersetzenden Wörter sind: $p = \#\omega$ und $n = \max \{ \#h(x) \mid x \in \mathcal{A} \}$.

Using Recursion in MSO Tree Transductions

Zoltán Fülöp[1] und Heiko Vogler[2]

[1] Department of Computer Science
József Attila University, Szeged, Hungary
[2] Department of Computer Science
Technical University of Dresden

MSO graph transducers were introduced in [EB98]; they are devices that transform graphs into graphs. An MSO graph transducer M contains a family of unary node formulas and a family of binary edge formulas; both, node formulas and edge formulas are taken from monadic second order (MSO) logic. Roughly speaking, given an input graph g_1, the transformation of g_1 by M, i.e., the output graph $M(g_1) = g_2$ is obtained by interpreting the node formulas and edge formulas on the graph g_1 and thereby constructing the output graph g_2.

It was shown in [EB98] that MSO tree transducers induce the same class of tree transformations as single use restricted attributed tree transducers with regular look-ahead. And in [EM98] the equality with the class of tree transformations induced by single use restricted macro tree transducers with regular look-ahead was shown.

In this talk we consider MSO graph transducers and equip them with the possibility to define the node formulas recursively by means of a finite system of equations, where the tree transformation induced by a recursive MSO graph transducer involves the standard construction of a fixpoint. We prove that this extension does not increase the tree transformation power of the transducer, i.e., recursive MSO tree transducers and MSO tree transducers induce the same as the class of tree transformations. The key argument in the proof is that the fixpoint can be specified by an MSO formula which can be "read" from the fixpoint theorem of Knaster-Tarski [FV99].

References

[EB98] R. Bloem and J. Engelfriet. A Comparison of Tree Transductions defined by Monadic Second Order Logic and by Attribute Grammars. Technical Report 98-02 Department of Computer Science, Leiden University. January 1998.

[EM98] J. Engelfriet and S. Maneth. Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations. Technical Report 98-09 Department of Computer Science, Leiden University. August 1998.

[FV99] Z. Fülöp and H. Vogler. Using Recursion in MSO Tree Transductions. work in progress, 1999.

Boolesche Operationen auf Wort- und Baum-Automaten mit kurzen Schleifen

Johannes Waldmann

Institut für Informatik
Universität Leipzig

1 Einleitung

Reguläre Sprachen von Wörtern oder Bäumen werden durch endliche Automaten beschrieben. Wir betrachten hier solche Automaten, in deren Graphen nur Schleifen der Länge eins vorkommen, und auch von diesen nicht beliebige.

Die einfachsten Automaten sind solche für *endliche* Sprachen. Sie enthalten überhaupt keine Schleifen: wir können die Zustandsmenge Q eines solchen Automaten A derart ordnen, daß bei jeder Transition $q \xrightarrow{a} q'$ gilt: $q > q'$.

Wenn wir in akzeptierenden Zuständen q Schleifen $q \xrightarrow{\Sigma} q$ zulassen, erhalten wir *definite* Sprachen. Beide Klassen (endliche, definite Sprachen) sind abgeschlossen unter Booleschen Operationen.

Es liegt nahe, Schleifen der Länge eins nicht nur in finalen, sondern in allen Zuständen zu erlauben. Die Klasse der durch solche Automaten erzeugten Sprachen ist aber nicht mehr Boolesch abgeschlossen.

Diesen Abschluß gewinnen wir zurück, wenn Schleifen in inneren Zuständen immer mit nur einem Buchstaben beschriftet sein müssen.

Wir übertragen schließlich dieses Resultat von Wörtern auf Bäume. Wir stellen die Sprachen nicht durch Automaten, sondern durch passend eingeschränkte reguläre Ausdrücke dar, und geben Algorithmen an, die für diese Ausdrücke Durchschnitt und Komplement ausrechnen.

Wir hoffen, daß derartige Baumsprachen gut zu gewissen Term-Ersetzungs-Systemen passen. Ein Resultat in dieser Richtung ist: Die Menge der kopfnormalisierenden $\mathbf{CL}(S)$-Terme hat eine Darstellung mit kleinen Schleifen.

