University Halle-Wittenberg
Institute of Computer Science

Theorietag Automaten und Formale Sprachen 2009

Wittenberg, 27. bis 30. September 2009

Jöran Mielke, Ludwig Staiger, Renate Winter (Eds.)

Technical Report 2009/03

Institute of Computer Science
Faculty of Natural Sciences III
Martin-Luther-University Halle-Wittenberg
D-06099 Halle, Germany

WWW: http://www.informatik.uni-halle.de/preprints

LaTeX-Style: designed by Winfried Geis, Thomas Merkle, University Stuttgart
adapted by Paul Molitor, Halle
Permit granted by University Stuttgart [25/05/2007]

University Halle-Wittenberg
Institute of Computer Science

Theorietag Automaten und Formale Sprachen 2009

Wittenberg, 27. bis 30. September 2009

Jöran Mielke, Ludwig Staiger, Renate Winter (Eds.)

September 2009 **Technical Report 2009/03**

Vorwort

Die Theorietage zu Automaten und Formalen Sprachen haben bereits eine lange Tradition. 1991 von der GI-Fachgruppe 0.1.5 *Automaten und Formale Sprachen* ins Leben gerufen, finden sie jedes Jahr zusammen mit einem Workshop und der Fachgruppensitzung statt. Die Serie begann 1991 in Magdeburg, wurde 1992 in Kiel, 1993 in Dagstuhl, 1994 in Herrsching, 1995 auf Schloss Rauischholzhausen, 1996 in Cunnersdorf, 1997 in Barnstorf, 1998 in Riveris, 1999 in Schauenburg-Elmshagen, 2000 in Wien und 2001 in Wendgräben, 2002 in Wittenberg, 2003 in Herrsching, 2004 in Caputh, 2005 in Lauterbad, 2006 in Wien, 2007 in Leipzig und 2008 in Wettenberg-Launsbach durchgeführt.

In diesem Jahr findet der Theorietag mit dem vorangestellten Workshop zur Komplexitätstheorie erneut in der Lutherstadt Wittenberg statt. Bereits zum zweiten Mal wurde die Leucorea, eine Stiftung des öffentlichen Rechts an der Martin-Luther-Universität Halle-Wittenberg, als Veranstaltungsort gewählt.

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

- Christian Glaßer (Würzburg)
- Harald Hempel (Jena)
- Jacobo Torán (Ulm)
- Klaus Wagner (Würzburg)

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

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

J. Mielke

L. Staiger

R. Winter Halle und Wittenberg, im September 2009

Theorietag Automaten und Formale Sprachen 2009

Inhaltsverzeichnis

Programm des Workshops

Montag, 28. September 2009

09.25 Uhr: Begrüßung der Teilnehmer des Workshops

09.30 - 10.30 Uhr: KLAUS WAGNER (Würzburg):

Spiele und Komplexität (1. Vortrag des Workshops)

10.30 - 11.00 Uhr: Kaffeepause

11.00 - 12.00 Uhr: CHRISTIAN GLASSER (Würzburg):

Structural Properties of NP-Complete Sets (2. Vortrag des Workshops)

12.00 - 13.30 Uhr: Mittagspause

13.30 - 14.30 Uhr: JACOBO TORÁN (Ulm):

Interaktive Beweise (3. Vortrag des Workshops)

14.30 - 14.45 Uhr: Pause

14.45 - 15.45 Uhr: HARALD HEMPEL (Jena):

Alternative Lösungen und modifizierte Instanzen (4. Vortrag des Workshops)

15.45 - 16.15 Uhr: Kaffeepause

16.15 - 17.15 Uhr: GERHARD LISCHKE (Jena):

Verallgemeinerte Periodizität und Primitivität von Wörtern

Programm des Theorietages

Dienstag, 29. September 2009

09.25 Uhr: Begrüßung der Teilnehmer des Theorietages

09.30 - 09.50 Uhr: CHRISTOPHE COSTA-FLORÊNCIO, HENNING FERNAU (Leuven, Trier):

Complexity of Consistency in Categorial Grammars

09.50 - 10.10 Uhr: JOHANNES WALDMANN (Leipzig):

Gewichtete Automaten und Ableitungskomplexität

10.15 - 10.40 Uhr: Kaffeepause

10.40 - 11.00 Uhr: MARTIN KUTRIB, ANDREAS MALCHER (Gießen):

Decidability Questions on Cellular Automata

11.00 - 11.20 Uhr: HERMANN GRUBER, MARKUS HOLZER, MARTIN KUTRIB (Gießen):

On Measuring Non-Recursive Trade-Offs

11.25 - 11.45 Uhr: HERMANN GRUBER, STEFAN GULAN (Gießen, Trier):

Simplifying Regular Expressions - A Quantitative Perspective

11.45 - 12.05 Uhr: MARCEL GOEHRING (Kassel):

