Universität Stuttgart
Fakultät Informatik

3. Theorietag
"Automaten und Formale Sprachen"

Schloß Dagstuhl, 7./8. Oktober 1993

Volker Diekert, Dan Teodosiu
(Herausgeber)

Bericht 1/1994

Vorwort

Die bisherigen Theorietage "Automaten und Formale Sprachen" fanden in Magdeburg (September 1991) und Kiel (Oktober 1992) statt. Im Oktober 1992 wurde die Tradition im Internationalen Begegnungs- und Forschungszentrum Schloß Dagstuhl fortgesetzt. Es nahmen 32 Teilnehmer aus Deutschland, Österreich und Frankreich teil. Das wissenschaftliche Programm bestand aus angemeldeten Beiträgen der Teilnehmer. Sie bilden einen Beweis für vielfältige Thematik der Vorträge. Die Kurzfassungen der Beiträge sind in diesem Bericht abgedruckt.

Ich danke allen Teilnehmern für Ihre interessanten Beiträge und die Bereitschaft zur wissenschaftlichen Diskussion. Ich danke auch der Universität Stuttgart für die Unterstützung und dem Schloß Dagstuhl für die freundliche Aufnahme. Dem nächsten Theorietag in München wünsche ich viel Erfolg.

Ein besonderer Dank gilt meinem Mitherausgeber Dan Teodosiu für die mühevolle Arbeit, diesen Bericht zu erstellen.

Stuttgart, im Januar 1994 Volker Diekert

Inhalt

Vortragsprogramm

Donnerstag, 7. Oktober

9.00 -10.30 Uhr Sitzungsleiter: Volker Diekert

- Mathias Bull

  Stack- und Zähler-Automaten mit Zeigern in Labyrinthen

- Holger Petersen

  Deterministische Zweiweg-Kellerautomaten: Bemerkungen zu Endlosschleifen

- Katja Landskron

  Über die Erzeugung von (un-)endlichen Sprachen bei stochastischen k-limitierten 0L-Systemen

11.00 - 12.15 Uhr Sitzungsleiter: Wolfgang Thomas

- Sebastian Seibert

  Two-dimensional Picture Languages

- Robert Cremanns

  Berechnung von Untergruppendarstellungen aus endlichen Automaten

- Henner Kröger

  Aktivierung zellularer Automaten

14.00 - 15.00 Uhr Fachgruppensitzung mit Wahl der Leitung

16.00 - 18.00 Uhr Sitzungsleiter: Jürgen Dassow

- Bernd Reichel

  Eine Bemerkung über Linksableitung indisch paralleler Grammatiken

- Stefan Skalla

  Zur Anzahl der aktiven Nichtterminale in kooperierenden Grammatiksystemen

- Rudolf Freund

  Kooperierende Systeme von Array-Grammatiken

- Torsten Roßnick

  Über Fragen des Determinismus bei endwachsenden fadenförmigen Systemen

20.00 - 21.30 Uhr Sitzungsleiter: Klaus-Jörn Lange

- Matthias Jantzen

  Neue Anwendungen von Ergebnissen über Petrinetze auf Matrix-Sprachen

- Franz J. Brandenburg

  The solvability of the Membership Problem for Context-Free Grammars with Regular Control Sets

Freitag, 8. Oktober

9.00 -10.30 Uhr Sitzungsleiter: Franz J. Brandenburg

- Ludwig Staiger

  On Syntactic Congruences for ω–languages

- Henning Fernau

  Bewertungen regulärer Ausdrücke, Zusammenhänge mit strenger Eindeutigkeit regulärer Ausdrücke sowie Anwendungen bei der Berechnung der Hausdorff-Dimension von Fraktalen, die mit regulären Ausdrücken beschrieben werden

- Helmut Seidl

  Least Solutions of Equations over $\mathcal{N}$

11.00 - 12.15 Uhr Sitzungsleiter: Volker Diekert

- Gerhard Buntrock

  Wachsend kontextsensitive Sprachen und Automaten

- Olaf Burkkart

  Pushdown Prozesse: Parallele Komposition und Model-Checking

- Markus Holzer

  Das Nichtleerheitsproblem für alternierende endliche Automaten

13.00 - 14.00 Uhr Sitzungsleiter: Ludwig Staiger

- Maria Huber

  Reguläre Grundnormalformsprachen und Linearisierung von Termersetzungssystemen

- Dieter Hofbauer

  Reduzierbarkeit, Grundreduzierbarkeit und Testmengen

The solvability of the Membership Problem for Context-Free Grammars with Regular Control Sets

Franz J. Brandenburg
Lehrstuhl für Informatik
Universität Passau

Context-free grammars with regular control sets are well-known systems of regulated rewriting. They are extensions of context-free grammars and are equivalent, e.g., to programmed grammars or matrix grammars.

Here we investigate the role of erasing productions and of chain productions under regular control sets.

By classical proof techniques we obtain the following result.

Lemma For every context-free grammar G and control set C there exist finitely many homomorphisms g, h and $h_i, i = 1, \ldots, n$ and a (nonerasing, nonerasing and chain-free or) context-free grammar in Chomsky normal form G' such that $L(G, C) = L(G', C')$ where $C' = g(h^{-1}(C) \cap \bigcap_{h_i}^{-1}(D_1'))$ with the semi-Dyck set over one pair of parenthesis D_1'.

Hence, nonerasing and chain-free productions are obtained at the cost of more complex control sets, which are related to Greibach's class PBLIND [G78]. PBLIND is the class of languages accepted by nondeterministic multitape Turing machines whose worktapes are partially blind counters. Equivalently, PBLIND is the class of Petri net languages, or is the smallest intersection closed full trio containing the semi-Dyck set D_1'. Due to the decidability of the reachability problem for Petri nets PBLIND is a class of recursive sets.

Corollary If C is a regular set or if C is in PBLIND, then C' is in PBLIND and $L(G, C)$ is recursive.

Thus we can conclude the decidability of the membership problem for contextfree grammars with regular control sets, or equivalently of matrix or programmed grammars. This problem is stated open e.g., in the textbooks by Salomaa [S73] and by Dassow and Paun [DP89].

It has been solved by other methods by Gonczarkowski and Warmuth [GW79]. The decidability is implicitly stated in the textbook of Dassow and Paun [DP 89] as was pointed out by M. Jantzen (see this report).

Theorem The membership problem for context-free grammars with regular control sets is decidable.

As a consequence the corresponding class of matrix languages is not closed under intersection and complement. Using the machine characterization of languages from the class PBLIND we directly obtain a simulation in terms of grammars with regular control sets. This relates Petri net language to matrix languages.

Lemma For every language L in PBLIND there is a context-free grammar G and a regular control set R such that $L = L(G, R)$.

Some of the results from above can be generalized. E.g. if G is a nonerasing and chain-free context-free grammar and C is recursive, then $L(G, C)$ is recursive. This follows from the fact the length of a derivation of a word w is bounded by $2|w| - 1$. However, every recursively enumerable set can be generated by a nonerasing regular grammar with a contextsensitive control set.

Moreover, if partially blind multicounter machines are extended by a single one-reversal pushdown stack, then such machines can be simulated by context-free grammars with regular control sets. Whether or not this holds with the extension by an unrestricted stack is open.

Acknowledgement I wish to thank the participants of the Theorietag and in particular M. Jantzen for pointing out a serious error in an earlier draft concerning the above open problem.

References

[DP89] J. Dassow, G. Paun. Regulated Rewriting in Formal Languge Theory, Springer (1989)

[G78] S.A. Greibach. Remarks on blind and partially blind one-way multicounter machines, Theoret. Comput. Sci. 7 (1978), 311-324

[GW79] J. Gonczarowski, M. Warmuth. Scattered Versus Context-Sensitive Rewriting, Acta Informatica 27 (1989), 81-95

[S73] A. Salomaa. Formal Languages, Academic Press (1973)

Stack- und Zähler-Automaten mit Zeigern in Labyrinthen

Mathias Bull
FB Informatik
Universität Rostock

In der Labyrinth-Theorie wird die Leistungsfähigkeit von Automaten bei Absuch- und Erkennungsprozessen auf ungerichteten, zusammenhängenden Graphen mit Kompaßsystemen (C-Graphen) und Rotationssystemen (R-Graphen) untersucht. Eine Charakterisierung erfolgt entweder durch den Entwurf eines Absuchprozesses/-algorithmus' oder durch die Konstruktion eines nicht absuchbaren Graphen, einer sogenannten Falle, für einen Automaten des jeweils betrachteten Automaten- sowie Labyrinthtyps.

Seit der Veröffentlichung eines Absuchalgorithmus' für endliche, zweidimensionale C-Graphen durch einen (1-Zeiger, 1-Zähler)-Automaten von A. Hemmerling (MFCS'86) war die analoge Fragestellung für planierte R-Graphen offen. Für Stackautomaten fehlten bisher jegliche Aussagen.

Die folgenden neuen Resultate werden vorgestellt.
Fallenkonstruktionen:

- Zu jedem (1-Zeiger,1-Zähler)-Automaten gibt es eine planierte R-Falle.
- Zu jedem 1-Stack-Automaten gibt es eine planierte R-Falle.

Absuchprozeß/-algorithmus:

- Es gibt einen (1-Zeiger,1-nichtlöschenden-Stack)-Automaten, der alle zusammenhängenden R-Graphen absucht, wobei er auf endlichen hält.

Wachsend kontextsensitive Sprachen und Automaten

Gerhard Buntrock
Institut für Informatik
Universität Würzburg

Die Sprachklasse der wachsend kontextsensitiven Sprachen ist definiert durch Grammatiken, deren Produktionen nur der Einschränkung unterliegen, daß ihre rechte Seite stets länger als ihre linke ist. Die einzige Ausnahme ist, wenn links nur das Startsymbol steht; dann darf das Startsymbol nicht auf der rechten Seite auftauchen. Diese Klasse ist von besonderer Bedeutung, weil sie ausdrucksstärker als die Klasse der kontextfreien Sprachen ist, aber ihr komplexitätstheoretischer Abschluß unter logarithmisch platzbeschränkten Reduktionen dem der kontextfreien gleicht.

Verwenden wir die Charakterisierung wachsend kontextsensitiver Sprachen durch Grammatiken, die nur bezüglich einer Gewichtung ihrer Symbole wachsen [BL92], läßt sich zeigen, daß auch eine Verallgemeinerung des Kellerautomaten diese Klasse charakterisiert. Ähnlich der Gewichtung der Grammatiksymbole kann man eine Gewichtung der Symbole, mit denen ein Automat arbeitet (Eingabe- und Arbeitsalphabet sowie die Zustände), vornehmen. Wenn nun die so entstehenden Gewichte der Konfigurationen stets abnehmen müssen, zeigt sich, daß diese Einschränkung bei *normalen* Kellerautomaten keine Auswirkung auf ihre Leistungsfähigkeit hat. Für einen Kellerautomaten mit zwei Kellern bedeutet das aber, daß er anstelle aller rekursiv aufzählbaren Sprachen nur noch wachsend kontextsensitive Sprachen akzeptiert. Darf in jedem Schritt das Gewicht der Konfigurationen auch gleich bleiben, so können kontextsensitive Sprachen akzeptiert werden.

Diese Automatencharakterisierung führt zu einer neuen interessanten Klasse: die der deterministisch wachsend kontextsensitiven Sprachen, die eine Erweiterung der deterministisch kontextfreien Sprachen sind.

References

[BL92] Gerhard Buntrock and Krzysztof Loryś. On growing context-sensitive languages. In *Proc. of 19th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 623 of *LNCS*, pages 77–88. Springer, 1992.

Pushdown Prozesse: Parallele Komposition und Model-Checking

Olaf Burkart
Department of Computer Science II
RWTH-Aachen
(in Zusammenarbeit mit Bernhard Steffen, Universität Passau)

In den letzten Jahren hat sich Model Checking als ein nützliches Werkzeug zur Analyse von nebenläufigen Prozessen entwickelt. Während Model Checking für Systeme mit endlicher Zustandsanzahl bereits wohl bekannt ist [EL86, Cle90, CS92], ist die Theorie für Systeme mit unendlichem Zustandsraum ein aktueller Gegenstand der Forschung. Bradfield und Stirling haben in [Bra91, BS91] ein tableau-basiertes Model-Check -Verfahren für allgemeine Systeme mit unendlicher Zustandsanzahl vorgestellt, welches jedoch nicht effektiv ist. Aus diesem Grunde hat sich viel Arbeit auf die *kontext-freien Prozesse*, einer Unterklasse der Systeme mit unendlichem Zustandsraum, konzentriert. So wurde in [BS92] ein iterativer Model-Check Algorithmus entwickelt, welcher den alternierungsfreien Teil des modalen Mu-Kalküls [Koz83] entscheidet. Weiterhin wurde in [HS93] gezeigt, wie dies mit tableau-basierten Methoden durchgeführt werden kann.

Leider sind kontext-freie Prozesse jedoch nicht unter paralleler Komposition abgeschlossen. So können beliebige Turingmaschinen bereits durch die parallele Komposition von zwei kontext-freien Prozessen modelliert werden. Auch die parallele Komposition eines kontext-freien Prozeßes mit einem endlichen System ergibt im allgemeinen keinen kontext-freien Prozeß. Kontext-freie Prozesse sind daher nur beschränkt geeignet für die Konstruktion von verteilten Systemen.

Wir betrachten eine strikte Verallgemeinerung von kontext-freien Prozessen, die *Pushdown Prozesse*, und zeigen, daß diese Klasse von Prozessen

- abgeschlossen ist unter paralleler Komposition mit endlichen Systemen; (Wir stellen eine Art Expansionstheorem im Sinne von Milner [Mil89] vor.)
- mit Hilfe einer Variante des in [BS92] vorgestellten Model Checkers höherer Ordnung, welche auf Kellerautomaten arbeitet, automatisch analysiert werden können.