2 Beispiele

Wörter. Wir fixieren das Alphabet $\Sigma = \{a, b\}$.

1.) Das Komplement von ab ist $\epsilon \cup b\Sigma^* \cup a(\epsilon \cup a\Sigma^* \cup b(a\Sigma^* \cup b\Sigma^*))$, eine definite Sprache.

2.) Die Sprache $X = \Sigma^*(aa \cup bb)\Sigma^* \cup b\Sigma^* \cup \Sigma^* a$ hat offensichtlich einen Automaten mit kurzen Schleifen (wir brauchen je eine für die Σ^*). Das Komplement von X ist $(ab)^*$, mit einer Schleife der Länge 2.

3.) Das Komplement von $Y = a^*b\Sigma^*$ ist a^*. Beachte, daß in Y das Σ^* ganz rechts steht, im Gegensatz zur Position in X aus dem vorigen Beispiel.

Bäume. Wir nehmen die Signatur mit Blattsymbol b und binärem Symbol $\cdot$, sowie die Sprache, die von Z in der folgenden Grammatik erzeugt wird:

$$M \to (b \cdot b) \cdot *,\ Z \to M,\ Z \to Z \cdot b,$$

wobei $*$ hier die Menge aller Bäume bezeichnet. Die zweite Regel ist offenbar eine kurze Schleife. Besitzt das Komplement von Z eine solche Darstellung?

Die in dieser Grammatik benutze Menge M ist eine definite Baumsprache, ihr Komplement ist

$$\overline{M} = b \cup b \cdot * \cup (* \cdot (* \cdot *)) \cdot * \cup ((* \cdot *) \cdot *) \cdot *,$$

und sowohl M als auch $\overline{M}$ sind mit kurzen Schleifen darstellbar. (Wir brauchen ja nur eine Schleife in dem Zustand, der $*$ erzeugt.)

Das Komplement von Z ist aber *nicht* die Sprache

$$Z' \to \overline{M},\ Z' \to Z' \cdot b,$$

denn $(b \cdot b) \cdot b \in (b \cdot b) \cdot * \subseteq Z$, aber auch $(b \cdot b) \cdot b \in (b \cdot *) \cdot b \subseteq \overline{M} \cdot b \subseteq Z'$.

Wir werden einen Algorithmus angeben, der eine Darstellung von $\overline{Z}$ mit kurzen Schleifen berechnen kann.

Literatur

[Heu88] U. Heuter. Generalized definite tree languages. Aachener Informatik-Berichte 88-24, RWTH Aachen, 1988.

[Pot94] Andreas Potthoff. *Logische Klassifizierung regulärer Baumsprachen.* PhD thesis, Christian-Albrechts-Universität Kiel, 1994. Bericht Nr. 9410.

[SY98] Kai Salomaa and Sheng Yu. Alternating finite automata and star-free languages. *Theoretical Computer Science*, page (submitted), 1998.

[Wal99] Johannes Waldmann. (Head normalization and) Top termination in CL(S). talk given at 4th International Workshop on Termination, Schloß Dagstuhl, May 1999.

Zerlegung polynomiell mehrdeutiger kontextfreier Grammatiken

Klaus Wich

Universität Kassel

Aufgrund der Existenz inhärent mehrdeutiger Sprachen sehen wir, dass mehrdeutige Grammatiken einen wichtigen Beitrag zur generativen Mächtigkeit der kontextfreien Sprachen leisten. Jedes von einer zykelfreien Grammatik G erzeugte Wort hat eine endliche Anzahl von Ableitungsbäumen. Falls es für diese Anzahl eine vom Wort unabhängige obere Schranke c gibt, spricht man von einer Grammatik mit Mehrdeutigkeitsgrad c. Doch in vielen Fällen wächst die Anzahl der Ableitungsbäume mit der Wortlänge. Der funktionale Zusammenhang zwischen der Wortlänge und der Anzahl der Ableitungsbäume wurde erstmals in [Wich 97] untersucht. Für semi-propere Grammatiken, das sind zykelfreie Grammatiken ohne nutzlose Symbole und ohne solche Nichtterminale, die ausschließlich das leere Wort erzeugen, konnte ein Kriterium gezeigt werden, dessen Erfüllung exponentielle Mehrdeutigkeit und dessen Verletzung polynomielle Mehrdeutigkeit impliziert.