PC-Systems of Restarting Automata

12.05 - 13.30 Uhr: Mittagspause

13.30 - 13.50 Uhr: DOMINIK FREYDENBERGER, DANIEL REIDENBACH: (Frankfurt, Loughborough):

E-deskriptive Pattern für unendliche Sprachen

13.50 - 14.10 Uhr: JOHANNES SCHNEIDER (Kaiserslautern):

Unterschiede zwischen der Mehrdeutigkeit von nicht-löschenden und löschenden Homomorphismen

14.15 - 14.35 Uhr: TERO HARJU, DIRK NOWOTKA (Turku,Stuttgart):

Zyklisch-wiederholungsfreie Wörter über kleinen Alphabeten

14.35 - 14.55 Uhr: ANNA KASPRZIK (Trier):

Learning residual Finite-State Automata Using Observation Tables

14.55 - 15.20 Uhr: Kaffeepause

15.20 - 15.40 Uhr: RUDOLF FREUND, MARIAN KOGLER (Wien):

Drip and Mate Operations Acting in Test Tube Systems and Tissue-like P systems

15.40 - 16.00 Uhr: RUDOLF FREUND, MARIAN KOGLER, SERGEY VERLAN (Wien, Wien, Paris):

P Automata with Controlled Use of Minimal Communication Rules

16.05 - 16.25 Uhr: BIANCA TRUTHE (Magdeburg):

Ziel basierte akzeptierende Netzwerke evolitionärer Prozessoren mit regulären Filtern

16.25 - 16.45 Uhr: KARIN QUAAS (Leipzig):

On the Supports of Recognizable Timed Series

16.45 - 17.05 Uhr: CHRISTIAN MATHISSEN (Leipzig):

On the monadic quantifier alternation hierarchy over texts

17.30 - 18.30 Uhr: Fachgruppensitzung und Wahl der Fachgruppenleitung

Mittwoch, 30. September 2009

09.30 - 09.50 Uhr: HENNING BORDIHN (Potsdam):

Vergleich zweier Vergleichsmethoden für Beschreibungsgrößen

09.50 - 10.10 Uhr: FRANK WEINBERG (Kaiserslautern):

Position-and-Length-Dependent Context-free Grammars

10.15 - 10.40 Uhr: Kaffeepause

10.40 - 11.00 Uhr: JENS DOLL (Ahrensburg):

Automaten mit Termalphabeten

11.00 - 11.20 Uhr: STEFAN GÖLLER, MARKUS LOHREY (Leipzig):

Branching time model-checking of one-counter processes

11.25 - 11.45 Uhr: MANFRED KUFLEITNER, ALEXANDER LAUSER (Stuttgart):

A New Equational Description of an Infinite Hierarchy within DA

12.00 - 13.30 Uhr: Mittagspause und Ende des Theorietages

Structural Properties of NP-Complete Sets

Christian Glaßer
Lehrstuhl für Informatik IV
Universität Würzburg
97074 Würzburg, Germany
E-Mail: glasser@informatik.uni-wuerzburg.de

Many natural sets contain redundant information. The membership of an element is often connected with that of other elements, in the sense that the membership can be determined by knowing the membership of some others elements. We call this property *membership redundancy*. The talk discusses the question of whether complete sets for complexity classes like NP and PSPACE have this and other structural properties.

Membership redundancy appears in different forms and in this talk we focus on two of the simplest forms: autoreducibility and mitoticity.

Autoreducibility captures the very notion of membership redundancy: a set A is autoreducible if the membership of x in A can be determined by examining the membership of y in A for several y other than x. By controlling the manner in which an instance of a language can reduce to other instances, we obtain several variants of autoreducibility. A set A is *polynomial-time many-one autoreducible* if A polynomial-time many-one reduces to A such that on input x the reduction does not query x. Similarly one can define *polynomial-time Turing autoreducibility*.

Mitoticity captures the stronger form of membership redundancy where the set can be split into disjoint subsets that contain exactly the same information as the original set. Again, by changing the underlying reduction we obtain different notions of mitoticity. A set A is *polynomial-time many-one mitotic* if there is a set $S \in \mathrm{P}$ such that the sets A, $A_1 = A \cap S$, and $A_2 = A \cap \overline{S}$ are all polynomial-time many-one reducible to each other. Similarly one can define *polynomial-time Turing mitoticity*.

We discuss several of the results known about these concepts. For instance, mitoticity always implies autoreducibility, but the converse holds in some situations (e.g., polynomial-time many-one reducibility) and fails in others (e.g., polynomial-time Turing reducibility). As a consequence we obtain that every infinite NP-complete set splits into two disjoint, infinite NP-complete sets.