Beide Resultate benutzen eine endliche Beschreibung von Pushdown Prozessen durch Kellerautomaten. Intuitiv besagt unser Expansionstheorem, daß sich die parallele Komposition eines Pushdown Prozeßes mit einem endlichen System aus der Synchronisation des Kellers mit dem Produkt aus Zustandskontrolle des zugrundeliegenden Kellerautomaten und endlichem System ergibt. Wir zeigen, daß das resultierende Problem der "Repräsentationsexplosion" nicht schlimmer als im bekannten "Zustandsexplosionsproblem" für endliche Systeme ist.

Unser iterativer Model Check Algorithmus entscheidet den alternierungsfreien Teil des modalen Mu-Kalküls für Pushdown Prozesse. Wie im Fall der kontext-freien Prozesse, basiert der Algorithmus auf einer Variante "höherer Ordnung" der Standard Model-Check Techniken. Er bestimmt *property transformer* für jedes *Fragment* des Kellerautomaten. Diese beschreiben die Menge der Formeln, welche am Startzustand eines Fragmentes gelten, relativ zu den Mengen von Formeln, welche an den Endzuständen des Fragments gelten. Hierbei bestimmt die Anzahl der Endzustände eines Fragments, welche identisch mit der Anzahl der Zustände des Kellerautomaten ist, die Stelligkeit des zugehörigen "property transformers". Dieser Algorithmus stellt daher eine Verallgemeinerung des Model-Checkers aus [BS92] dar, welcher nur unäre "property transformer" benutzt. Nach der Bestimmung der "property transformer" wird das Model-Check Problem einfach dadurch gelöst, daß überprüft wird, ob die gegebene Formel in der Menge enthalten ist, die man durch Applikation des mit dem initialen Fragments assoziierten "property transformers" auf die Mengen von Formeln, welche an den Endzuständen gelten, erhält.

Daß Model-Checking für Pushdown Prozesse entscheidbar ist, folgt ebenfalls aus der Entscheidbarkeit der monadischen Logik zweiter Stufe für diese Klasse von Prozessen [MS85], da der modale Mu-Kalkül in dieser

Logik ausgedrückt werden kann. Neu ist jedoch die Effizienz des Algorithmus: nur die Größe der Zustandskontrolle des Kellerautomaten und der betrachteten Formel sind kritisch.

References

[Bra91] J.C. Bradfield. *Verifying Temporal Properties of Systems with Applications to Petri Nets.* PhD thesis, University of Edinburgh, 1991.

[BS91] J.C. Bradfield and C. Stirling. Local Model Checking for Infinite State Spaces. Technical Report ECS-LFCS-90-115, LFCS, Jun 1991.

[BS92] O. Burkart and B. Steffen. Model Checking for Context-Free Processes. In *CONCUR '92, LNCS 630*, pages 123–137. Springer, 1992.

[Cle90] R. Cleaveland. Tableau-Based Model Checking in the Propositional Mu-Calculus. *Acta Informatica*, 27:725–747, 1990.

[CS92] R. Cleaveland and B. Steffen. A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. In *CAV '91, LNCS 575*, pages 48–58. Springer, 1992.

[EL86] E.A. Emerson and C.-L. Lei. Efficient Model Checking in Fragments of the Propositional Mu-Calculus. In *Proc. 1th Annual Symp. on Logic in Computer Science*, pages 267–278. IEEE Computer Society Press, 1986.

[HS93] H. Hungar and B. Steffen. Local Model-Checking for Context-Free Processes. In *ICALP '93, LNCS 700*, pages 593–605, 1993.

[Koz83] D. Kozen. Results on the Propositional μ-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice–Hall, 1989.

[MS85] D.E. Muller and P.E. Schupp. The Theory of Ends, Pushdown Automata, and Second-Order Logic. *Theoretical Computer Science*, 37:51–75, 1985.

Berechnung von Untergruppendarstellungen aus endlichen Automaten

Robert Cremanns
Fachbereich Mathematik, FG Informatik
Gesamthochschule Kassel

Endliche Wortersetzungssysteme können benutzt werden um Gruppen darzustellen. Dabei interessieren insbesondere kanonische Wortersetzungssysteme, denn für kanonische Wortersetzungssysteme sind viele Entscheidungsprobleme entscheidbar, z.B. das Wortproblem.

Wir betrachten das Problem, aus einem endlichen, kanonischen Wortersetzungssystem, das eine Gruppe G darstellt, und einer endlichen Menge U von Wörtern eine Darstellung für die von U erzeugte Untergruppe H von G zu bestimmen. Wir stellen ein Verfahren zur Lösung dieses Problems vor und wenden es auf verschiedene Klassen von kanonischen Darstellungen von kontextfreien Gruppen an. Das Verfahren ist in vier Schritte unterteilt.

Im ersten Schritt wird ein endlicher Automat A konstruiert, der eine Sprache $L \subseteq \langle U \rangle$ akzeptiert, so daß jedes Element der Untergruppe H durch mindestens ein Wort in L repräsentiert wird. Hierbei bezeichnet $\langle U \rangle$ die Menge aller Wörter, die Elemente in H beschreiben. Sei R das betrachtete Wortersetzungssystem. Im zweiten Schritt konstruieren wir einen Kellerautomaten P, der Reduktionen bzgl. R durchführt. Bei Eingabe w ist die Menge der End-Kellerinhalte $SC_P(w)$ eine Menge von R-Nachfolgern von w. Im dritten Schritt wird ein Algorithmus angewendet, der bei Eingabe A und P einen endlichen Automaten bestimmt, der die Sprache $SC_P(L)$ der End-Kellerinhalte von P bei Eingaben aus L akzeptiert. Die von diesem Automaten akzeptierte Sprache erfüllt bestimmte Abschlußeigenschaften, so daß wir im vierten Schritt daraus eine Darstellung für H ableiten können.

Die Schritte 1, 3 und 4 sind mit polynomialem Zeitaufwand durchführbar. Die Zeitkomplexität des zweiten Schritts ist von der betrachteten Klasse von Darstellungen von kontextfreien Gruppen abhängig.

Bewertungen regulärer Ausdrücke, Zusammenhänge mit strenger Eindeutigkeit regulärer Ausdrücke sowie Anwendungen bei der Berechnung der Hausdorff-Dimension von Fraktalen, die mit regulären Ausdrücken beschrieben werden

Henning Fernau
Lehrstuhl für Informatik
Universität Karlsruhe (TH)

Brüggemann-Klein u.a. hat jüngst sogenannte streng eindeutige reguläre Ausdrücke näher untersucht [1]. Der Begriff der strengen Eindeutigkeit läßt sich mit Hilfe von Bewertungen kennzeichnen. Bewertungen β sind Monoidmorphismen von $(\Sigma_n^*, \cdot, \lambda)$ nach $((0,\infty), \cdot, \lambda)$, die sich durch $\beta(L) = \sum_{w \in L} \beta(w) \in [0,\infty]$ leicht auf Sprachen erweitern lassen. Entlang der rekursiven Definition regulärer Ausdrücke läßt sich nun rekursiv die Bewertung von regulären Ausdrücken β_R definieren: $\beta_R(w) = \beta(w)$ für $w \in \Sigma_n^*$, $\beta_R(R_1 R_2) = \beta_R(R_1)\beta_R(R_2)$, $\beta_R(R_1 \cup R_2) = \beta_R(R_1) + \beta_R(R_2)$, $\beta_R(R_1^*) = \sum_{i=0}^{\infty}(\beta_R(R_1))^i$. Ein Ausdruck R ist streng eindeutig genau dann, wenn für eine Bewertung β mit $\beta_R(R) < \infty$ die Zahl $\beta_R(R)$ mit der Bewertung $\beta([R])$ der von R beschriebenen Sprache $[R]$ übereinstimmt.

Bewertungen sind ein Konzept, das zunächst als Hilfsmittel zur Bestimmung von Hausdorff-Dimensionen von formalsprachlich definierten Fraktalen eingeführt wurde [3, 2]. So läßt sich das obige Ergebnis auch zur einfachen Dimensionsbestimmung von Fraktalen, die durch streng eindeutige reguläre Ausdrücke beschrieben werden, benutzen.

References

[1] A. Brüggemann-Klein. Regular expressions into finite automata. In *LATIN'92*, Band 483 aus der Reihe *LNCS*, Seiten 87–98, 1992.

[2] H. Fernau. Valuations of languages, with applications to fractal geometry. Submitted for publication, Sept. 1993.

[3] H. Fernau. *Varianten Iterierter Funktionensysteme und Methoden der Formalen Sprachen.* Dissertation, Universität Karlsruhe (TH) (Germany), 1993.

Kooperierende Systeme von Array-Grammatiken

Rudolf Freund
Technische Universität Wien

Jürgen Dassow
Fakultät für Informatik
Universität Magdeburg

Gheorghe Păun [1]
Institut für Mathematik
Rumänische Akademie der Wissenschaften

Wir untersuchen den Effekt kooperierender Systeme bei der Erzeugung von Array-Sprachen mittels kontextfreier Array-Grammatiken. Erwartungsgemäß ist die Erzeugungskraft kooperierender Systeme kontextfreier Array-Grammatiken (mit einer vorgegebenen festen Anzahl von Ableitungsschritten, mit einer Anzahl von Ableitungsschritten größer oder gleich einer vorgegebenen Zahl oder mit der maximal möglichen Anzahl von Ableitungsschritten in der aktivierten Komponente) größer als die Erzeugungskraft einfacher kontextfreier Array-Grammatiken. Das gleiche Resultat erhält man auch für Systeme regulärer Array-Grammatiken, was im Gegensatz zu den für (kooperierende Systeme von) String-Grammatiken erzielten Ergebnissen steht.

Arrays und Array-Grammatiken

V^{2+} bezeichnet die Menge der zweidimensionalen Arrays über dem Alphabet V, also die Menge aller Patterns, die man durch Markierung endlich vieler Einheitsquadrate in der Ebene mit Symbolen aus V erhält (die restlichen Einheitsquadrate sind mit dem Blanksymbol # markiert); Teilmengen von V^{2+} heißen *Array-Sprachen*. Eine *isometrische Array-Grammatik* ist ein 5-Tupel $G = (N, T, S, P, \#)$, wobei N, T disjunkte Alphabete sind, $S \in N$, # das Blanksymbol und P eine endliche Menge

[1] Gefördert von der Alexander von Humboldt-Stiftung

von Array-Produktionen $\alpha \to \beta$ ist, wobei α, β endliche Patterns über $N \cup T \cup \{\#\}$ sind, welche den folgenden Bedingungen genügen:

1. Die Umrisse von α und β sind identisch, i.e. sie sind *isometrisch*.
2. α enthält mindestens ein Symbol aus N (ein *Nonterminal*).
3. Die Elemente aus T (*Terminale*), die in α vorkommen, werden in β nicht verändert.

Für eine isometrische Array-Grammatik $G = (N, T, S, P, \#)$ definieren wir die Ableitungsrelation $x \Longrightarrow y$ folgendermaßen:
Für $x, y \in (N \cup T \cup \{\#\})^{2+}$ heißt x ableitbar aus y mittels $\alpha \to \beta \in P$, falls α ein Teil-Pattern in x ist und man y aus x durch Ersetzen des Teil-Patterns α in x durch β (α und β sind ja isometrisch) erhält. Die reflexive und transitive Hülle von $\Longrightarrow$ wird mit $\Longrightarrow^*$ bezeichnet; die von G erzeugte Array-Sprache ist definiert durch

$$L(G) = \{x \in T^{2+} \mid S \Longrightarrow^* x\}.$$

Eine isometrische Array-Grammatik heißt *monoton*, falls in allen Array-Produktionen $\alpha \to \beta$ Symbole aus $(N \cup T)$ in α nicht durch # in β ersetzt werden; G heißt *kontextfrei*, falls überdies in jeder Array-Produktion α nur aus genau einem Nonterminal und einigen Blanksymbolen # besteht; sind alle Array-Produktionen von G von einer der folgenden Formen, dann nennt man G *regulär* (A, B Nonterminale, a ein Terminal):

$$\# A \to B\, a,\ A\, \# \to a\, B,\ \begin{matrix}\# \\ A\end{matrix} \to \begin{matrix}B \\ a\end{matrix},\ \begin{matrix}A \\ \#\end{matrix} \to \begin{matrix}a \\ B\end{matrix}, A \to a.$$

Bezeichnet man mit $IA, MA, CFA, REGA$ die Familien von Array-Sprachen, die von allgemeinen, monotonen, kontextfreien and regulären isometrischen Array-Grammatiken erzeugt werden, so bilden diese Familien von Array-Sprachen eine Chomsky-Hierarchie[2]:

$$REGA \subset CFA \subset MA \subset IA.$$

Kooperierende Systeme

Ein *kooperierendes System von Array-Grammatiken* (vom Typ X, $X \in \{REGA, CFA\}$, und vom Grad n, $n \geq 1$) ist ein Konstrukt

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \#),$$

wobei N, T disjunkte Alphabete sind, $S \in N$ und $P_1, P_2, \ldots, P_n$ endliche Mengen kontextfreier bzw. regulärer Array-Produktionen über $N \cup T$ sind. Für jedes i, $1 \leq i \leq n$, betrachten wir die übliche Ableitungsrelation $\Longrightarrow_{P_i}$,

$$x \Longrightarrow_{P_i} y \text{ gdw. } x = x_1 A x_2, y = x_1 z x_2, x_1, x_2 \in (N \cup T)^*, A \to z \in P_i,$$

und definieren beliebige Ableitungen ($\Longrightarrow^*_{P_i}$) sowie, für ein gegebenes $k \geq 1$, Ableitungen mit genau k Schritten, ($\Longrightarrow^{=k}_{P_i}$), mit mindestens beziehungsweise höchstens k Schritten ($\Longrightarrow^{\geq k}_{P_i}, \Longrightarrow^{\leq k}_{P_i}$), sowie die maximale Ableitungsrelation,

$$x \Longrightarrow^t_{P_i} y \text{ gdw. } x \Longrightarrow^*_{P_i} y \text{ und für kein } z \in (N \cup T)^* \; y \Longrightarrow_{P_i} z.$$

Sei nun $F = \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$. Für ein System Γ and $f \in F$ ist die durch Γ im Ableitungsmodus f erzeugte Array-Sprache definiert durch

$$\begin{aligned} L_f(\Gamma) = \{x \in T^* \mid S \Longrightarrow^f_{P_{i_1}} x_1 \Longrightarrow^f_{P_{i_2}} x_2 \ldots \Longrightarrow^f_{P_{i_m}} x_m = x, \\ m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}. \end{aligned}$$

Die Familie von Array-Sprachen, die von kooperierenden Systemen von Array-Grammatiken mit höchstens n bzw. beliebig vielen Komponenten vom Typ X im Ableitungsmodus f erzeugt werden, bezeichnen wir mit $CD_n(X, f)$, $n \geq 1$, bzw. $CD(X, f)$, $X \in \{REGA, CFA\}$, $f \in F$.