Eine Grammatik heißt extrem (bezüglich ihrer Mehrdeutigkeit), wenn sie entweder eindeutig oder exponentiell mehrdeutig ist. Jede Instanz des Postschen Korrespondenz Problems I lässt sich effektiv in eine extreme Grammatik G codieren, so dass G exponentiell mehrdeutig ist, wenn I eine Lösung hat, und sonst eindeutig ist. Damit ist das obige Kriterium unentscheidbar.

Falls wir von einer Grammatik bereits wissen, dass sie höchstens polynomiell mehrdeutig ist, lassen sich durch Anwendung des obigen Kriteriums effektiv einige Beziehungen zu eindeutigen Sprachen herstellen. So kann man zu jeder semi-properen Grammatik G effektiv eine Konstante c und eine Teilmenge der Produktionsmenge bestimmen, so dass die Produktionen dieser Menge höchstens c mal in einem Ableitungsbaum angewendet werden können. Durch den Einbau geeigneter Markierungen in diese Produktionen erhält man eine extreme Grammatik G', die eindeutig ist, wenn G subexponentiell mehrdeutig ist. Polynomielle Mehrdeutigkeit kann also durch eine beschränkte Anzahl von Markierungen, die nur von der Grammatik nicht aber von der Wortlänge abhängt, eliminiert werden. Die maximale Anzahl der in einem Wort auftauchenden Markierungen bildet eine obere Schranke für den Polynomgrad der Mehrdeutigkeitsfunktion. Exponentiell mehrdeutige Sprachen können durch Einbau einer beschränkten Anzahl von Markierungen hingegen nicht eindeutig gemacht werden.

Man kann diese Überlegung benutzen um zu zeigen, dass der Abschluß eindeutiger Sprachen unter Vereinigung und Substitution von Symbolen mit endlichem Rang Teilmenge der polynomiell mehrdeutigen Sprachen ist. Der Rang eines Symbols ist dabei die maximale Anzahl von Vorkommen in einem Wort der erzeugten Sprache. Tatsächlich kann sogar jede semi-propere Grammatik effektiv in einen regulärartigen Ausdruck über extremen Teilgrammatiken zerlegt werden, der nur Vereinigungen sowie Substitutionen von Terminalen mit endlichem Rang enthält. Die Teilgrammatiken sind dabei alle eindeutig, wenn G polynomiell

mehrdeutig ist, sonst ist wenigstens eine Teilgrammatik exponentiell mehrdeutig. Somit sind die polynomiell mehrdeutigen Grammatiken charakterisiert durch den Abschluß eindeutiger Sprachen unter Vereinigung und Substitution von Symbolen mit endlichem Rang.

Literatur

[Wich 97] Klaus Wich:
Kriterien für die Mehrdeutigkeit kontextfreier Grammatiken
Diplomarbeit, FB Informatik, Johann–Wolfgang–Goethe–Universität
Frankfurt am Main, 1997

Überarbeitete Version von [Wich 97]
http://www.db.informatik.uni-kassel.de/~wich/publications.html

Eine unter Bildung der Präfixsprache abgeschlossene Teilmenge der Church-Rosser-Sprachen

Jens Woinowski

Fachbereich Informatik
TU Darmstadt

Aufgrund der Einfachheit und Mächtigkeit des Sprachmodells bietet es sich an, mit Church-Rosser-Sprachen (Church-Rosser Languages, CRL) auch Syntaxanalyse von Programmiersprachen zu betreiben. Allerdings sind die CRL eine Basis für die rekursiv aufzählbaren Sprachen, so daß u.U. die Präfixsprache einer CRL selbst keine CRL sein kann. Das führt unter anderem zu dem Problem, daß die Lokalisierung von Fehlern in nicht akzeptierten Wörtern mehr oder weniger unmöglich wird. Das Sprachmodell ist also noch zu mächtig für die Syntaxanalyse.