A possible application of autoreducibility and mitoticity is that they could help us to separate complexity classes. For instance, look at the following question. *Are there sets that are polynomial-time Turing complete for double-exponential time, but that are not polynomial-time Turing autoreducible?* A positive answer shows PH $\neq$ EXP, while a negative answer implies P $\neq$ PSPACE. So any answer would solve a major open problem in computational complexity.

Alternative Lösungen und modifizierte Instanzen

Harald Hempel
Instsitut für Informatik
Friedrich-Schiller-Universität Jena
07745 Jena, Germany
E-Mail: harald.hempel@uni-jena.de

Die Menschheit ist auf der Suche nach Lösungen. Das war schon immer so und wird sicherlich auch so bleiben. So suchte der Steinzeitmensch z.B. nach Lösungen für das WIE-BEWAHRT-MAN-DAS-FEUER-Problem, und heute suchen wir z.B. nach Lösungen für das GLOBAL-WARMING-Problem und das ECONOMIC-RECESSION-Problem. In der Informatik ist der Lösungsbegriff eng mit der Klasse NP verbunden: erfüllende Belegungen, kurze Rundreisen durch Graphen und Färbungen sind Beispiele für Lösungen (auch Beweise oder Zertifikate genannt) bei bekannten NP-vollständigen Berechnungsproblemem wie etwa SATISFIABILITY, TRAVELING SALES PERSON, oder COLORABILITY. Es ist allgemein bekannt, dass für die genannten und viele weitere praxisrelevante Probleme nicht nur die Berechnung einer Lösung sondern insbesondere bereits die Beantwortung der Frage nach der Existenz einer Lösung im Allgemeinen nur mit hohem Rechenaufwand möglich ist.

Aber vielleicht lohnt sich dieser hohe Rechenaufwand ja doch in der Praxis weil nachgeordnete Fragen dann umso leichter zu benatworten sind?

Im Vortrag wird zum Einen der Frage nachgegangen inwieweit die Kenntnis einer (eventuell durch großen Rechenaufwand erzielten ersten) Lösung das Finden weiterer Lösungen erleichtert. Dabei ist die erste Lösung jeweils Teil der Eingabe und somit müsste sie eine wesentliche Hilfe bei der Suche nach weiteren Lösungen sein, oder etwa nicht?

Zum Anderen wird sich der Vortrag der Frage widmen, ob die Kenntnis einer (ersten) Lösung hilfreich beim Suchen nach einer Lösung für eine leicht modifizierte Eingabe ist.

References

[ABS03] C. Archetti, L. Bertazzi, and M. G. Speranza. Reoptimzing the traveling salesman problem. *Networks*, 42(3):154–159, 2003.

[AEMP06] G. Ausiello, B. Escoffier, J. Monnot, and V. Th. Paschos. Reoptimization of minimum and maximum traveling salesman's tours. In *Algorithm theory—SWAT 2006*, pages 196–207. Springer-Verlag *Lecture Notes in Computer Science # 4059*, 2006.

[AH98] A. M. Abdelbar and S. M. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102(1):21–38, 1998.

[Ber08] T. Berg. On the Complexity of Modified Instances, 2008. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena.

[BH09] T. Berg and H. Hempel. Reoptimization of traveling salesperson problems: Changing single edge-weights. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, pages 141–151. Springer-Verlag *Lecture Notes in Computer Science #5457*, 2009.

[BHMW08] H. J. Böckenhauer, J. Hromkovič, T. Mömke, and P. Widmayer. On the hardness of reoptimization. In *Proceedings of the 34th International Conference on Current Trends in Theory and Practice of Computer Science*, pages 50–65. Springer-Verlag *Lecture Notes in Computer Science #4910*, 2008.

[BST99] C. Bazgan, M. Santha, and Z. Tuza. On the approximation of finding a(nother) Hamiltonian cycle in cubic Hamiltonian graphs. *Journal of Algorithms*, 31(1):249–268, 1999.

[dB04] M. de Bondt. On the ASP-completeness of cryptarisms. Technical Report 0419, Department of Mathematics, Radboud University of Nijmegen, 2004.

[EMP07] B. Escoffier, M. Milanič, and V. Paschos. Simple and fast reoptimizations for the steiner tree problem, 2007. Lamsade Technical Report No. 245, LAMSADE, Université Paris-Dauphin.

[KH08] M. Krüger and H. Hempel. Approximating alternative solutions. In *Proceedings of the 14th Annual International Computing & Combinatorics Conference*, pages 203–214. Springer-Verlag *Lecture Notes in Computer Science #5092*, 2008.

[Krü08] M. Krüger. On the Complexity of Alternative Solutions, 2008. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena.

[Lib04] P. Liberatore. The complexity of modified instances. *arXiv.org*, cs/0402053, 2004.