Ergebnisse

1. *Für* $n \geq 1$, $f \in \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$ und $X \in \{REGA, CFA\}$ gilt

$$CD_n(X, f) = CD(X, f) = X.$$

2. *Sei* $k, k' \geq 1$.
   i) $CD_n(REGA, = k) = CD_1(REGA, = k)$ *gilt für alle* $n \geq 1$.
   ii) Ist k' *ein Teiler von* k, *so ist*
   $CD_1(REGA, = k)$ *echt enthalten in* $CD_1(REGA, = k')$.
   iii) Ist weder k *ein Teiler von* k' *noch* k' *ein Teiler von* k, *so sind die Familien von Array-Sprachen* $CD_1(REGA, = k)$ *und* $CD_1(REGA, = k')$ *unvergleichbar, aber nicht disjunkt.*

3. *Für* $n \geq 1$ *und* $1 \leq k' < k$ *gilt*

$$CD_n(REGA, \geq k) \subset CD_n(REGA, \geq k').$$

4. *Für* $n \geq 2$ *und* $f \in \{= k \mid k \geq 2\} \cup \{\geq k \mid k \geq 2\}$ *gilt*

$$CFA = CD_1(CFA, f) \subset CD_n(CFA, f).$$

5. *Sei* $m \geq 2, n \geq 3$. *Dann gilt*

$$REGA = CD_1(REGA, t) \subset CD_2(REGA, t) = CD_m(REGA, t)$$

und

$$\begin{aligned} CFA &= CD_1(CFA, t) \subset CD_2(CFA, t) \\ &\subset CD_3(CFA, t) = CD_n(CFA, t). \end{aligned}$$

Literatur

1. E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems*, Gordon and Breach, 1993.

2. C. R. Cook, P. S. P. Wang, A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing* **8** (1978), pp. 144 - 152.

3. D. L. Milgram, A. Rosenfeld, Array automata and array grammars, *Inform. Processing '71*, North-Holland, 1972, pp. 69 – 74.

4. Y. Yamamoto, K. Morita, K. Sugata, Context-sensitivity of two-dimensional regular array grammars, pp. 17 – 41; in P. S. P. Wang (ed.), *Array Grammars, Patterns and Recognizers*, World Scientific Series in Computer Science **18**, World Scientific, 1989.

Reduzierbarkeit, Grundreduzierbarkeit und Testmengen

Dieter Hofbauer
Fachbereich Informatik
TU Berlin

Abstract

In der Theorie der Termersetzung spielen die Baumsprachen der reduzierbaren bzw. der irreduzierbaren Grundterme eine wichtige Rolle. Von besonderem Interesse sind die *grundreduzierbaren* Terme (mit Variablen), also die Terme, deren Grundinstanzen sämtlich reduzierbar sind.

Für endliche linkslineare Termersetzungssysteme ist die Sprache der reduzierbaren Grundterme immer regulär und die Grundreduzierbarkeit somit entscheidbar. Aber auch im nichtlinearen Fall bleibt die Grundreduzierbarkeit entscheidbar. Es wird gezeigt, daß immer endliche *Testmengen* existieren, so daß ein Term genau dann grundreduzierbar ist, wenn die – endlich vielen – Instanzierungen des Terms durch Terme aus der Testmenge reduzierbar sind.

Grundreduzierbarkeit hat sich als ein Schlüsselkonzept bei der Verifikation wichtiger Eigenschaften von Termersetzungssystemen erwiesen, so etwa für die Überprüfung der vollständigen Definiertheit von Operatoren bezüglich gegebener Konstruktoren (*sufficient completeness*) oder für das Beweisen induktiv gültiger Gleichungen (*"inductionless induction"*).

Sei R ein Termersetzungssystem über der Signatur Σ. Wir bezeichnen mit $Red(R)$ die Menge der Grundterme über Σ, die mit R reduzierbar sind, und mit $Nf(R)$ die Komplementmenge, also die Grundterme in Normalform bezüglich R. $Ground(t)$ sei die Menge aller Grundinstanzen eines Terms $t \in \mathcal{T}_\Sigma(X)$. Ein Term $t \in \mathcal{T}_\Sigma(X)$ ist *grundreduzierbar mit* R, wenn alle Grundinstanzen von t mit R reduzierbar sind, d.h. falls

$Ground(t) \subseteq Red(R)$ gilt.

Für linkslineare Systeme R und lineare Terme t ist die Grundreduzierbarkeit leicht zu entscheiden: in diesem Fall sind sowohl $Ground(t)$ als auch $Red(R)$ immer reguläre Baumsprachen, $Ground(t) \subseteq Red(R)$ also entscheidbar. Aber auch im nichtlinearen Fall bleibt die Entscheidbarkeit erhalten, wie Plaisted [13] sowie Kapur, Narendran und Zhang [8] gezeigt haben. Hierbei wird folgendes Problem gelöst:

- GEGEBEN: ein endliches System R über Σ und ein Term t.
- GESUCHT: eine endliche Menge $T \subseteq \mathcal{T}_\Sigma$ so, daß t genau dann grundreduzierbar mit R ist, wenn alle T-Instanzen von t mit R reduzierbar sind.

Eine T-Instanz von t ist hier ein Term, der aus t durch Substitution aller Variablen durch Terme aus T entsteht.

Weil der Grundreduzierbarkeitstest damit auf endlich viele Reduzierbarkeitstests zurückgeführt wird, heißt T auch *Testmenge* für R und t. Da Testmengen im allgemeinen sehr groß werden können, ist ein Nachteil dieses Ansatzes, daß die Testmenge vom Term t abhängt, also für jeden Test neu berechnet werden muß. Nun läßt sich aber tatsächlich eine Testmenge angeben, die nur von R abhängt; hierbei enthalten die Terme der Testmenge im allgemeinen Variablen. Wir zeigen [6, 4, 5], daß auch das folgende Problem lösbar ist:

- GEGEBEN: ein endliches System R über Σ.
- GESUCHT: eine endliche Menge $T \subseteq \mathcal{T}_\Sigma(X)$ so, daß t genau dann grundreduzierbar mit R ist, wenn alle T-Instanzen von t mit R reduzierbar sind.

Diese Testmengen sind durch Kounalis [5] inspiriert; sein ursprünglicher Ansatz hat sich allerdings als inkorrekt erwiesen. Nachdem von Huber [6] gezeigt wurde, wie sich dies reparieren läßt, hat auch Kounalis eine korrigierte Version vorgestellt [10].

Allgemeiner zeigen wir, daß sich alle Mengen T, die

(1) *vollständig* für R,

(2) *expandiert* bezüglich R und

(3) *typisch* für R

sind, als Testmengen eignen, und wie solche Mengen berechnet werden können. Dabei ist die Vollständigkeit von T (definiert durch $Nf(R) \subseteq Ground(T)$) für die Korrektheit der einen Richtung des Test zuständig: Ist t nicht grundreduzierbar, hat t also eine irreduzierbare Grundinstanz $t\gamma$, dann existiert auch eine T-Instanz $t\tau$, die $t\gamma$ als Instanz hat, die also selber irreduzierbar ist. Die Expandiertheit ist eine eher technische, leicht zu erfüllende Bedingung.

Dagegen bildet Bedingung (3), die für linkslineare Systeme überflüssig ist, den Kernpunkt unseres Ansatzes. Hier wird formuliert, daß nicht nur alle irreduzierbaren Grundterme als Instanzen eines Terms in der Testmenge repräsentiert sind – Bedingung (1) – sondern daß die Testmenge darüber hinaus auch noch die „Nichtlinearitäten" von R widerspiegelt.

Ein Term t heißt *typisch* für R, wenn für jede Variable x in t eine unendliche Menge $G_x \subseteq \mathcal{T}_\Sigma$ existiert, so daß $t\gamma \in Nf(R)$ für alle Substitutionen γ, wobei $x\gamma \in G_x$ für alle Variablen x in t. Entsprechend heißt eine Menge T typisch für R, wenn alle t in T typisch für R sind.

Diese Bedingung garantiert nun umgekehrt, daß jede irreduzierbare T-Instanz eines Terms t auch eine irreduzierbare Grundinstanz besitzt. Terme mit irreduzierbaren T-instanzen können also nicht grundreduzierbar sein.

Ausführlich gehen wir abschließend darauf ein, wie sich durch eine genaue Beweisanalyse im allgemeinen deutlich kleinere Testmengen finden lassen; unter worst-case-Betrachtungen ergeben sich allerdings keine Unterschiede in der Größenordnung.

References

[1] D. Hofbauer und M. Huber. Computing linearizations using test sets. In *Proc. 3rd CTRS*, LNCS 656, S. 287–301, 1992.

[2] D. Hofbauer und M. Huber. Linearizing term rewriting systems using test sets. Erscheint im Journal of Symbolic Computation.

[3] M. Huber. Testmengen für Grundreduzierbarkeit: Konstruktionen, Komplikationen, Korollare. Diplomarbeit, Technische Universität Berlin, 1991.

[4] D. Kapur, P. Narendran und H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.

[5] E. Kounalis. Testing for inductive (co)-reducibility. In *Proc. 15th CAAP*, LNCS 431, S. 221–238, 1990.

[6] E. Kounalis. Testing for the ground (co-)reducibility property in term-rewriting systems. *Theoretical Computer Science*, 106:87–117, 1992.

[7] D. A. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65(2/3):182–215, 1985.

Das Nichtleerheitsproblem für alternierende endliche Automaten [2]

Markus Holzer
Institut für Informatik
Technische Universität München

Jones zeigte im Jahre 1975 [3], daß das Nichtleerheitsproblem für deterministische und nichtdeterministische endliche Automaten $NSPACE(\log n)$-vollständig ist.

In weiterer Folge wurden Einschränkungen und Verallgemeinerungen des Nichtleerheitsproblems für deterministische und nichtdeterministische endliche Automaten von Galil [2], Kozen [4] und Lange und Rossmanith [5] betrachtet. Hierbei gelang es natürliche vollständige Probleme für die Klassen $\mathcal{NP}$, $\mathcal{PSPACE}$ und Klassen mit eingschränktem Nichtdeterminismus anzugeben.

Angesichts der Ergebnisse für deterministische und nichtdeterministische endliche Automaten, stellt sich die Frage welche Komplexität das Nichtleerheitsproblem für alternierende endliche Automaten [1], einem Maschinenmodell das ebenfalls die regulären Sprachen charakterisieren, hat.

Es ergeben sich natürliche vollständige Probleme für die Klassen $\mathcal{P}$, $\mathcal{NP}$ und $\mathcal{PSPACE}$. Für das Nichtleerheitsproblem für alternierende endliche Automaten mit unärem Eingabealphabet konnte die Äquivalenz zum Nichtleerheitsproblem für Extended Lindenmayersysteme, kurz E0L Systeme, bewiesen werden. Erst kürzlich wurde in [6] die $\mathcal{PSPACE}$-Vollständigkeit des Nichtleerheitsproblems für E0L Syteme gezeigt.

Auf Grund der Zusammenhänge zwischen Nichtleerheitsproblemen für alternierende endliche Automaten und Nichtleerheitsproblemen für Lindenmayersyteme kann gezeigt werden, daß schon sehr eingeschränkte Nichtleerheitsproblem für alternierende endliche Automaten $\mathcal{NP}$ bzw. $\mathcal{PSPACE}$-vollständig sind.

[2] Gefördert durch die Deutsche Forschungsgemeinschaft unter Projekt DFG-La 618/1-1

References

[1] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.

[2] Z. Galil. Hierarchies of complete problems. *Acta Informatica*, 6:77–88, 1976.

[3] N. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11:68–85, 1975.

[4] D. Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Foundations of Computer Science*, pages 254–266, 1977.

[5] K.-J. Lange and P. Rossmanith. The emptiness problem for intersections of regular languages. In *Proceedings of the 17th Conference on Mathematical Foundations of Computer Science*, number 629 in LNCS, pages 346–354. Springer, 1992.

[6] A. Monti and A. Roncato. On the computational complexity of some decidable problems for E0L systems and BSTA. Technical Report TR-12/93, University of Pisa, June 1993.

Reguläre Grundnormalformsprachen und Lineariesierung von Termersetzungssystemen

Maria Huber
CRIN/INRIA Lorraine
Nancy

Abstract

In Termersetzungssystemen lassen sich Regeln, die nicht linkslinear sind, häufig durch linkslineare ersetzen ohne die Sprache der irreduzierbaren Grundterme zu verändern. Um zu entscheiden, ob ein Termersetzungssystem in diesem Sinn "linearisierbar" ist, und um diese Linearisierung gegebenenfalls durchzuführen, werden wir geeignete Testmengen – eine endliche Repräsentation der Grundterme in Normalform – benutzen. Wir zeigen, daß diese Linearisierung genau dann möglich ist, wenn die Grundnormalformsprache eine reguläre Baumsprache bildet.