In dem Vortrag geht es nun um eine Einschränkung an das einer CRL L zugrunde liegende Church-Rosser-System C. Diese ermöglicht es, mit denselben Alphabeten ein Church-Rosser-System C' zu konstruieren, das unter bestimmten hinreichenden Bedingungen zur Definition einer CRL L' dienen kann, so daß $L' = \mathrm{Pref}(L)$. In diesem Fall wird C' korrekt genannt.

Diese hinreichenden Bedingungen werden vorgestellt. Unter anderem werden dazu erweiterte Thue-Systeme definiert, die außer den eigentlichen Ersetzungsregeln auch noch Zusatzinformationen beinhalten. Mittels dieser Zusatzinformationen ist es nicht nur möglich, die Korrektheit von C' festzustellen, sondern es kann für jedes Präfix $w \in L'$ auch eine Ergänzung w' angegeben werden, so daß $ww' \in L$.

Darüber hinaus ist es wegen des Spiegel-Abschlusses der Church-Rosser-Sprachen auch möglich, in ähnlicher Weise Systeme für Suffix- und Infixsprachen zu betrachten. Allerdings muß ein Church-Rosser-System, das für eine der drei Sprachen zu einem neuen Church-Rosser-System führt, noch lange nicht auch ein korrektes System für die anderen beiden haben. Bei der Frage nach der Korrektheit solcher Präfix-, Suffix- oder Infixsysteme sind auch einige (teilweise offene) Entscheidbarkeitsprobleme zu betrachten.

Des weiteren soll noch kurz auf Church-Rosser-Systeme für einige der typischen Programmiersprachenkonstrukte eingegangen werden. Es werden u.a. für arithmetische Ausdrücke und Fallunterscheidungen Lösungen angegeben, die sowohl zu Präfix- und Suffix- als auch zu Infixsystemen führen. Durch die Verlängerungsinformation, die diese Systeme (bzw. ihre Erweiterungen) liefern, bieten sich evtl. neue Perspektiven für selbstkorrigierende Syntaxanalyseverfahren.

Teilnehmerverzeichnis

- **Jean Berstel**
  Université Marne-la-Vallée
  Institut Gaspard Monge
  Champs-sur-Marne
  F-77454 Marne-la-Vallée
  - France -
  Tel.: +33-1-60957-564
  Fax: +33-1-60957-557
  E-mail: jean.berstel@univ-mlv.fr
  WWW: http://www-igm.univ-mlv.fr/~berstel/

- **Norbert Blum**
  Institut für Informatik IV
  Rheinische Friedrich-Wilhelms-Universität Bonn
  Römerst. 164
  53117 Bonn
  Tel.: 0228/73-4250
  Fax: 0228/73-4571
  E-mail: blum@informatik.uni-bonn.de
  WWW: http://web.informatik.uni-bonn.de/IV/blum/blum.html

- **Henning Bordihn**
  Fakultät für Informatik
  Otto-von-Guericke-Universität Magdeburg
  Postfach 4120
  39016 Magdeburg
  Tel.: 0391/67-12851
  Fax: 0391/67-12018
  E-mail: bordihn@iws.cs.uni-magdeburg.de
  WWW: http://fuzzy.cs.uni-magdeburg.de/~bordihn/

- **Thomas Buchholz**
  Institut für Informatik
  Justus-Liebig-Universität Gießen
  Arndtstr. 2
  35392 Gießen
  Tel.: 0641/99-32143
  Fax: 0641/99-32149
  E-mail: Thomas.Buchholz@informatik.uni-giessen.de
  WWW: http://www.informatik.uni-giessen.de/staff/buchholz

- **Gerhard Buntrock**
  Institut für Informatik
  Justus-Liebig-Universität Gießen
  Arndtstr. 2
  35392 Gießen
  Tel.: 0641/99-32150
  Fax: 0641/99-32149
  E-mail: bunt@tcs.informatik.de
  und:
  Institut für Theoretische Informatik
  Universität zu Lübeck
  Wallstr. 40
  23560 Lübeck
  Tel.: 0451/7030437
  Fax: 0451/7030438
  WWW: http://www.tcs.mu-luebeck.de/pages/buntrock/