[McP03] B. P. McPhail. The complexity of puzzles: NP-completeness results for nurikabe and minesweeper. bachelor thesis, The Division of Mathematics and Natural Sciences, Reed College, Portland, 2003.

[NK92] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. Technical Report RR-92-48, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 1992.

[Sch97] M. W. Schäffter. Scheduling with forbidden sets. *Discrete Appl. Math.*, 72(1-2):155–166, 1997. Models and algorithms for planning and scheduling problems (Loveno di Menaggio, 1993).

[Set02] T. Seta. The complexity of puzzles, cross sum and their another solution problems (ASP). Senior Thesis, Department of Infomation Science, University of Tokyo, 2002.

[UN96] N. Ueda and T. Nagao. NP-completeness results for nonogram via parsimonious reductions. Technical report, Tokyo Institute of Technology, 1996.

[YS02] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. In *In Proceedings of the National Meeting of the Information Processing Society of Japan (IPSJ)*, 2002.

Interaktive Beweise

Jacobo Torán
Universität Ulm
E-Mail: jacobo.toran@uni-ulm.de

Nicht nur für die Mathematik sondern auch für das Gebiet der Informatik ist das Konzept des Beweises grundlegend. In konkreten logischen Systemen kann man Beweise genau formalisieren, aber solche klassischen formalen Beweise sind oft lang und unintuitiv.

In diesem Vortrag wird gezeigt, wie Beweise durch Interaktion zwischen zwei Parteien einfach und effizient werden können. Ein interaktiver Beweis kann erklärt werden als Spiel zwischen zwei Spielern, in dem ein Spieler den anderen zu überzeugen versucht.

Beispiele von interaktiven Beweisen und Anwendungen dieses Konzeptes beleuchten verschiedene Aspekte der Informatik-Gebiete Algorithmik, Programmverifikation und Kryptologie.

Spiele und Komplexität

Klaus Wagner
Institut für Informatik
Julius-Maximilians-Universität Würzburg
97074 Würzburg, Germany
E-Mail: wagner@informatik.uni-wuerzburg.de

In diesem Vortrag sollen sowohl klassische Resultate zu den engen Beziehungen zwischen kombinatorischen Spielen und der Komplexität von parallelen Berechnungen als auch sehr aktuelle Entwicklungen zur Komplexität von Nash-Gleichgewichten von Spielen dargestellt werden.

Ein Spieler in einem Zweipersonenspiel hat eine sichere Gewinnstrategie, wenn er auf jeden Zug seines Gegners einen Zug kennt, der so beschaffen ist, daß er auf jeden nächsten Zug seines Gegners seinerseits einen nächsten Zug kennt, so daß ... Am Ende muß er bei jeder Zugfolge seines Gegners in eine Gewinnsituation gelangen. Dieses Wechelspiel von „für jeden“ und „es existiert“ entspricht genau den Bedingungen, die man an die Akzeptierung einer Eingabe durch sogenannte *alternierende* Maschinen oder Algorithmen stellt.

Alternierende Maschinen sind ein einfaches theoretisches Modell für parallel arbeitende Machinen oder Algorithmen. Sie entsprechen jedoch in ihrer Leistung der vollen Leistung komplizierterer Modelle paralleler Maschinen. Es gibt interessante Beziehungen zwischen den Komplexitätsklassen alternierender Maschinen und denen deterministischer Maschinen.

Es stellt sich nun heraus, daß man die Akzeptierung von Eingaben durch alternierende Maschinen auch als Zweipersonenspiel ansehen kann. Es gibt also eine Entsprechung in beiden Richtungen. So ist es auch nicht verwunderlich, daß das Problem der Existenz sicherer Gewinnstrategien für bekannte Spiele wie Schach, Dame und Go vollständig für bestimmte Komplexitätsklassen alternierender Maschinen sind, dort also zu den schwierigsten und charakteristischen Problemen gehören.

Für Einpersonenspiele wie Puzzle, Minesweeper, Tetris ist die Frage nach der Existenz einer Gewinnstrategie im allgemeinen einfacher. Aber auch hier kann man in vielen Fällen die Komplexität dieser Frage durch Vollständigkeitsresultate charakterisieren.

Ein Nash-Gleichgewicht bei einem Mehrpersonenspiel ist eine Situation, in der kein Spieler seine Lage dadurch verbessern kann, daß er einseitig sein Verhalten ändert. Oft stellen sich bei unregulierten ökonomischen Vorgängen Nash-Gleichgewichte ein (was nicht unbedingt heißt, daß diese Situationen für die Beteiligten vorteilhaft sind). Für verschiedene Typen von Spielen wird die Komplexität der Frage nach der Existenz von Nash-Gleichgewichten beschrieben.