Endliche Testmengen werden häufig verwendet, um die Sprache der nicht reduzierbaren Grundterme zu charakterisieren. So sind sie ein geeignetes Hilfsmittel um zu entscheiden, ob alle Grundinstanzen eines Terms mit einem gegebenen Termersetzungssystem reduzierbar sind. Diese Grundreduzierbarkeit von Termen wiederum ist zum Beispiel wichtig, um zu entscheiden, ob ein Symbol vollständig definiert ist oder ob eine Gleichung für alle Grundterme gilt. Plaisted [13] und Kapur, Narendran, Zhang [8], [9] haben gezeigt, daß es ausreicht, eine endliche Menge von Grundinstanzen eines Terms zu betrachten, um zu entscheiden, ob dieser Term grundreduzierbar ist. Kleinere Testmengen, die auch Terme mit Variablen enthalten, wurden später u.a. in [7] und [9] für linkslineare Termersetzungssysteme verwendet.

Testmengen, die für einen Grundreduzierbarkeitstest auch im nichtlinearen Fall verwendet werden können, sind sehr viel komplizierter und wurden in [10] und [6] genauer untersucht. Comon gibt in [3] bedingte Grammatiken an, die die Sprachen der irreduzierbaren Grundterme erzeugen.

Neuere Ansätze basieren auf der Konstruktion von Automaten, die genau die Grundnormalformsprache akzeptieren, und deren Leerheitsproblem entscheidbar ist. Während Bogaert und Tison [1] Nichtlinearitäten nur eingeschränkt erlauben (gleiche Variablen in linken Regelseiten sind nur direkt unterhalb der Wurzel erlaubt), sind die Automaten von Caron, Coquide und Dauchet [2] für beliebige Termersetzungssysteme verwendbar.

Wir zeigen, wie die Testmengen, die in [6] ursprünglich zur Entscheidung der Grundreduzierbarkeit konzipiert waren, verwendet werden können, um in linken Regelseiten nichtlineare Variablen durch eine endliche Menge von Grundtermen zu ersetzen, falls dieses möglich ist ohne die Grundnormalformsprache zu verändern. Unser Algorithmus wird durch eine einzige Ableitungsregel beschrieben, die nichtdeterministisch eine nichtlineare Regel auswählt und so Regel für Regel linearisiert. Ist die Regel nicht mehr anwendbar, dann sind entweder alle Regeln linearisiert – es gibt keine nichtlineare Regel mehr –, oder es gibt keine endliche lineare Regelmenge mit der gleichen Grundnormalformsprache. Dazu werden alle möglichen Instanzen einer nichtlinearen linken Regelseite gebildet, die durch Substitution der Variablen durch Terme der Testmenge entstehen. Von diesen Instanzen werden diejenigen aufbewahrt, die nur an der Wurzel und zwar nur mit der Regel, die gerade untersucht wird, reduzierbar sind. Sind alle diese Instanzen durch Substitution nichtlinearer Variablen durch Grundterme entstanden, wird in der Regelmenge die ursprüngliche Regel durch die entsprechenden Instanzen ersetzt; die Regel ist linearisiert, ihre nichtlinearen Variablen waren also in gewisser Weise überflüssig. Andernfalls ist diese Regel und damit die gesamte Regelmenge nicht linearisierbar. Auf diese Weise ersetzt die Ableitungsregel alle "faulen" Variablen (die Variablen, die überflüßig sind, in dem Sinn, daß endlich viele Grundterme die gleiche Arbeit verrichten) durch eine endliche Menge von Grundtermen. Der Algorithmus terminiert nach maximal k vielen Schritten, wobei k die Anzahl der nicht linkslinearen Regeln des Termersetzungssystems ist.

Testmengen lassen sich hier verwenden, da sie die irreduzierbaren Grundterme zu diesem Zweck ausreichend genau representieren. Jeder irreduzierbare Grundterm ist Instanz eines Terms in der Testmenge. Und jeder Term mit Variablen in der Testmenge repräsentiert eine unendliche Menge irreduzierbarer Grundterme. Die Konstruktion von Testmengen basiert auf einer Pumping-Eigenschaft irreduzierbarer Grundterme. Anschaulich sind an den Stellen, wo in den Grundtermen gepumpt werden kann, in den Testmengentermen Variablen zu finden.

Mit Hilfe dieser Pumping-Eigenschaft und dem üblichen Pumping-Lemma für reguläre Baumsprachen läßt sich auch zeigen, daß ein Termersetzungssystem genau dann linearisiert werden kann, wenn seine Grundnormalformsprache eine reguläre Baumsprache bildet. Das wurde bereits von Kucherov in [11] gezeigt, der allerdings zunächst die Linearisierbarkeit nicht entscheiden konnte. Die eine Richtung ist einfach, da zu einer linkslinearen Regelmenge immer eine reguläre Grammatik konstruiert werden kann, die die Sprache der reduzierbaren Grundterme erzeugt. Damit ist auch die Komplementsprache regulär.

Ähnliche Verfahren zur Entscheidung von Linearisierbarkeit und zur Konstruktion einer Linearisierung wurden unabhängig von Hofbauer und Huber [4], [5], von Kucherov und Tajine [12] und von Vágvölgyi und Gilleron [14] angegeben. Alle Ansätze basieren auf der Verwendung von Pumping-Eigenschaften für die Grundnormalformsprache. Nur in [4], [5] werden Testmengen verwendet, die das Wissen über Pumping-Stellen bereits beinhalten.

Neben der Möglichkeit zu entscheiden, ob ein Term grundreduzierbar ist, ob die Grundreduzierbarkeit an endlich oder unendlich vielen Grundinstanzen scheitert etc., hat die Untersuchung der Grundnormalformsprachen Auswirkungen auf das Design von Termersetzungssystemen. Handelt es sich zum Beispiel um ein konstruktorbasiertes System – ein System, in dem zwischen zu definierenden Operationen und Konstruktoroperationen unterschieden wird –, in dem sich Konstruktorterme nur wieder zu Konstruktortermen ableiten lassen, das außerdem termierend und grundkonfluent ist, dann reicht es aus, nichtlineare Variablen in solchen linken Regelseiten zuzulassen, die nur aus Konstruktoren aufgebaut sind. Zur vollständigen Definition von Operationen reichen also linksli-

Neuere Ansätze basieren auf der Konstruktion von Automaten, die genau die Grundnormalformsprache akzeptieren, und deren Leerheitsproblem entscheidbar ist. Während Bogaert und Tison [1] Nichtlinearitäten nur eingeschränkt erlauben (gleiche Variablen in linken Regelseiten sind nur direkt unterhalb der Wurzel erlaubt), sind die Automaten von Caron, Coquide und Dauchet [2] für beliebige Termersetzungssysteme verwendbar.

Wir zeigen, wie die Testmengen, die in [6] ursprünglich zur Entscheidung der Grundreduzierbarkeit konzipiert waren, verwendet werden können, um in linken Regelseiten nichtlineare Variablen durch eine endliche Menge von Grundtermen zu ersetzen, falls dieses möglich ist ohne die Grundnormalformsprache zu verändern. Unser Algorithmus wird durch eine einzige Ableitungsregel beschrieben, die nichtdeterministisch eine nichtlineare Regel auswählt und so Regel für Regel linearisiert. Ist die Regel nicht mehr anwendbar, dann sind entweder alle Regeln linearisiert – es gibt keine nichtlineare Regel mehr –, oder es gibt keine endliche lineare Regelmenge mit der gleichen Grundnormalformsprache. Dazu werden alle möglichen Instanzen einer nichtlinearen linken Regelseite gebildet, die durch Substitution der Variablen durch Terme der Testmenge entstehen. Von diesen Instanzen werden diejenigen aufbewahrt, die nur an der Wurzel und zwar nur mit der Regel, die gerade untersucht wird, reduzierbar sind. Sind alle diese Instanzen durch Substitution nichtlinearer Variablen durch Grundterme entstanden, wird in der Regelmenge die ursprüngliche Regel durch die entsprechenden Instanzen ersetzt; die Regel ist linearisiert, ihre nichtlinearen Variablen waren also in gewisser Weise überflüssig. Andernfalls ist diese Regel und damit die gesamte Regelmenge nicht linearisierbar. Auf diese Weise ersetzt die Ableitungsregel alle "faulen" Variablen (die Variablen, die überflüßig sind, in dem Sinn, daß endlich viele Grundterme die gleiche Arbeit verrichten) durch eine endliche Menge von Grundtermen. Der Algorithmus terminiert nach maximal k vielen Schritten, wobei k die Anzahl der nicht linkslinearen Regeln des Termersetzungssystems ist.

Testmengen lassen sich hier verwenden, da sie die irreduzierbaren Grundterme zu diesem Zweck ausreichend genau representieren. Jeder irreduzierbare Grundterm ist Instanz eines Terms in der Testmenge. Und jeder Term mit Variablen in der Testmenge repräsentiert eine unendliche Menge irreduzierbarer Grundterme. Die Konstruktion von Testmengen basiert auf einer Pumping-Eigenschaft irreduzierbarer Grundterme. Anschaulich sind an den Stellen, wo in den Grundtermen gepumpt werden kann, in den Testmengentermen Variablen zu finden.

Mit Hilfe dieser Pumping-Eigenschaft und dem üblichen Pumping-Lemma für reguläre Baumsprachen läßt sich auch zeigen, daß ein Termersetzungssystem genau dann linearisiert werden kann, wenn seine Grundnormalformsprache eine reguläre Baumsprache bildet. Das wurde bereits von Kucherov in [11] gezeigt, der allerdings zunächst die Linearisierbarkeit nicht entscheiden konnte. Die eine Richtung ist einfach, da zu einer linkslinearen Regelmenge immer eine reguläre Grammatik konstruiert werden kann, die die Sprache der reduzierbaren Grundterme erzeugt. Damit ist auch die Komplementsprache regulär.

Ähnliche Verfahren zur Entscheidung von Linearisierbarkeit und zur Konstruktion einer Linearisierung wurden unabhängig von Hofbauer und Huber [4], [5], von Kucherov und Tajine [12] und von Vágvölgyi und Gilleron [14] angegeben. Alle Ansätze basieren auf der Verwendung von Pumping-Eigenschaften für die Grundnormalformsprache. Nur in [4], [5] werden Testmengen verwendet, die das Wissen über Pumping-Stellen bereits beinhalten.

Neben der Möglichkeit zu entscheiden, ob ein Term grundreduzierbar ist, ob die Grundreduzierbarkeit an endlich oder unendlich vielen Grundinstanzen scheitert etc., hat die Untersuchung der Grundnormalformsprachen Auswirkungen auf das Design von Termersetzungssystemen. Handelt es sich zum Beispiel um ein konstruktorbasiertes System – ein System, in dem zwischen zu definierenden Operationen und Konstruktoroperationen unterschieden wird –, in dem sich Konstruktorterme nur wieder zu Konstruktortermen ableiten lassen, das außerdem termierend und grundkonfluent ist, dann reicht es aus, nichtlineare Variablen in solchen linken Regelseiten zuzulassen, die nur aus Konstruktoren aufgebaut sind. Zur vollständigen Definition von Operationen reichen also linksli-

neare Regeln aus.

References

[1] B. Bogaert und S. Tison. Equality and disequality constraints on direct subterms in tree automata. In *Proc. 9th STACS*, LNCS 577, S. 161–172, 1992.

[2] A.C. Caron, J.L. Coquide und M. Dauchet. Encompassment properties and automata with constraints. In *Proc. 5th RTA*, LNCS 690, S. 328–342, 1993.

[3] H. Comon. Unification et disuuification. Théories et applications. Thèse de Doctorat d'Université, Institut Polytechnique de Grenoble (France), 1988.

[4] D. Hofbauer und M. Huber. Computing linearizations using test sets. In *Proc. 3rd CTRS*, LNCS 656, S. 287–301, 1992.

[5] D. Hofbauer und M. Huber. Linearizing term rewriting systems using test sets. Erscheint im Journal of Symbolic Computation.

[6] M. Huber. Testmengen für Grundreduzierbarkeit: Konstruktionen, Komplikationen, Korollare. Diplomarbeit, Technische Universität Berlin, 1991.

[7] J.P. Jouannaud und E. Kounalis. Proof by induction in equational theories without constructors. In *Proc. 1st LICS*, S. 358–366, 1986.

[8] D. Kapur, P. Narendran und H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.

[9] D. Kapur, P. Narendran und H. Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation*, S. 83–111, 1991.

[10] E. Kounalis. Testing for the ground (co-)reducibility property in term-rewriting systems. *Theoretical Computer Science*, 106:87–117, 1992.

[11] G. Kucherov. On relationship between term rewriting systems and regular tree languages. In *Proc. 4th RTA*, LNCS 488, S. 299–311, 1991.

[12] G. Kucherov und M. Tajine. Decidability of regularity and related properties of ground normal form languages. In *Proc. 3rt CTRS*, LNCS 656, S. 272–286, 1992.

[13] D. A. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65(2/3):182–215, 1985.

[14] S. Vagvoelgyi und R. Gilleron. For a rewrite system it is decidable whether the set of irreducible ground terms is recognizable. In EATCS 48, S. 197–209, 1992.

Neue Anwendungen von Ergebnissen über Petrinetze auf Matrix-Sprachen

Matthias Jantzen
FB Informatik
Universität Hamburg

Unter Verwendung eines sehr allgemeinen Ergebnisses von Dirk Hauschildt (Doktorarbeit, Hamburg 1991) werden u. a. folgende Ergebnisse erzielt:

1. Matrixsprachen über einem einelementigen Alphabet, die ohne Vorkommenstest erzeugt werden sind, stets regulär.
2. Es ist entscheidbar, ob Matrixgrammatiken ohne Vorkommenstest
   a) eine endliche Sprache generieren
   b) eine Sprache mit semilinearem Parikh-Bild generieren.

Weitere Ergebnisse folgen aus der Entscheidbarkeit des Erreichbarkeitsproblems und werden kurz diskutiert.