- **Jürgen Dassow**
  Fakultät für Informatik
  Otto-von-Guericke-Universität Magdeburg
  Postfach 4120
  39016 Magdeburg
  Tel.: 0391/67-18853
  Fax: 0391/67-12018
  E-mail: dassow@iws.cs.uni-magdeburg.de
  WWW: http://fuzzy.cs.uni-magdeburg.de/theo/dassow_eng.htm

- **Frank Drewes**
  Fachbereich 3
  Universität Bremen
  Postfach 330 440
  28334 Bremen
  Tel.: 0421/218-4335
  Fax: 0421/218-4322
  E-mail: drewes@informatik.uni-bremen.de
  WWW: http://www.informatik.uni-bremen.de/~drewes

- **Rudolf Freund**
  Institut für Computersprachen
  Technische Universität Wien
  Karlsplatz 13
  A-1040 Wien
  - Österreich -
  Tel.: +43-1-58801-18542
  Fax: +43-1-58801-18597
  E-mail: rudi@logic.at
  WWW: http://www.logic.at/staff/rudi/

- **Annegret Habel**
  Fachbereich Informatik
  Abteilung Formale Sprachen
  Carl v. Ossietzky Universität Oldenburg
  Postfach 2503
  26111 Oldenburg
  Tel.: 0441/798-2998 / -2426
  Fax: 0441/798-2965
  E-mail: Annegret.Habel@informatik.uni-oldenburg.de
  WWW: http://www.Informatik.Uni-Oldenburg.de/leute/habel.html

- **Dieter Hofbauer**
  Fachbereich Mathematik/Informatik
  Arbeitsgruppe Theoretische Informatik
  Universität-GH Kassel
  34109 Kassel
  Tel.: 0561/804-4481
  Fax: 0561/804-4008
  E-mail: dieter@theory.informatik.uni-kassel.de
  WWW: http://www.db.informatik.uni-kassel.de/INF_TH/dieter/

- **Maria Huber**
  Fachbereich Mathematik/Informatik
  Universität-GH Kassel
  34109 Kassel
  Tel.: 0561/804-4307
  Fax: 0561/804-4008
  E-mail: maria@theory.informatik.uni-kassel.de
  WWW: http://www.db.informatik.uni-kassel.de/INF_TH/~maria/

- **Andreas Klein**
  Institut für Informatik
  Justus-Liebig-Universität Gießen
  Arndtstr. 2
  35392 Gießen
  Tel.: 0641/99-32153
  Fax: 0641/99-32149
  E-mail: andreas.klein@math.uni-giessen.de

- **Yuji Kobayashi**
  Department of Information Science
  Faculty of Science
  Toho University
  Funabashi 274
  - Japan -
  E-mail: Kobayashi@is.sci.thoho.u.ac.jp

  zur Zeit (bis Ende 1999):
  Fachbereich Mathematik/Informatik
  Universität-GH Kassel
  34109 Kassel
  Tel.: 0561/804-4101
  Fax: 0561/804-4008

- **Hans-Jörg Kreowski**
  Fachbereich Mathematik/Informatik
  Universität Bremen
  Postfach 330 440
  28334 Bremen
  Tel.: 0421/218-1956
  Fax: 0421/218-4322
  E-mail: kreo@informatik.uni-bremen.de
  WWW: http://www.informatik.uni-bremen.de/~kreo

- **Dietrich Kuske**
  Institut für Algebra
  Technische Universität Dresden
  01062 Dresden
  Tel.: 0351/463-3642
  Fax: 0351/463-4235
  E-mail: kuske@math.tu-dresden.de
  WWW: http://www.math.tu-dresden.de/~kuske

- **Martin Kutrib**
  Institut für Informatik
  Justus-Liebig-Universität Gießen
  Arndtstr. 2
  35393 Gießen
  Tel.: 0641/99-32144
  Fax: 0641/99-32149
  E-mail: kutrib@informatik.uni-giessen.de
  WWW: http://www.informatik.uni-giessen.de/staff/kutrib/