Vergleich zweier Vergleichsmethoden für Beschreibungsgrößen

Henning Bordihn
Institut für Informatik, Universität Potsdam
D-14482 Potsdam, Germany
E-Mail: henning@cs.uni-potsdam.de

Für viele Familien formaler Sprachen gibt es verschiedene Mechanismen, die sie charakterisieren. Als Beispiele können reguläre Ausdrücke, deterministische und nichtdeterministische endliche Automaten sowie rechtslineare Grammatiken betrachtet werden. In solchen Fällen stellt sich die Frage, welche der Mechanismen eine geringere Beschreibungsgröße aufweisen, wenn dieselben Sprachen beschrieben werden. Etwas allgemeiner kann die Frage wie folgt formuliert werden: Seien $\mathcal{G}_1$ und $\mathcal{G}_2$ zwei sprachbeschreibende Mechanismen und $\mathcal{L}_1$ und $\mathcal{L}_2$ die zugehörigen Sprachfamilien, welcher Mechanismus ist kompakter hinsichtlich gewisser Parameter, wenn Sprachen aus $\mathcal{L}_1 \cap \mathcal{L}_2$ beschrieben werden. In der Literatur gibt es verschiedene Ansätze, um zwei Mechanismen in dieser Hinsicht miteinander zu vergleichen. Hier werden zwei typische solcher Methoden zueinander in Bezeihung gesetzt: nicht-rekursive Tradeoffs und die klassischen Vergleichsoperatoren hnsichtlich verschiedener Größenparameter nach Gruska. Die gewonnenen Erkenntnisse werden angewendet, um neue Erkenntnisse über den Vergleich der Beschreibungsgrößen von CD Grammatiksystemen und Chomsky-Grammatiken zu gewinnen.

Complexity of Consistency in Categorial Grammars

C. Costa-Florêncio
Department of Computer Science
Katholieke Universiteit Leuven, Belgium
E-Mail: christophe.costaflorencio@cs.kuleuven.be

H. Fernau
FB 4—Abteilung Informatik
Universität Trier
54286 Trier, Germany
E-Mail: fernau@uni-trier.de

We study consistency problems of the following form for some family $(\mathcal{L}_k)$ of language classes: Given a finite language F and some parameter k, is there some language $L \supseteq F$ contained in $\mathcal{L}_k$? We derive non-trivial complexity results for languages of structures of categorial grammars, in terms of classical complexity, as well as in terms of parameterized complexity and approximability.

1 Introduction

For many families of language classes, the described consistency problem is trivial. Consider for example the class $\mathcal{A}_k$ of languages that can be accepted by a finite automaton with at most k states. Then, we can always answer YES to the consistency problem, since if $F \subseteq \Sigma^*$ was given, then the automaton with one state that accepts Σ^* would justify this. However, this trivial type of reply is no longer possible if the universal language is not (automatically) in each of the language classes of interest. Examples are provided by categorial grammars, a formalism known to be (weakly) equivalent to context-free grammars, see [CF03].

The language families $\mathcal{L}_k$ are usually defined via grammar families $\mathcal{G}_k$. As a variant of the mentioned consistency problem, we may be given a finite set of derivation structures and some parameter k and ask if there is a grammar $G \in \mathcal{G}_k$ that produces those structures (and possibly more). Note that this can be also seen as a special case of the first problem formulation, if we consider *languages of structures*.

We study the computational complexity of consistency problems both from a classical (P vs. NP) perspective, as well as from the perspective of parameterized complexity and approximability.

Apart from the sketched complexity results, we also introduce new families of categorial languages, based by the so-called sum-value of a categorial grammar (details can be found in the next

section) that are interesting on their own right, since they form a new strict hierarchy of categorial languages, where each level is learnable from positive samples. So, the new results presented in this paper touch and bridge three areas of research: complexity, formal languages, and learning theory (resp., grammatical inference).

2 Results

Due to space restrictions, it is impossible to provide all necessary definitions. We therefore only sketch those as far as it is necessary to state our results.

A *categorial grammar* G can be viewed as a mapping from Σ into finite subsets of types T. Accordingly, we can associate a *value function* v that maps $a \in \Sigma$ onto $|G(a)|$, i.e., the number of types that G maps to a. As discussed in [DF08], at least two natural size measures can be derived from v, depending on the chosen metric: The *max-value* of G is $\max_{a\in\Sigma} v(a)$, while the *sum-value* of G is $\sum_{a\in\Sigma} v(a)$. A categorial grammar G is called *k-max-valued* (*k-sum-valued*, resp.) if its max-value (sum-value, resp.) is upper-bounded by k. The according grammar classes are denoted by $\mathcal{G}_{k-\text{max-val}}$ and $\mathcal{G}_{k-\text{sum-val}}$, resp. In the literature [CF03, Kan98], k-max-valued grammars are also known as k-valued grammars. 1-max-valued grammars are also known as *rigid grammars*. The languages that can be described with k-max-valued (k-sum-valued, resp.) grammars are comprised in the classes $\mathcal{L}_{k-\text{max-val}}$ and $\mathcal{L}_{k-\text{sum-val}}$, resp. If we want to fix the alphabet to Σ, we arrive at the language classes $\mathcal{L}^{\Sigma}_{k-\text{max-val}}$ and $\mathcal{L}^{\Sigma}_{k-\text{sum-val}}$, resp. The languages of structures that can be generated by grammars from $\mathcal{G}_{k-\text{max-val}}$ and $\mathcal{G}_{k-\text{sum-val}}$, resp., are written $\mathcal{FL}_{k-\text{max-val}}$ and $\mathcal{FL}_{k-\text{sum-val}}$, resp.