Aktivierung zellularer Automaten

Henner Kröger
Arbeitsgruppe Informatik
Justus–Liebig–Universität Gießen

Extended Abstract: Die auf J. v. Neumann um 1950 zurückgehenden zellularen Räume, –Automaten und –Algorithmen lassen sich für (1–dimensionale) Spracherkennung und (mehr–dimensionale) Bilderkennung sowie für Mustertransformation und Simulation von Wachstumsprozessen anwenden. Zellulare Räume arbeiten mit unendlich vielen Zellen, wobei der d–dimensionale Vektorraum über den ganzen Zahlen als Menge von Identifikatoren für die Zellen dient und zugleich eine geographische Nachbarschaft festlegt; die (davon verschiedene) Nachbarschaft für den Informationsfluß wird jeweils explizit im jeweiligen zellularen Raum festgelegt. Zellulare Automaten arbeiten nur auf einem endlichen Teilgebiet des Raumes (sogenannte Retina). Zellulare Algorithmen bilden eine „Familie“ zellularer Automaten mit verwandten Retina-Gebilden und gemeinsamer Überführungsfunktion.

Derartige Strukturen gewinnen wegen des aktuellen Interesses an massiv–parallelen Rechnermodellen und wegen des heutigen technischen Entwicklungsstandes an Bedeutung; zugleich nimmt die Größenordnung solcher Algorithmen zu und erfordert entsprechende Disziplin beim Algorithmenentwurf. In Anlehnung an den Fall sequentieller Algorithmen sind unsere Untersuchungen vor dem Hintergrund zu sehen, beim Entwurf komplizierter zellularer Algorithmen modulare Kompositionstechniken zu nutzen: Entwurf und Korrektheit des gesamten Algorithmus sollen auf die korrekte Komposition korrekter Teilalgorithmen abgestützt werden.

Sequentielle Algorithmen und Rechnermodelle wie von–Neumann–Rechner und Turing–Maschinen haben recht einfache Konzepte des Startens und des Stoppens einer Rechnung und erlauben dementsprechend in einfacher Weise z.B. das Hintereinanderschalten (ALGOL–Semikolon) von Teil–Algorithmen.

Zellulare Konzepte dagegen müssen parallele Arbeitsweisen integrieren, die mit Schlagworten wie *lokal*, *myopisch*, *verteilt* zu charakterisieren sind. Zellulare Räume etc. setzen einen globalen Taktgeber voraus, durch den die Zellen synchron arbeiten. Es gibt das Konzept der Startkonfiguration (zum Zeitpunkt $t = 0$): wie die Anfangszustände in die Zellen technisch eingefüllt werden, bleibt dabei in gewissem Sinne offen oder außerhalb der Theorie. Sieht man mal vom Spezialfall von Erkennungs–Algorithmen mit ausgezeichneten ja/nein–Antwortzellen ab, so dienen als Endkonfigurationen solche Konfigurationen, die durch die lokale Überführungsfunktion in sich selbst überführt werden, also für einen globalen Betrachter, der sich außerhalb über dem zellularen Raum befindet, ein stabiles Bild liefern: stabile Konfigurationen. Die Bezeichnung Stop–Konfiguration ist nicht ganz glücklich, da die Konfigurationsübergänge im zellularen Raum nicht im eigentlichen Sinne terminieren sondern gegebenenfalls stabil werden. Für die nur lokal arbeitenden einzelnen Zellen zellularer Strukturen ist es daher äußerst knifflig, eine stabile Endkonfiguration (der ganzen Retina oder des ganzen Raumes) zu erkennen und dann als Startkonfiguration eines anzuschließenden nächsten Algorithmus zu benutzen.

Um von einem Algorithmus P in den nächsten Algorithmus Q umzusteigen, wird man in der Regel spezielle Eigenschaften der Algorithmen ausnutzen müssen, etwa daß der Algorithmus P einen FSSP–Mechanismus auslösen kann und so durch (zeitaufwendige) Synchronisation aller Retinazellen den eigentlich nächsten Algorithmus Q startet.

Die von Bleck und Kröger (1988, 1992) eingeführten zeitlich verzerrten zellularen Algorithmen nutzen aus, daß stabile Endkonfigurationen des Algorithmus P in der Regel nicht schlagartig in der ganzen Retina erreicht werden, sondern daß sich gewisse Teilgebiete bereits vorzeitig stabilisieren; damit ergibt sich die Möglichkeit, den anzuschließenden Algorithmus Q (beziehungsweise eine entsprechend verzerrte Version Q' davon) hier vorzeitig zu starten: statt der gleichzeitigen, globalen Aktivierung der ganzen Retina erfordert dies eine schrittweise Aktivierung, ausgehend von gewissen vorgegebenen Aktivitätszentren der Retina. Da die Zellen und ihre Nachbarn zu unterschiedlichen Taktzeiten aktiviert werden, müssen die Zellen verschiedene Taktzustände speichern und mehrfach für Rechnungen bereitstellen: der verzerrte Algorithmus wird (z.B. im 1–dimensionalen Fall mit klassischer von–Neumann–Nachbarschaft um einen Faktor 2) langsamer, so daß sich gegenüber einer eingeschobenen FSSP–Synchronisation nur bei speziellen Zeit–Retina–Verhältnissen ein zeitlicher Vorteil ergibt, etwa wenn $p + (2n - 2) + q$ größer als $p + 2q$ ist, wobei p, q die Laufzeiten der Algorithmen P, Q sind und $(2n - 2)$ die FSSP–Synchronisation des 1–dimensionalen Retina–Intervalles der Länge n ist. Bleck und Kröger (1988, 1992) geben an, wie man im 1–dimensionalen Fall zu vorgegebenem zellularen Algorithmus mit klassischer von–Neumann-Nachbarschaft und vorgegebenen Aktivitätszentren einen entsprechenden zeitlich verzerrten zellularen Algorithmus gewinnt. Becker (1990) hat (weiterhin 1–dimensionale) verallgemeinerte von–Neumann–Raster $H(k, l) = \{-k, 0, +l\}$ eingeführt und hierfür die Verzerrungstechnik übertragen; außerdem wird dort eine schnellere Variante entwickelt, die sich auf einen Modulo–3–Zähler stützt, wie er bei der Synchronisation asynchroner zellularer Räume benutzt wird.

Inzwischen können die Techniken zur Konstruktion zeitlich verzerrter zellularer Algorithmen auf zellulare Algorithmen beliebiger Dimension mit beliebiger Nachbarschaft und beliebiger (lokaler) Überführungsfunktion übertragen werden, sofern die Retina–Gebilde bezüglich der Nachbarschaften gewisse „anständige“ Eigenschaften erfüllen. Bei gewissen „exotischen“ Retina-Familien muß man vom Konzept des zellularen Algorithmus abrücken und sich wieder auf zellulare Automaten mit einzelner Retina zurückziehen.

Darüber hinaus lassen sich Algorithmen, die das Wachstum zum Beispiel von Schneeflocken steuern, zur Aktivierung und Steuerung anderer Algorithmen im obigen Sinne einsetzen.

References

[1] Ullrich Becker:„Methoden zur Konstruktion und Komposition zeitlich verzerrter zellularer Algorithmen“, Diplomarbeit, AG Informatik, FB Mathematik, Universität Gießen, Mai 1990.

[2] Barbara Bleck und Henner Kröger: „Time-Distorted Cellular Algorithms“, in: ‘Beiträge zur Theorie der Polyautomaten — vierte Folge’, Bericht anläßlich des 4. Zellulartreffens in Braunschweig am 18./19. Februar 1988, herausgegeben von R. Vollmar und U. Golze, TU Braunschweig 1988, 3–10.

[3] Barbara Bleck und Henner Kröger: „Cellular Algorithms“, in: D.J. Evans (ed.): ‘Advances in Parallel Computing’, Vol. 2, JAI Press Ltd., London 1992, 115–143.

Über die Erzeugung von (un-)endlichen Sprachen bei stochastischen k-limitierten 0L-Systemen

Katja Landskron
Institut für Theoretische Informatik
Technische Universität Braunschweig

Abstract

Es werden stochastische k-limitierte unäre und propagierende Lindenmayer-Systeme eingeführt. In Abhängigkeit eines vorgegebenen Schwellwertes $0 \leq \lambda < 1$ kann die erzeugte Sprache eines gegebenen Systems endlich oder unendlich sein. Das Hauptresultat ist ein Entscheidbarkeitskriterium, mit dem für *fast* alle Schwellwerte λ entschieden werden kann, ob die erzeugte Sprache endlich ist oder nicht.

SklPU0L-Systeme

Stochastische 0L-Systeme sind u.a. von Eichhorst und Ruskey eingeführt worden (s. [ER81]). Bei der von ihnen zugrundegelegten Definition werden die Produktionen der Tafel eines gegebenen 0L-Systems K mit Wahrscheinlichkeiten versehen. Da weiterhin das Axiom eine gewisse Auftretenswahrscheinlichkeit besitzt, werden alle Wörter aus $L(K)$ mit gewissen Wahrscheinlichkeiten erzeugt. Sie konnten zeigen, daß stochastische 0L-Systeme, die propagierend und unär sind, nur endliche Sprachen erzeugen können, wenn mindestens zwei Produktionen vorhanden sind, und nur Wörter zur Sprache gehören, die mit Wahrscheinlichkeiten erzeugt werden, die echt größer sind als ein vorgegebener Schwellwert $0 < \lambda < 1$. Im folgenden soll gezeigt werden, daß dieses Resultat nicht mehr gültig ist, wenn eine beschränkte Ersetzung vorausgesetzt wird. Dabei handelt es sich um die von D. Wätjen in [Wä88a] Landskroneingeführten k-limitierten Lindenmayersysteme (kl0L-Systeme) $K = (\Sigma, h, \omega, k)$ mit $k \in I\!N$, bei denen in jedem Ableitungsschritt für alle $a \in \Sigma$ in dem zu ersetzenden Wort w genau $\min\{\#_a w, k\}$ Zeichen ersetzt werden. Dabei wird auch hier weiterhin vorausgesetzt, daß die zu betrachtenden Systeme propagierend und unär sind (kurz klPU0L-Systeme). Die erzeugten

Wörter lassen sich eindeutig über ihre Längen charakterisieren. Es reicht also aus, im folgenden nur die assoziierten klPUL-Systeme $K' = (h', i, k)$ zu betrachten, wobei $i = |\omega|$ gilt und jede Produktion $a \to a^n \in h$ des zugehörigen klPU0L-Systems in h' durch n ersetzt wird. Für solche Systeme läßt sich mit Hilfe der sogenannten Wachstumsmenge angeben, wie groß das Wachstum in einem Ableitungsschritt ist, wenn k Zeichen ersetzt werden. Es sei also $K = (h, i, k)$ ein klPUL-System. Dann ist $\mathcal{G}(K) = \{\, g \mid g = \sum_{p \in h} \lambda_p \cdot (p-1),\ \lambda_p \in \mathbb{N}_0,\ \sum_{p \in h} \lambda_p = k \,\}$ die *Wachstumsmenge*, deren Elemente für propagierende Systeme ≥ 0 sind. Mit Hilfe der Wachstumselemente $g \in \mathcal{G}(K)$ läßt sich die von K erzeugte Sprache $L(K)$ angeben (s. [Wä88b]).

Definition 1 *Ein stochastisches klPUL-System (SklPUL-System) $S = (h, \rho, i, k)$ ist gegeben durch die endliche Menge der Produktionen $h = \{h_1, \ldots, h_s\}$ ($h_j \in \mathbb{N}$), einer Abbildung $\rho : h \to [0,1]$ mit $\sum_{j=1}^{s} \rho(h_j) = 1$ (Wahrscheinlichkeiten der Produktionen), und durch das Axiom $i \in \mathbb{N}$ mit der Auftretenswahrscheinlichkeit 1 sowie der k-Limitierung $k \in \mathbb{N}$.*

Die Wachstumsmenge von S (kurz: $\mathcal{G}(S)$) stimmt mit der Wachstumsmenge des zugrundeliegenden klPUL-Systems $K = (h, i, k)$ überein.

Da das Axiom eine Auftretenswahrscheinlichkeit besitzt, werden auch die von S erzeugten Wörter mit bestimmten Wahrscheinlichkeiten generiert. Im folgenden werde die *Ableitungswahrscheinlichkeit, y aus x (in beliebig vielen Schritten) abzuleiten*, mit $P(x, y)$ bezeichnet. Dabei werden nur Ableitungen der Form $x \Rightarrow^* y$ mit $x = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w_n = y$ und $y \neq w_i\ \forall i = 0, \ldots, n-1$ betrachtet. In jedem Ableitungsschritt $w_j \Rightarrow w_{j+1}$ werden alle Wahrscheinlichkeiten der angewendeten Produktionen mit der Wahrscheinlichkeit von $P(x, w_j)$ multipliziert (es gilt $P(x, x) = 1$). Alle möglichen Ableitungen mit beliebig vielen Schritten werden betrachtet und deren Wahrscheinlichkeiten aufsummiert. Die *Ableitungswahrscheinlichkeit eines Wortes y* wird nun mit $P(y)$ bezeichnet, und es gilt $P(y) = P(i, y)$.