- **Klaus-Jörn Lange**
  Universität Tübingen
  WSI
  Sand 13
  72076 Tübingen
  Tel.: 07071/29-77567
  Fax: 07071/29-5061
  E-mail: lange@linux.de
  WWW: http://www-fs.informatik.uni-tuebingen.de

- **Martin Leucker**
  Ahornstr. 55
  52074 Aachen
  Tel.: 0241/80-21210
  Fax: 0241/8888-217
  E-mail: leucker@informatik.rwth-aachen.de
  WWW: http://www-i2.informatik.rwth-aachen.de/leucker

- **Christof Löding**
  Lehrstuhl für Informatik VII
  Rhein.-Westf. Technische Hochschule Aachen
  52056 Aachen
  Tel.: 0241/80-4558
  Fax: 0241/8888-215
  E-mail: loeding@informatik.rwth-aachen.de
  WWW: http://www-i7.informatik.rwth-aachen.de/~loeding/

- **Jan Thomas Löwe**
  Institut für Informatik
  Justus-Liebig-Universität Gießen
  Arndtstr. 2
  35392 Gießen
  Tel.: 0641/99-32153
  Fax: 0641/99-32149
  E-mail: jan-thomas.b.loewe@informatik.uni-giessen.de
  WWW: http://www.informatik.uni-giessen.de/staff/loewe

- **Markus Lohrey**
  Institut für Informatik
  Universität Stuttgart
  Breitwiesenstr. 20-22
  70565 Stuttgart
  Tel.: 0711/781-6408
  Fax: 0711/781-6310
  E-mail: lohreyms@informatik.uni-stuttgart.de
  WWW: http://www.informatik.uni-stuttgart.de/ifi/ti/personen/Lohrey/Lohrey.html

- **Klaus Madlener**
  Fachbereich Informatik
  Universität Kaiserslautern
  Postfach 3049
  67653 Kaiserslautern
  Tel.: 0631/205-2268
  Fax: 0631/205-2156
  E-mail: madlener@informatik.uni-kl.de
  WWW: http://www-madlener.informatik.uni-kl.de/ag-madlener/staff/madlener/madlener.html

- **Gundula Niemann**
  Fachbereich Mathematik/Informatik
  Universität-GH Kassel
  34109 Kassel
  Tel.: 0561/804-4443
  Fax: 0561/804-4008
  E-mail: niemann@theory.informatik.uni-kassel.de
  WWW: http://www.db.informatik.uni-kassel.de/INF_TH/~niemann/

- **Thomas Noll**
  Lehrstuhl für Informatik II
  Rhein.-Westf. Technische Hochschule Aachen
  52056 Aachen
  Tel.: 0241/80-21213
  Fax: 0241/8888-217
  E-mail: noll@informatik.rwth-aachen.de
  WWW: http://www-i2.informatik.rwth-aachen.de/noll/

- **Friedrich Otto**
  Fachbereich Mathematik/Informatik
  Universität-GH Kassel
  34109 Kassel
  Tel.: 0561/804-4573
  Fax: 0561/804-4008
  E-mail: otto@theory.informatik.uni-kassel.de
  WWW: http://www.db.informatik.uni-kassel.de/INF_TH/otto/

- **Holger Petersen**
  Institut für Informatik
  Universität Stuttgart
  Breitwiesenstr. 20-22
  70565 Stuttgart
  Tel.: 0711/781-6451
  Fax: 0711/781-6310
  E-mail: petersen@informatik.uni-stuttgart.de

- **Detlef Plump**
  Fachbereich Mathematik und Informatik
  Universität Bremen
  Postfach 330 440
  28334 Bremen
  Tel.: 0421/218-4854
  Fax: 0421/218-4322
  E-mail: det@informatik.uni-bremen.de

- **Bernd Reichel**
  Fakultät für Informatik
  Otto-von-Guericke-Universität Magdeburg
  Postfach 4120
  39016 Magdeburg
  Tel.: 0391/67-12851
  Fax: 0391/67-12018
  E-mail: reichel@iws.cs.uni-magdeburg.de
  WWW: http://fuzzy.cs.uni-magdeburg.de/~reichel/