Kanazawa [Kan98] has proved the following hierarchy result:

Thm. 1 *For any alphabet Σ and any $k \geq 1$, $\mathcal{L}^{\Sigma}_{k-max\text{-}val} \subsetneq \mathcal{L}^{\Sigma}_{(k+1)-max\text{-}val}$.*

We complement this result by showing:

Thm. 2 *For any alphabet Σ and any $k \geq 1$, $\mathcal{L}^{\Sigma}_{k-sum\text{-}val} \subsetneq \mathcal{L}^{\Sigma}_{(k+1)-sum\text{-}val}$.*

We also have analogous results for $\mathcal{FL}_{k-\text{max-val}}$ and $\mathcal{FL}_{k-\text{sum-val}}$, resp.

The language families that we defined are particularly interesting from the point of view of learning theory. They provide non-trivial examples of language families of finite elasticity (see [Kan98] for details), so that we can state:

Thm. 3 *Each family from the hierarchies $\mathcal{L}_{k-max\text{-}val}$, $\mathcal{L}_{k-sum\text{-}val}$, $\mathcal{FL}_{k-max\text{-}val}$ and $\mathcal{FL}_{k-sum\text{-}val}$ is learnable from text, i.e., given positive samples only.*

Parameterized complexity makes sense in particular if the parameter of interest can be assumed to be small. The hierarchy level k in our formulation of the consistency problem might be such a small parameter. So, we arrive at problems that we call, for instance, $\mathcal{L}_{k-\text{MAX-VAL}}$-CONSISTENCY in order to make the parameter explicit. However, we cannot always hope for finding nice parameterized algorithms. More specifically, we can derive as a corollary from [CF03, Theorem 5.32]:

Cor. 4 *Unless $P = NP$, there is no FPT algorithm that decides $\mathcal{L}_{k-\text{MAX-VAL}}$-CONSISTENCY.*

Thm. 5 *$\mathcal{L}_{k-\text{MAX-VAL}}$-CONSISTENCY and $\mathcal{FL}_{k-\text{MAX-VAL}}$-CONSISTENCY are W[2]-hard.*

Cor. 6 *There are constants $c_1, c_2 > 0$ such that $\mathcal{L}_{k-\text{MAX-VAL}}$-CONSISTENCY and $\mathcal{FL}_{k-\text{MAX-VAL}}$-CONSISTENCY cannot be approximated up to a factor of $c_1 \log(n)$ and $c_2 \log(n)$, respectively, unless $NP \subset DTime(n^{\log\log(n)})$.*

Thm. 7 *$\mathcal{L}_{k-\text{SUM-VAL}}$-CONSISTENCY and $\mathcal{FL}_{k-\text{SUM-VAL}}$-CONSISTENCY are NP-complete. This is also true when bounding the alphabet size.*

Thm. 8 *$\mathcal{L}_{k-\text{SUM-VAL}}$-CONSISTENCY and $\mathcal{FL}_{k-\text{SUM-VAL}}$-CONSISTENCY are W[1]-hard.*

We could also provide more positive (FPT) results when introducing an additional step parameter.

References

[CF03] C. Costa Florêncio. *Learning Categorial Grammars.* PhD thesis, Universiteit Utrecht, The Netherlands, 2003.

[DF08] J. Dassow and H. Fernau. Comparison of some descriptional complexities of 0L systems obtained by a unifying approach. *Information and Computation*, 206:1095–1103, 2008.

[Kan98] M. Kanazawa. *Learnable Classes of Categorial Grammars.* PhD, CSLI, 1998.

Automaten mit Termalphabeten

Jens-D. Doll
Universität Hamburg
E-Mail: jens.doll@studium.uni-hamburg.de

Im Vortrag soll eine Automatenklasse mit unendlichem Alphabet vorgestellt werden, die für die Verifikation von Software relevant ist.