Definition 2 *Es sei $S = (h, \rho, i, k)$ ein SklUL-System, und $\lambda \in [0, 1)$ sei ein sogenannter Schwellwert. Dann ist $L_\lambda(S) = \{y \mid P(y) > \lambda\}$ die von S erzeugte Sprache.*

Definition 3 *Es sei $N = \max\{h_j \mid j = 1, \ldots, s$ und $\rho(h_j) > 0\}$. Die Ein-Schritt-Wahrscheinlichkeit $p(n, m)$ ist definiert durch $p(0, 0) = 1$, $p(0, m) = 0$ für $m \neq 0$, und für $n > 0$ durch $p(n, m) =$*

$$\begin{cases} p(k, m-n+k), & \text{falls } n > k \\ \sum_{j=0}^{N} \rho(j) p(n-1, m-j) & \text{sonst.} \end{cases}$$

Satz 4 *Gegeben sei ein SklPUL-System $S = (h, \rho, i, k)$ mit $\mathcal{G}(S) = \{g_1, \ldots, g_r\}$ ($g_i < g_j$ für $i < j$). Dann ist $\sum_{j=1}^{r} p(k, k+g_j) = 1$. Für die Bestimmung der Ableitungswahrscheinlichkeiten gelten weiterhin die folgenden Formeln: $P(n, m) = p(n, m) + \sum_{j=n}^{m-1} p(n, j) P(j, m) = p(n, m) + \sum_{j=1}^{r} p(n, n+g_j) P(n+g_j, m)$. Weiterhin existiert im Fall $g_1 \neq 0$ ein Wort n_0, so daß für alle $n \geq n_0$ die Ableitungswahrscheinlichkeiten von n rekursiv definiert sind durch $P(n) = \sum_{j=1}^{r} p(n-g_j, n) P(n-g_j)$. Im Fall $g_1 = 0$ erhält man durch Umformungen $P(n, m) = \frac{1}{1-p(n,n)} \cdot \left(p(n, m) + \sum_{i=2}^{r} p(n, n+g_j) P(n+g_j, m)\right)$, und für alle $n \geq n_0$ gilt $P(n) = \frac{1}{1-p(n,n)} \sum_{j=2}^{n} p(n-g_j, n) P(n-g_j)$.*

Beispiel 5 *$S = (\{1, 3\}, \rho, 1, k)$ mit $\rho(1) = \frac{1}{4}$, $\rho(3) = \frac{3}{4}$, $k = 2$ und $P(1) = 1$. Es gilt $\mathcal{G}(S) = \{0, 2, 4\}$. Mit Hilfe der obigen Formeln läßt sich $P(n)$ für alle n berechnen, z.B. gilt $P(1) = P(3) = 1$, $P(5) = \frac{2}{5}$, $P(7) = \frac{19}{25}$, usw. Wählt man als Schwellwert $\lambda_1 = 0{,}627$, dann erzeugt S die endliche Sprache $L_{\lambda_1}(S) = \{1, 3, 7, 11, 15, 19, 23\}$. Im Fall $\lambda_2 = 0{,}62$ gilt $L_{\lambda_2}(S) = \{1, 3, 7, 11, 15\} \cup \{2n+1 \mid n \geq 9\}$. Offensichtlich hängt also die Erzeugungsmächtigkeit von der Wahl des Schwellwertes ab.*

Rekursive Folgen

Satz 6 *Es sei $m \in \mathbb{N}$, $a_1, a_2, \ldots, a_m \in \mathbb{R}$. Für $n > m$ sei die rekursive Folge $a_n = q_1 a_{n-m} + q_2 a_{n-m+1} + \ldots + q_{m-1} a_{n-2} + q_m a_{n-1}$ gegeben, wobei $0 \leq q_i \leq 1$ ($i = 1, \ldots, m$), $\sum_{i=1}^{m} q_i = 1$ gelte. Die Indizes der positiven Koeffizienten seien $i_1, \ldots, i_r$ und werden ihrer Größe nach geordnet: $i_1 < i_2 < \ldots < i_r$. Gilt $ggt(i_2 - i_1, \ldots, i_r - i_1, m+1-i_1) = 1$, dann folgt:*

$$\lim_{n \to \infty} a_n = \frac{q_1 \cdot a_1 + (q_1 + q_2) \cdot a_2 + \ldots + (q_1 + q_2 + \ldots + q_m) \cdot a_m}{m \cdot q_1 + (m-1) \cdot q_2 + \ldots + 2 \cdot q_{m-1} + q_m} < \infty.$$

Beweisidee: Durch Anwendung der Theorie der Differenzengleichungen und einigen Sätzen über stochastische Matrizen kann die Konvergenz bewiesen werden. Mit Hilfe eines induktiven Beweises wird eine äquivalente Darstellung der Folge nachgewiesen. Durch Anwendung von Grenzwertsätzen für Folgen ist hierdurch die explizite Berechnung des Grenzwertes möglich.

Satz 7 *Gilt in Satz 6* $ggt(i_2 - i_1, \ldots, i_r - i_1, m + 1 - i_1) = t > 1$, *dann folgt, daß die Folge t Häufungspunkte besitzt und* α *aus Satz 6 der arithmetische Mittelwert dieser Häufungspunkte ist.*

Beweisidee: Zerlegung der gegebenen Folge in t konvergente Teilfolgen, die disjunkt sind und nach Satz 6 konvergieren.

Beispiel 8 *(Fortsetzung von Beispiel 5)* $a_n := P(n)$. *Für* $n > 5$ *gelte die rekursive Folge* $a_n = \frac{2}{5}a_{n-2} + \frac{3}{5}a_{n-4}$, *wobei* $a_1 = 1$, $a_2 = 0$, $a_3 = 1$, $a_4 = 0$, $a_5 = \frac{2}{5}$ *gelte. Die Koeffizienten sind folgendermaßen bestimmt worden:* $q_4 = \frac{p(2,4)}{1-p(2,2)} = \frac{16}{15} \cdot \frac{3}{8} = \frac{2}{5}$ *und* $q_2 = \frac{p(2,6)}{1-p(2,2)} = \frac{16}{15} \cdot \frac{9}{16} = \frac{3}{5}$. *Folglich besitzt die Folge* $ggt(4-2, 5+1-2) = 2$ *Häufungspunkte: Für die geraden Wörter ist die Ableitungswahrscheinlichkeit* 0, *also* $(a_{2m})_{m \in \mathbb{N}} \longrightarrow 0 \quad (m \to \infty)$, *und für ungerade Wörter gilt für* $m \geq 3$ $a_{2m+1} = \frac{p(2,4)}{1-p(2,2)}a_{2m-1} + \frac{p(2,6)}{1-p(2,2)}a_{2m-3} = \frac{2}{5}a_{2m-1} + \frac{3}{5}a_{2m-3}$ *mit den Anfangswerten* $a_3 = 1$, $a_5 = \frac{2}{5}$. *Es folgt mit Satz 6:* $(a_{2m+1})_{m \in \mathbb{N}} \longrightarrow 0,625 \quad (m \to \infty)$.

Ein Entscheidbarkeitskriterium

Satz 9 *Für fast alle Schwellwerte* $0 \leq \lambda < 1$ *ist es entscheidbar, ob die von einem SklPUL-System* $S = (h, \rho, i, k)$ *erzeugte Sprache* $L_\lambda(S)$ *endlich ist oder nicht.*

Beweis: Es sei $\mathcal{G}(S) = \{g_1, \ldots, g_r\}$. Im Fall $\lambda = 0$ oder $r = 1$ (also $|h| = 1$), ist es trivialerweise entscheidbar, ob $L_\lambda(S)$ endlich ist, nämlich genau dann, wenn 1 die einzige Produktion ist. Es sei also nun $\lambda > 0$, und in h existieren mindestens 2 Produktionen mit positiver Wahrscheinlichkeit (also $r \geq 2$). Mit Hilfe der Formeln aus Satz 4 wird eine rekursive

Folge aufgestellt. Für „genügend" viele Wörter, deren Wahrscheinlichkeiten den Anfangswerten entsprechen, wird die Ableitungswahrscheinlichkeit sukzessive mittels der Formel $P(i,n) = p(i,n) + \sum_{j=i}^{n-1} p(i,j)P(j,n)$ bestimmt. Die Koeffizienten der rekursiven Folge sind durch die Ein-Schritt-Wahrscheinlichkeiten $p(k, k + g_j)$ für alle Wachstumselemente $g_j \in \mathcal{G}(S)$ gegeben (vgl. Satz 4). Die Ableitungswahrscheinlichkeiten der übrigen Wörter lassen sich nun mit Hilfe der rekursiven Folge bestimmen. Durch Anwendung der Sätze 7 und 6 werden die endlich vielen Häufungspunkte der rekursiven Folge berechnet. Der größte Häufungspunkt sei α_t. In Abhängigkeit von der Wahl des Schwellwertes λ gilt dann:

- $\lambda > \alpha_t$: $L_\lambda(S)$ endlich,
- $\lambda < \alpha_t$: $L_\lambda(S)$ unendlich,
- $\lambda = \alpha_t$: i.a. mit obigem Kriterium keine Aussage möglich.

References

[ER81] *Peter Eichhorst, Frank Ruskey.* On Unary Stochastic Lindenmayer Systems. Information and Control **48** (1981), 1-10.

[Wä88a] *D. Wätjen.* k-limited 0L Systems and Languages. J. Inf. Process. Cybern. EIK **24** (1988) 6, 267-285.

[Wä88b] *D. Wätjen.* On Unary *k*-limited 0L Systems and Languages. J. Inf. Process. Cybern. EIK **24** (1988) 9, 415-429.

Deterministische Zweiweg-Kellerautomaten: Bemerkungen zu Endlosschleifen

H. Petersen
Fachbereich Informatik
Universität Hamburg
(in Zusammenarbeit mit M. Ladermann)

Deterministische Zweiweg-Kellerautomaten (kurz 2-DPDA) wurden in [8] (dort unter der Bezeichnung „off-line") eingeführt und mit Variationen in [3, 1, 2] untersucht. Dabei erwiesen sich die unterschiedlichen Definitionen von Arbeits- und Akzeptierungsmodi als äquivalent—bis auf ein Detail, daß bis heute ungelöst geblieben ist: Falls ein 2-DPDA auf einer Eingabe in eine Endlosschleife gelangen darf und dies wie in [3] als Ablehnung gewertet wird, ist nicht klar, ob solche Schleifen eliminiert werden können, um den Automaten zum Halten zu bringen (in [8] wurde Halten vorausgesetzt, eine unentscheidbare und daher problematische Forderung.)

Zunächst verdeutlichen wir die Leistungsfähigkeit von 2-DPDA am Beispiel der Berechnung primitiver Wurzeln (Definition in [4]) durch einen 2-DPDA mit einem Eingabekopf. Die Hauptidee ist hier die Anwendung von Zeichenketten-Vergleich zwischen zwei Kopien der Eingabe (um erstes und letztes Zeichen verkürzt), die auf dem Keller gehalten werden, und der Eingabe selbst. Die Länge des verbleibenden Kellerinhaltes kann dann zur Bestimmung der primitiven Wurzel benutzt werden.

In [5] wurde der Versuch unternommen, die von Sipser [7] im Bereich der platzbeschränkten deterministischen Turingmaschinen entwickelte Technik der Rückwärtssimulation auf 2-DPDA zu übertragen. Der Beweis läßt allerdings offen, wie eine der wesentlichen Voraussetzungen der Sipser-Technik, nämlich die Endlichkeit des Konfigurationsraumes, bei 2-DPDA gewährleistet werden kann. Wir verdeutlichen dies an einem Beispiel, bei dem die Rückwärtssimulation zu einem Kellerüberlauf führt.

Die Endlosschleifen eines 2-DPDA lassen sich einem der beiden folgenden Typen zuordnen:

- Der Automat wiederholt Konfigurationen
- Die Rechnung besteht aus lauter verschiedenen Konfigurationen, der Keller wächst unbeschränkt

Unsere Füllsel-Konstruktion transformiert Schleifen der ersten Art in solche zweiter Art. Genauer führen wir ein Hilfs-Kellerbodensymbol und ein weieteres neues Kellersymbol (das Füllsel) ein. Jedem Zustand fügen wir eine Schleife hinzu, die das Füllsel löscht, und ergänzen Übergänge, die nichtleere Zeichenketten in den Keller schreiben so, daß vorher ein Füllsel geschrieben wird. Eine Argumentation über minimale Kellerinhalte in einer Endlosschleife zeigt, daß Schleifen der ersten Art nun ausgeschlossen sind.

Es ergeben sich folgende Resultate:

- Zu jedem k-Kopf 2-DPDA gibt es einen äquivalenten $2k$-Kopf 2-DPDA, der immer hält.
  - Ein zweiter Satz Köpfe (mit endlicher Kontrolle) dient als Tiefenbegrenzung für den Keller.
  - Zur Vermeidung von Schleifen wird entweder Rückwärtssimulation oder
  - Füllsel-Konstruktion durchgeführt.
- Jedes Komplement einer k-Kopf 2-DPDA Sprache wird von einem k-Kopf 2-NPDA erkannt.
  - Durch die Füllsel-Konstruktion kann in Schleifen ein Kellerüberlauf erzwungen werden.
  - Der Kellerüberlauf wird „spontan" mit k Köpfen geprüft.
  - Akzeptierende und ablehnende Zustände werden vertauscht.
- Folgerungen: Sind 2-NPDA oder 2-DPDA nicht komplementabgeschlossen, dann sind 2-DPDA weniger mächtig als 2-NPDA mit gleicher Kopfzahl.

- Zu jedem 2-DPDA gibt es einen äquivalenten 2-DPDA mit schwachem Zähler [6], der immer hält.
  - Die Kellertiefe einer haltenden Rechnung ist polynomiell begrenzt, die Anzahl der Schritte daher exponentiell.
  - Die Laufzeitschranke kann durch Zählen im Keller auf dem schwachen Zähler konstruiert werden.
  - In jedem Schritt einer Vorwärtssimulation wird der schwache Zähler erniedrigt.

References

[1] A. V. Aho, J. E Hopcroft, J. D. Ullman: *Time and tape complexity of pushdown automaton languages,* Inf. Contr. 13 (1968), pp. 186–206.

[2] S. A. Cook: *Characterization of pushdown machines in terms of time-bounded computers,* JACM 18 (1971), pp. 4–18.

[3] J. N. Gray, M. A. Harrison, O. H. Ibarra: *Two-way pushdown automata,* Inf. Contr. 11 (1967), pp. 30–70.

[4] M. A. Harrison: *Introduction to Formal Language Theory,* Addison-Wesley, Reading Massachusetts, (1978).

[5] O. H. Ibarra, M. A. Palis, J. H. Chang: *On efficient recognition of transductions and relations,* TCS 39 (1985), pp. 89–106.

[6] S. Miyano: *Remarks on two-way automata with weak-counters,* IPL 18 (1984), pp. 105–107.

[7] M. Sipser: *Halting space bounded computations,* TCS 10 (1980), pp. 335–338.