- **Birgit Reinert**
  Fachbereich Informatik
  Universität Kaiserslautern
  Postfach 3049
  67653 Kaiserslautern
  Tel.: 0631/205-3344
  E-mail: reinert@informatik.uni-kl.de
  WWW: http://www-madlener.informatik.uni-kl.de/ag-madlener/staff/reinert/reinert.html

- **Andrea Sattler-Klein**
  Wolfsgrubenweg 13a
  67069 Ludwigshafen
  Tel.: 0621/6296408
  E-mail: sattler@informatik.uni-kl.de

- **Helmut Seidl**
  Fachbereich IV - Informatik
  Universität Trier
  54286 Trier
  Tel.: 0651/201-2835
  Fax: 0651/201-2835
  E-mail: seidl@uni-trier.de
  WWW: http://www.informatik.uni-trier.de/~seidl

- **Ludwig Staiger**
  Institut für Informatik
  Universität Halle
  Kurt-Mothes-Str. 1
  06120 Halle
  Tel.: 0345/55-24714
  E-mail: staiger@informatik.uni-halle.de
  WWW: http://www.informatik.uni-halle.de/~staiger/

- **Wolfgang Thomas**
  Lehrstuhl für Informatik VII
  Rhein.-Westf. Technische Hochschule Aachen
  52056 Aachen
  Tel.: 0241/80-21700
  Fax: 0241/8888-215
  E-mail: thomas@informatik.rwth-aachen.de
  WWW: http://www-i7.informatik.rwth-aachen.de

- **Bianca Truthe**
  Dr.-Grosz-Str. 8
  39126 Magdeburg
  E-mail: truthe@csmd.cs.uni-magdeburg.de

- **Heiko Vogler**
  Fakultät Informatik
  Technische Universität Dresden
  Mommsenstr. 13
  01062 Dresden
  Tel.: 0351/463-8232
  E-mail: vogler@orchid.inf.tu-dresden.de
  WWW: http://orchid.inf.tu-dresden.de/gdp/vogler.html

- **Walter Vogler**
  Institut für Informatik
  Universität Augsburg
  86135 Augsburg
  Tel.: 0821/598-2120
  Fax: 0821/598-2200
  E-mail: Walter.Vogler@Informatik.uni-augsburg.de
  WWW: http://www.informatik.uni-augsburg.de/~vogler/

- **Johannes Waldmann**
  Institut für Informatik
  Universität Leipzig
  Augustusplatz 10-11
  04109 Leipzig
  Tel.: 0341/9732-204
  Fax: 0341/9732-209
  E-mail: joe@informatik.uni-leipzig.de
  WWW: http://www.informatik.uni-leipzig.de/~joe/

- **Klaus Wich**
  Alfred-Delp-Str. 1
  34132 Kassel
  Tel.: 0561/804-4242
  Fax: 0561/804-4008
  E-mail: wich@theory.informatik.uni-kassel.de
  WWW: http://www.db.informatik.uni-kassel.de/INF_TH/~wich/

- **Jens Woinowski**
  Fachbereich Informatik
  Fachgruppe AFS
  Technische Universität Darmstadt
  Wilhelminenstr. 7
  64283 Darmstadt
  Tel.: 06151/16-6182 (dienstlich)
  06151/718150 (privat)
  Fax: 06151/16-6185
  E-mail: woinowski@iti.informatik.tu-darmstadt.de
  WWW: http://www.iti.informatik.tu-darmstadt.de/~woinowski/

Nachtrag zum Teilnehmerverzeichnis
9. Theorietag

- **Roman König**
  Fachbereich Informatik
  Universität Erlangen-Nürnberg
  Martensstr. 3
  91058 Erlangen

  Tel.: 09131/85-27921
  Fax: 09131/3-9388
  E-mail: koenig@informatik.uni-erlangen.de
  WWW: http://www1.informatik.uni-erlangen.de/tree/Persons/koenig/