---

1 Einzelheiten

Ausgehend von einem Termalphabet über einer Trägermenge werden Automaten mit solchen Alphabeten betrachtet und ein Weg aufgezeigt, wie man diese Automaten reduzieren und deren Terminierung beweisen kann. Die zu den Automaten gehörige Sprache liegt in L(0) und das Reduktionsverfahren basiert auf speziellen Operatoren und einer zugehörigen Algebra. Die existierenden Bezüge zwischen formalen Sprachen und der Algebra werden durch diese Automaten transparent.

Ein allgemeines Verfahren liefe dem Halteproblem und dem Gödelschen Unvollständigkeitssatz zuwider, ein eingeschränktes Verfahren widerspräche dem Satz von Rice. Für definierbare Sprachklassen gibt es jedoch empirische Beweise und intuitive Plausibilität. An der formalen Beweisführung arbeitet der Verfasser zur Zeit. Der Vortrag ist in die Teile Definition, Reduktion und Empirik aufgeteilt.

1.1 Definition der Automaten

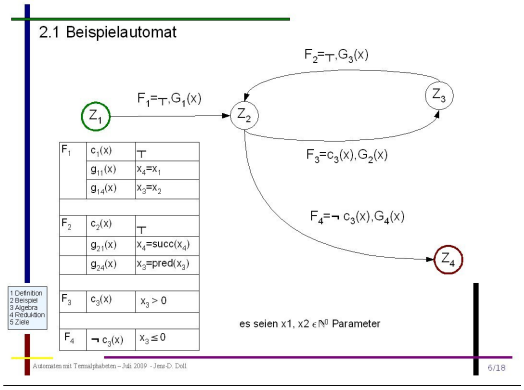
Im ersten Teil wird das formale Gerüst der Termautomaten dargestellt und an einem Beispielautomaten erläutert. Dann werden Operatoren definiert, aus denen sich ein formalsprachliches Äquivalent für Termautomaten bilden läßt.

1.2 Skizzierung des Reduktionsverfahrens

Aus den drei Operatoren Komposition, Disjunktion und Integration lassen sich reguläre Ausdrücke bilden, die das formalsprachliche Gerüst für jegliche Berechenbarkeit bilden. Die sogenannte diskrete Integration bildet die Grundlage für ein Reduktionsverfahren, das Termautomaten in eine funktionale Darstellung umwandelt. Anschließend werden die bestehenden Grenzen des Reduktionsverfahrens aufgezeigt.

1.3 Empirische Beweise

Für die Termautomaten und das Reduktionsverfahren gibt es seit 2005 eine prototypische Implementierung, die grundsätzlich für alle prozeduralen Sprachen geeignet ist. Es können Beispielrechnungen für algebraische Funktionen gezeigt werden, die den Stammbaum algebraischer Funktionen verifizieren.



Drip and Mate Operations Acting in Test Tube Systems and Tissue-like P systems

Rudolf Freund

Institute of Computer Languages, Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria

E-Mail: rudi@emcc.at

Marian Kogler

Institute of Computer Languages, Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria

E-Mail: marian@emcc.at

The operations drip and mate considered in (mem)brane computing resemble the operations cut and recombination well known from DNA computing. We here consider sets of vesicles with multisets of objects on their outside membrane interacting by drip and mate in two different setups: in test tube systems, the vesicles may pass from one tube to another one provided they fulfill specific constraints; in tissue-like P systems, the vesicles are immediately passed to specified cells after having undergone a drip or mate operation. In both variants, computational completeness can be obtained, yet with different constraints for the drip and mate operations.

1 DNA and (Mem)brane Computing with Strings

Whereas in (tissue) P systems the objects are placed inside the membranes, in the variant of brane systems introduced by Luca Cardelli (see [Car05]), the objects are placed on the membranes. The computations in these models also called *brane calculus* are based on specific ways to divide and fuse membranes and to redistribute the objects on the membranes, the rules usually being applied in a sequential way in contrast to the (maximally) parallel way of applying rules in P systems. Computational completeness was shown for (mem)brane systems with the mate and drip operations working on strings placed on the membranes in [FO07] and for test tube systems using cutting and recombination with the minimal number of two test tubes in [FF96].

2 Test Tube Systems and Tissue-like P Systems with Mate and Drip Rules Working on Vesicles Carrying Multisets of Objects

By simulating deterministic register machines, computational completeness can be shown for test tube systems and tissue-like P systems with drip and mate rules:

$$TTS_m\left(axiom_l, mate_p\right)(k) = PsRE(k) \quad \text{for all} \quad \mathrm{m} \geq 3, \mathrm{l} \geq 3, \mathrm{p} \geq 5, \mathrm{k} \geq 1$$

and, by trading the usage of drip rules for axiom complexity,

$$TTS_m\left(axiom_l, drip_q, mate_p\right)(k) = PsRE(k) \quad \text{for all} \quad \mathrm{m} \geq 3, \mathrm{l} \geq 1, \mathrm{p} \geq 5, \mathrm{q} \geq 4, \mathrm{k} \geq 1,$$

i.e., test tube systems with three test tubes, mate rules of size five and either axioms of size three or drip rules of size four together with axioms of size one are computationally complete.