[8] R. E. Stearns, J. Hartmanis, P. M. Lewis II: *Hierarchies of memory limited computations,* IEEE Conf. on Switch. Circuit Theory and Logical Design (1965), pp. 179–190.

[9] K. Wagner, G. Wechsung: *Computational Complexity,* Reidel, Dordrecht (1986).

Eine Bemerkung über Linksableitung indisch paralleler Grammatiken

Bernd Reichel
Otto-von-Guericke-Universität
Magdeburg

Indisch parallele Grammatiken sind eine Version der kontextfreien Grammatiken mit gesteuerter Ableitung, d.h. Grammatiken mit kontextfreien Produktionen bei nicht-kontextfreier Anwendung. Diese Grammatiken wurden eingeführt, um bei Verwendung kontextfreier Produktionen nicht-kontextfreie Sprachen zu erzeugen. Indisch parallele Grammatiken und Sprachen wurden zuerst in [5] und [6] untersucht.

Linksableitung ist ein interessanter Abspekt der Theorie Formaler Sprachen in bezug auf die Erzeugungsmächtigkeit von Grammatiken. So ist die erzeugte Sprachklasse der Typ-0-Grammatiken bei Linksableitung nur die Familie der kontextfreien Sprachen [1], wobei sich bei kontextfreien Grammatiken bekanntlich die erzeugte Sprachklasse bei Linksableitung nicht ändert. Die Erzeugungsmächtigkeit einiger Arten von kontextfreien Grammatiken mit gesteuerter Ableitung wurde z.B. in [3], [4] und [2] untersucht.

Eine *indisch parallele Grammatik* G ist ein 4-Tupel $G = (N, T, P, S)$, wobei N das Alphabet der Variablen, T das Alphabet der Terminale ist und $N \cap T = \emptyset$ ist, es sei $N \cup T = V$. $P \subseteq N \times V^*$ ist die endliche Menge der Produktionen und $S \in N$ das Startsymbol.

Ein Wort ω_1 über V erzeugt in G direkt ein Wort ω_2 über V (bezeichnet als $\omega_1 \underset{G}{\Longrightarrow} \omega_2$, oder $\omega_1 \Longrightarrow \omega_2$ falls G aus dem Kontext ersichtlich ist) genau dann, wenn
i) $\omega_1 = \gamma_0 A \gamma_1 A \ldots A \gamma_n$ mit $A \in N$, $\gamma_i \in (V \setminus \{A\})^*$ für $i = 0, 1, \ldots, n$,
ii) $\omega_2 = \gamma_0 \alpha \gamma_1 \alpha \ldots \alpha \gamma_n$ und
iii) $A \longrightarrow \alpha$ in P ist.

Falls in i) $\gamma_0 \in T^*$ gilt, sagen wir, daß ω_1 in Linksableitung ω_2 erzeugt (wir schreiben dafür $\omega_1 \underset{L}{\Longrightarrow} \omega_2$), und falls in i) $\gamma_n \in T^*$ gilt, sagen wir, daß ω_1 in Rechtsableitung ω_2 erzeugt (bezeichnet als $\omega_1 \underset{R}{\Longrightarrow} \omega_2$).

Die Sprache, die von einer indisch parallelen Grammatik G erzeugt wird, ist definiert als $L(G) = \{w \in T^* \mid S \overset{*}{\Longrightarrow} w\}$, wobei $\overset{*}{\Longrightarrow}$ den reflexiven und transitiven Abschluß von $\Longrightarrow$ darstellt. Die Sprachen, die von einer indisch parallelen Grammatik G in Links- und Rechtsableitung erzeugt werden, definieren wir als $L_L(G) = \{w \in T^* \mid S \overset{*}{\underset{L}{\Longrightarrow}} w\}$ bzw. $L_R(G) = \{w \in T^* \mid S \overset{*}{\underset{R}{\Longrightarrow}} w\}$, wobei wieder $\overset{*}{\underset{L}{\Longrightarrow}}$ und $\overset{*}{\underset{R}{\Longrightarrow}}$ den reflexiven und transitiven Abschluß von $\underset{L}{\Longrightarrow}$ bzw. $\underset{R}{\Longrightarrow}$ darstellen. Wir bezeichnen mit $\mathcal{L}(\mathrm{IP})$, $\mathcal{L}_L(\mathrm{IP})$ und $\mathcal{L}_R(\mathrm{IP})$ die Familien der Sprachen, die von indisch parallelen Grammatiken in algemeiner, in Links- bzw. in Rechtsableitung erzeugt werden.

Dann haben wir das Resultat, daß die Familien $\mathcal{L}(\mathrm{IP})$, $\mathcal{L}_L(\mathrm{IP})$ und $\mathcal{L}_R(\mathrm{IP})$ paarweise unvergleichbar sind.

References

[1] G. H. Matthews. A note on asymmetry in phrase structure grammars. *Inform. Control* **7** *(1964); 360–365.*

[2] Gh. Păun. On leftmost derivation restriction in regulated rewriting. *Rev. Roum. Math. Pures Appl.* **30** *(1985); 751–758.*

[3] A. Salomaa. Matrix grammars with a leftmost derivation. *Inform. Control* **20** *(1972); 143–149.*

[4] A. Salomaa. *Formal Languages.* Academic Press, New York and London, 1973.

[5] R. Siromoney and K. Krithivasan. Parallel context-free languages. *Inform. Control* **24** *(1974); 155–162.*

[6] S. Skyum. Parallel context-free languages. *Inform. Control* **26** *(1974); 280–285.*

Über Fragen des Determinismus bei endwachsenden fadenförmigen Systemen

Torsten Roßnick
Otto-von-Guericke-Universität
Magdeburg

Endwachsende fadenförmige Systeme (im weiteren kurz als AGFS bezeichnet - engl.: apical growth filamentous systems) wurden 1983 von NIRMAL und KRIHIVASAN als Spezialfall der L-Systeme eingeführt. Dabei wird die parallele Ersetzung auf das äußerste linke bzw. rechte Symbol beschränkt und weiterhin eingeschränkt, daß keine Verzweigungen auftreten sollen. BĂLĂNESCU, GHEORGHE und PĂUN griffen die Idee auf und erlaubten die parallele Ersetzung der äußersten k Symbole. Als offenes Problem wurde die erzeugende Kraft des Determinismus genannt.
Bei k-AGFS wird die Regelmenge P als Vereinigung zweier Teilmengen P_1 und P_2 aufgefaßt. Dabei sind die Regeln, die auf linke Symbole angewandt werden können, in P_1 enthalten und die für rechte Symbole in P_2. Sind die Mengen P_1 und P_2 -jeweils für sich betrachtet- deterministisch, so spreche ich von deterministischen k-AGFS. Ist jedoch die Menge P deterministisch, so spreche ich von streng deterministischen k-AGFS.
Die deterministischen k-AGFS ergeben für konstante k eine zu den nichtdeterministischen k-AGFS vergleichbare Struktur. Die streng deterministischen k-AGFS verhalten sich ähnlich der (nicht)deterministischen k-AGFS, wobei aber einige Verschiebungen von spracherzeugenden Klassen zu verzeichnen sind.
Für konstantes k und festes $X \in \{F, \lambda\}$, $Y \in \{F, \lambda\}$ sowie $Z \in \{P, \lambda\}$ läßt sich die Teilmengenbeziehung

$$D_sXYZ\,\mathcal{P}(k) \;\subseteq\; DXYZ\,\mathcal{P}(k) \;\subset\; XYZ\,\mathcal{P}(k)$$

nachweisen. Die Echtheit der Teilmengenbeziehung zwischen den streng deterministischen und deterministischen Sprachklassen mit den selben erzeugenden Bedingungen wird vermutet, ist jedoch noch nicht bewiesen.

References

[1] BĂLĂNESCU, Tudor ; GHEORGHE, Marian ; PĂUN, Gheorghe
Three Variants of Apical Growth Filamentous Systems
Intern. J. Comp. Math. **3-4** (1987), 227-238

[2] BĂLĂNESCU, Tudor ; GHEORGHE, Marian ; PĂUN, Gheorghe
Apical Growth Filamentous Systems with regulated rewriting
Bull. Math. Soc. Sci. Math. de Roumanie **34** (82), **2** (1990), 99-113

[3] NIRMAL, Nalinakshi ; KRITHIVASAN, Karmal
Filamentous Systems with Apical Growth
Intern. J. Comp. Math. **12** (1983), 203-215

[4] PĂUN, Gheorghe
Some further remarks on regular controlled AGFS
Bull. Math. Soc. Sci. Math. RSR **32** (80), **4** (1988), 341-344

Two-dimensional Picture Languages

Sebastian Seibert
Institut für Informatik
Christian-Albrechts-Universität Kiel

Dora Giammaresi und Antonio Restivo
Università di Palermo

Wolfgang Thomas
Institut für Informatik
Universität Kiel

Wir betrachten Bilder als zweidimensionale Felder von Buchstaben in Rechteckform und vergleichen logische Definierbarkeit von Bildsprachen mit der Beschreibung durch Parkettierungen mit einer endlichen Menge von Teilbildern, wobei auf jeden Buchstaben ein Zustand gelegt wird. Als eine natürliche Erweiterung des Erkennbarkeitsbegriffs von Worten auf Bilder stellt sich die Parkettierung durch Teilbilder der Größe 2×2 dar. Wir zeigen, daß sich damit genau die Bildsprachen erkennen lassen, die durch existentielle monadische Logik zweiter Stufe definierbar sind. Gleichzeitig wird nachgewiesen, daß Parkettierungen mit größeren Teilbildern, bei denen es erlaubt ist, die Anzahl der Vorkommen der Teilbilder bis zu einer Schranke festzulegen („threshold counting"), zu demselben Erkennbarkeitsbegriff führen.

Least Solutions of Equations over $\mathcal{N}$

Helmut Seidl
Fachbereich Informatik
Universität des Saarlandes

We consider the problem of computing the least solution $X_i, i = 1, \ldots, n$, of a system S of equations $x_i = f_i$, $i = 1, \ldots, n$, over $\mathcal{N}$, i.e., the naturals (extended by ∞), where the right hand sides f_i are expressions built up from constants and variables by operations taken from various sets Ω. Many compile time analyses of programs rely on computations of least solutions of such systems of equations.

We present efficient algorithms in case where Ω consists of

(1) minimum and maximum;

(2) maximum, addition and multiplication;

(3) minimum, addition and multiplication; and

(4) minimum, maximum, addition and multiplication.

The algorithms use multiplications only provided S contains multiplications. Also, we design polynomial time algorithms *without multiplications* which compute the set of all i where $X_i = \infty$. This result is used to decide in polynomial time whether or not the costs of tree automata with cost functions of a very general form are bounded.

Zur Anzahl der aktiven Nichtterminale in kooperierenden Grammatiksystemen

Stefan Skalla
Otto-von-Guericke-Universität
Magdeburg

Kooperierende Grammatiksysteme stellen ein formales Modell für „blackboard architectures" der künstlichen Intelligenz dar, wobei dieses Tafelmodell im wesentlichen aus

- mehreren verteilten Wissensressourcen/Experten,
- der Tafel, die den aktuellen Stand der Lösung enthält,
- einem Ordner, der die Reihenfolge der Experten im Lösungsprozeß festlegt,

besteht. Bei derartigen Systemen von Grammatiken entsprechen die einzelnen Grammatiken den Experten, die Satzformen den partiellen Lösungen auf der Tafel, die Arbeit des Ordners wird durch Mechanismen zur Steuerung des Ableitungsprozesses modelliert.

Ein Nichtterminal A heißt *aktiv in einer Komponente* G_i, wenn für dieses Nichtterminal A in dieser Komponente mindestens eine Regel $A \to w$ mit $w \neq A$ existiert.

Betrachtet werden zunächst die Hierarchien der Sprachfamilien $\mathcal{L}_{(m,[n])}$ (m – maximale Anzahl der Komponenten, n – maximale Anzahl der aktiven Nichtterminale pro Komponente). Für Grammatiksysteme ohne zusätzliche Steuerung im $*$-, $\leq k$-, $= k$-, $\geq k$- und t-Modus sowie für Hybridsysteme, bei denen jede Komponente in ihrem speziellen Modus arbeitet, wird gezeigt, daß $\mathcal{L}_{(m-1,[n])} \subset \mathcal{L}_{(m,[n])}$ und $\mathcal{L}_{(m,[n-1])} \subset \mathcal{L}_{(m,[n])}$.

Weiterhin wurden Systeme mit Steuerung durch Graphen in verschiedenen Ableitungsmodi und Start-und Stopbedingungen verschiedener Typen untersucht. Auch für diese Systeme werden diese hierarchischen Beziehungen nachgewiesen.

Außerdem ist die Frage nach oberen Schranken interessant, wenn jeweils nur einer der beschriebenen Parameter in die Betrachtungen einbezogen wird. Hinsichtlich der Anzahl der Komponenten wurden bereits von Csuhaj-Varju, Dassow, Mitrana, Păun und Vicolov für verschiedene Arten von Grammatiksystemen derartige Schranken bestimmt.

Hinsichtlich der maximalen Anzahl n der aktiven Nichtterminale pro Komponente wird für Systeme ohne eine zusätzliche Steuerung im $*$-, $\leq k$-, $= 1$-Modus bewiesen, daß $n = 1$ eine solche Schranke darstellt. Jedes beliebige System im t-Modus kann durch ein System mit höchstens vier aktiven Nichtterminalen pro Komponente simuliert werden. Betrachtet man Systeme im $*$-, $\leq k$-, $= 1$-und t-Modus mit Steuerung durch Graphen, so erhält man ebenfalls eine Schranke von $n = 1$.

Weiterhin wurden Systeme mit Steuerung durch *memories* untersucht. Die Regeln dieser Systeme sind in der Lage, an alle Komponenten des Systems bestimmte Nachrichten zu senden, die von diesen gespeichert werden, den Speicher der eigenen Komponente auf das Enthaltensein eines bestimmten *checkwords* zu testen und vor jedem Zugriff die Satzform auf das Erfülltsein von Kontext-Startbedingungen zu überprüfen.

Von Csuhaj-Varju wurde nachgewiesen, daß bei Zulässigkeit von ε-Regeln diese Systeme die Menge der rekursiv aufzählbaren Sprachen erzeugen.

Bezüglich der maximalen Anzahl n der aktiven Nichtterminale pro Komponente wird gezeigt, daß $n \leq 2$ ausreichend ist, wobei diese Aussage dahingehend verschärft wird, daß sogar nur zwei Produktionen pro Komponente genügen.

On Syntactic Congruences for ω–languages and the minimization of ω-automata

Ludwig Staiger
Lehrstuhl für Informatik II Technische Universität Cottbus

Oded Maler
LGI-IMAG (Campus) Grenoble

It is well-known that the minimization problem for ω-automata (*Muller*-automata) is not as simple as the one for automata accepting languages. It occurs that an ω-language E has more than one minimal automaton recognizing it (e.g. $\{a,b\}^*a^\omega$, see [Mu63] or [St83]). Moreover, this same example shows that, in contrast to the language case, the automaton derived from the right congruence[3] $\sim_E$ does not recognize at all the ω-language E. So there are several possibilities occurring in the ω-case:

A.1 The minimal-state automaton $\mathcal{A}_E$ isomorphic to the right congruence $\sim_E$ accepts the ω-language E, and is, therefore, the unique (up to isomorphism) minimal ω-automaton accepting E.

A.2 There is a unique (up to isomorphism) minimal ω-automaton recognizing E which does not coincide with $\mathcal{A}_E$.

A.3 There are several minimal ω-automata recognizing E. Then necessarily none of them coincides with $\mathcal{A}_E$.

In the case of languages $W \subseteq \Sigma^*$ the well-known Kleene–Myhill–Nerode Theorem states that W is regular iff its right congruence $\sim_W$, or equivalently its syntactic congruence $\simeq_W$ are of finite index. Moreover these relations are the coarsest (right) congruences such that W is representable as a union of congruence classes, and $\sim_W$ is isomrphic to the minimal deterministic automaton accepting $W \subseteq \Sigma^*$.

As pointed out above, in case of ω-languages things are more complicated. Though Arnold [Ar85] proved that for regular ω-languages E there is

[3] Definitions may be found e.g. in [MS93]

a unique coarsest congruence relation $\cong_E$ which recognizes E, up to now no application of this fact to the minimization problem for ω-automata is given.

Thus, several questions arise in connection with this problem:

B.1 If the congruence $\simeq_E$ derived from $\sim_E$ coincides with Arnold's congruence $\cong_E$, that is, $\simeq_E$ already recognizes the regular ω-language E, does then the minimal-state automaton $\mathcal{A}_E$ accept E, and vice versa, does $\simeq_E = \cong_E$ hold if $\mathcal{A}_E$ accepts E?

B.2 Give a characterization of those (regular) ω-languages E for which the relations $\simeq_E$ and $\cong_E$ coincide, or which are accepted by their minimal–state automaton $\mathcal{A}_E$.

B.3 Do there exist regular ω-languages E which have (including the choice of the initial state) a unique minimal automaton $\mathcal{A} \neq \mathcal{A}_E$ accepting E? (The ω-language $(a^*bab^*)^\omega$ has exactly two two-state automata accepting it which differ only in the choice of the initial state.)

Concerning B.1 and B.2 it is shown in [St83] that all regular ω-languages E in the Borel class $F_\sigma \cap G_\delta$ are accepted by their minimal-state automaton $\mathcal{A}_E$, but there are regular ω-languages $E \notin F_\sigma \cap G_\delta$ which are also accepted by $\mathcal{A}_E$, and in [MS93] it is shown that for the same class of regular ω-languages the relations $\simeq_E$ and $\sim_E$ coincide. (In [MS93] both results are derived for the whole class $F_\sigma \cap G_\delta$, that is including ω-languages which are not necessarily regular.)

Next we shall give examples that the conditions are likewise independent:

Ex.1 $E_1 := \{a, bb\}^*a^\omega$ is accepted by $\mathcal{A}_{E_1}$ but $\simeq_{E_1} \neq \cong_{E_1}$.

Ex.2 For $E_2 := \{a,b\}^*a^\omega \cup ca^\omega$ the relations $\simeq_{E_2}$ and $\cong_{E_2}$ coincide, but $\mathcal{A}_{E_2}$ does not accept E_2.

Ex.3 The ω-language $E_3 := \{a,b\}^*a^\omega$ is neither accepted by $\mathcal{A}_{E_3}$ nor do the relations $\simeq_{E_3}$ and $\cong_{E_3}$ coincide

Moreover in [MS93] an alternative notion of recognition of ω-languages by FORCs (families of right-congruences) is developed. Using this type of recognition we give a full characterization of those regular ω-languages E which are accepted by their minimal-state automaton $\mathcal{A}_E$.

References

[Ar85] A. Arnold, A syntactic congruence for rational ω-languages, *Theoret. Comput. Sci.* **39**, 333–335, 1985.

[MS93] O. Maler and L. Staiger, On syntactic congruences for ω-languages, Aachener Informatik–Berichte **93-13**. (a preliminary version appeared in: Proc. STACS 93, LNCS **665**, Springer, Berlin 1993, pp. 586–594.)

[Mu63] D.E. Muller, Infinite sequences and finite machines, in Proc. 4th Ann. IEEE Symp. *Switch. Th. & Logic. Design*, Chicago 1963, pp. 3–16.

[St83] L. Staiger, Finite-state ω-languages, *J. Comput. System Sci.* **27**, 434–448, 1983.

GI–Fachgruppe "Automaten und Formale Sprachen"

Wahl der Fachgruppenleitung

Die Wahl der Leitung der GI–Fachgruppe 0.1.5 wurde am 7.10.1993 im Rahmen der Fachgruppen–Sitzung auf dem 3. Theorietag "Automaten und Formale Sprachen" in Schloß Dagstuhl durchgeführt. Anwesend waren die folgenden Fachgruppen–Mitglieder:

Henning Bordihn (Magdeburg), Franz–J. Brandenburg (Passau), Mathias Bull (Rostock), Gerhard Buntrock (Würzburg), Olaf Burkart (Aachen), Jürgen Dassow (Magdeburg), Volker Diekert (Stuttgart), Henning Fernau (Karlsruhe), Rudolf Freund (Wien), Markus Holzer (München), Matthias Jantzen (Hamburg), Henner Kröger (Gießen), Katja Landskron (Braunschweig), Klaus–Jörn Lange (München), Helmut Lescow (Kiel), Anca Muscholl (Stuttgart), Friedrich Otto (Kiel), Holger Petersen (Hamburg), Bernd Reichel (Magdeburg), Klaus Reinhardt (Stuttgart), Peter Rossmanith (München), Torsten Rossnick (Magdeburg), Sebastian Seibert (Kiel), Helmut Seidl (Saarbrücken), Stefan Skalla (Magdeburg), Ludwig Staiger (Aachen), Ralf Stiebe (Magdeburg), Wolfgang Thomas (Kiel).

Zum Wahlleiter wurde Herr Sebastian Seibert bestimmt. Von den 28 abgegebenen Stimmen waren 27 gültig. Es wurden gewählt

Jürgen Dassow (Magdeburg), Volker Diekert (Stuttgart), Klaus–Jörn Lange (München), Ludwig Staiger (Aachen), Wolfgang Thomas (Kiel).

Alle Gewählten nahmen die Wahl an. Das Wahlprotokoll wurde verlesen und genehmigt.

Wahl des Fachgruppensprechers

Die Wahl des Fachgruppensprechers fand ebenfalls am 7.10.1993, in Schloß Dagstuhl statt. Anwesend waren alle Mitglieder der neugewählten

Fachgruppenleitung. Herr Thomas eröffnete die Wahlversammlung. Als Kandidat für das Sprechamt wurde J. Dassow, als Kandidat für das Amt des Stellvertreters K.–J. Lange vorgeschlagen. In offener Wahl wurden J. Dassow als Sprecher der Fachgruppe und K.–J. Lange als stellvertretender Sprecher der Fachgruppe einstimmig gewählt. Beide nahmen die Wahl an.

Teilnehmerliste

1. Henning Bordihn
   Technische Universität Magdeburg
   Fakultät für Informatik
   Postfach 4120
   39016 Magdeburg
   e-mail: <petersen@informatik.uni-hamburg.de>
   fax: 040/54715-246

2. Franz-J. Brandenburg
   Fachbereich Informatik der Universität Passau
   Innstr. 27
   4032 Passau
   e-mail: <brandenb@fmi.uni-passau.de>

3. Mathias Bull
   Universiät Rostock
   Fachbereich Informatik
   18051 Rostock
   e-mail: <mb@informatik.uni-rostock.de>

4. Gerhard Buntrock
   Institut für Informatik
   Universität Würzburg
   Am Exerzierplatz 3
   97072 Würzburg
   e-mail: <bunt@informatik.uni-wuerzburg.de>

5. Olaf Burkart
Department of Computer Science II
RWTH-Aachen
Ahornstr. 55
52056 Aachen
e-mail: <burkart@zeus.informatik.rwth-aachen.de>

6. Robert Cremanns
Fachbereich Mathematik, FG Informatik
Gesamthochschule Kassel
Postfach 10 13 80
34013 Kassel
e-mail:
<Robert.Cremanns@theory.informatik.uni-kassel.de>

7. Jürgen Dassow
Technische Universität Magdeburg
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
e-mail: <dassow@irb.cs.uni-magdeburg.de>

8. Volker Diekert
Institut für Informatik
Universität Stuttgart
Breitwiesenstraße 20 -22
70565 Stuttgart
e-mail: <diekert@informatik.uni-stuttgart.de>
fax: + 49 - 711 - 78 01 310

9. Henning Fernau
Universität Karlsruhe
Lehrstuhl für Informatik

76128 Karlsruhe
e-mail: <fernau@ira.uka.de>
fax: +49(721)698675

10. Rudolf Freund
Technische Universität Wien
Resselgasse 3
A–1040 Wien
e-mail: <freund@csdec1.tuwien.ac.ate>

11. Dieter Hofbauer
Technische Universität Berlin
Fachbereich Informatik
Franklinstr. 28/29
10587 Berlin
e-mail: <dieter@cs.tu-berlin.de>

12. Markus Holzer
Institut für Informatik
Technische Universität München
80290 München
e-mail: <holzer@informatik.tu-muenchen.de>

13. Maria Huber
INRIA Lorraine
Campus Scientifique
615, Rue du Jardin Botanique
54602 Villers les Nancy Cedex
FRANCE
e-mail: <Maria.Huber@loria.fr>

14. Matthias Jantzen
FB Informatik der Univ. Hamburg

Vogt-Kölln-Straße 30, Haus C
22527 Hamburg
e-mail: <jantzen@rz.informatik.uni-hamburg.dbp.de>

15. Henner Kröger
Universität Gießen
Fachbereich Mathematik
Arndtstr. 2
35392 Gießen
e-mail: <kroeger@turing.informatik.uni-giessen.de>

16. Katja Landskron
TU Braunschweig
Institut für Theoretische Informatik
Fallersleber-Tor-Wall 22
38023 Braunschweig
e-mail: <landskro@iti.cs.tu-bs.de>

17. Klaus-Jörn Lange
Institut für Informatik
Technische Universität München
80290 München
e-mail: <lange@informatik.tu-muenchen.de>

18. Helmut Lescow
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24118 Kiel
e-mail: <Helmut.Lescow@informatik.uni-kiel.dbp.de>

19. Anca Muscholl
Institut für Informatik

Universität Stuttgart
Breitwiesenstraße 20 -22
70565 Stuttgart
e-mail: <muscholl@informatik.uni-stuttgart.de>

20. Gundula Niemann
Institut für Informatik
Universität Würzburg
Am Exerzierplatz 3
97072 Würzburg
e-mail: <Gundula.Niemann@informatik.uni-wuerzburg.de>

21. Friedrich Otto
Fachbereich Mathematik, FG Informatik
Gesamthochschule Kassel
Postfach 10 13 80
34013 Kassel
e-mail: <otto@theory.informatik.uni-kassel.de>

22. Holger Petersen
FB Informatik der Univ. Hamburg
Vogt-Kölln-Straße 30, Haus C
22527 Hamburg
e-mail: <petersen@informatik.uni-hamburg.de>
fax: 040/54715-246

23. Bernd Reichel
Technische Universität Magdeburg
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
e-mail: <reichel@irb.cs.tu-magdeburg.de>

24. Klaus Reinhardt
Institut für Informatik
Universität Stuttgart
Breitwiesenstraße 20 -22
70565 Stuttgart
e-mail: <reinhardt@informatik.uni-stuttgart.de>

25. Peter Rossmanith
Institut für Informatik
Technische Universität München
80290 München
e-mail: <rossmanith@informatik.tu-muenchen.de>

26. Torsten Rossnick
Technische Universität Magdeburg
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
e-mail: <Torsten.Rossnick@irb.cs.tu-magdeburg.de>

27. Sebastian Seibert
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24118 Kiel
e-mail: <Sebastian.Seibert@informatik.uni-kiel.dbp.de>

28. Helmut Seidl
Fachbereich Informatik
der Universität des Saarlandes
Am Stadtwald
66123 Saarbrücken

e-mail: <seidl@cs.uni-sb.de>

29. Stefan Skalla
Technische Universität Magdeburg
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
e-mail: <skalla@irb.cs.tu-magdeburg.de>

30. Ludwig Staiger
Lehrstuhl für Informatik II
RWTH-Aachen
Ahornstr. 55
52056 Aachen
e-mail: <staiger@zeus.informatik.rwth-aachen.de>

31. Ralf Stiebe
Technische Universität Magdeburg
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
e-mail: <Ralf.Stiebe@irb.cs.tu-magdeburg.de>

32. Wolfgang Thomas
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24118 Kiele-mail: <wt@informatik.uni-kiel.dbp.de>