For tissue-like P systems we need at least five cells as well as mate and drip rules of size five and axioms of size three, i.e.,

$$tP_m\left(axiom_l, drip_q, mate_p\right)(k) = PsRE(k) \quad \text{for all} \quad \mathrm{m} \geq 5, \mathrm{l} \geq 3, \mathrm{p} \geq 5, \mathrm{q} \geq 5, \mathrm{k} \geq 1.$$

References

[Car05] L. Cardelli. Brane calculi. Interactions of biological membranes. In *Computational Methods in Systems Biology: International Conference CMSB 2004*, pages 257–280, Berlin, Germany, 2005. Springer-Verlag.

[FF96] R. Freund and F. Freund. Test tube systems or How to bake a DNA cake. *Acta Cybernetica*, 12:445–459, 1996.

[FO07] R. Freund and M. Oswald. Tissue P systems and (mem)brane systems with mate and drip operations working on strings. *Electronic Notes in Theoretical Computer Science*, 171 (1):105–115, 2007.

P Automata with Controlled Use of Minimal Communication Rules

Rudolf Freund
Institute of Computer Languages, Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
E-Mail: rudi@emcc.at

Marian Kogler
Institute of Computer Languages, Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
E-Mail: marian@emcc.at

Sergey Verlan
LACL, Département Informatique
UFR Sciences et Technologie, Université Paris XII
61, av. Général de Gaulle, 94010 Créteil, France
E-Mail: verlan@univ-paris12.fr

We introduce new variants of transition modes for P systems by adding rule control to the conventional transition modes used so far. This formalism of rule control allows us to specify which rules have to be applied together or not. We show that computational completeness can be obtained by using either minimal symport rules or minimal antiport rules together with uniport rules being applied in the maximally parallel transition mode with rule control.

---

1 P Automata

We consider some new special variants of P automata – a variation of P systems introduced in [CVV03] which act as acceptors rather than producers; a similar concept of analyzing P systems was considered in [FO02]. A multiset is accepted by a P automaton if and only if the automaton halts when having been started with the multiset present in the input membrane.

Here we extend this formalism by imposing an additional condition on the applicable multisets of rules, i.e., we specify which rules have to (or can) be used together.

We introduce an additional control mechanism on the applicability of rules within a multiset of rules operating as an "overlay" on the transition modes. As special forms of communication rules, we consider symport rules and antiport rules.

For our control mechanism, we consider a partitioning of the rule set into n non-empty, but not necessarily disjoint sets and a set of vectors with arity n. The components with value 1 in a vector specify those rule sets from which at least one rule has to be taken into a multiset of rules

to be applied, whereas from the components with value 0 in the control vector no rule is allowed to be applied.

2 Computational Completeness

We consider communication rules (see also [PP02]), namely symport rules and antiport rules:

Symport rules in R are of the form $x[_i \rightarrow [_i x$ meaning that the multiset x from outside membrane i is moved into the region inside membrane i, or $[_i x \rightarrow x[_i$ meaning that the multiset x from inside membrane i is moved into the region surrounding membrane i, with x being an object and i the label of a membrane. The weight of a symport rule is $|x|$. Symport rules with weight n are called sym_n rules; sym_1 rules are called *uniport* rules.

Antiport rules in R are of the form $x[_i y \rightarrow y[_i x$ meaning that the multiset x from outside membrane i is exchanged with the multiset y in the region inside membrane i, with x and y being objects and i the label of a membrane. The weight of an antiport rule is defined as $max(|x|, |y|)$. Antiport rules with weight n are called $anti_n$ rules.

By simulating deterministic register machines, we show that communication P automata with rule control using only minimal symport rules (sym_2 rules) or minimal antiport rules ($anti_1$ rules) together with uniport rules (sym_1 rules) in only one membrane are able to accept any recursively enumerable set of vectors of natural numbers.

References

[CVV03] E. Csuhaj-Varjú and G. Vaszil. P automata or purely communicating accepting P systems. In *WMC-CdeA '02: Revised Papers from the International Workshop on Membrane Computing*, pages 219–233, London, UK, 2003. Springer-Verlag.

[FO02] R. Freund and M. Oswald. A short note on analysing P systems with antiport rules. *Bulletin of the EATCS*, 78:231–236, 2002.

[PP02] A. Păun and G. Păun. The power of communication: P systems with symport/antiport. *New Gen. Comput.*, 20(3):295–305, 2002.

E-deskriptive Pattern für unendliche Sprachen

Dominik D. Freydenberger
Institut für Informatik
Goethe-Universität
Postfach 111932, 60054 Frankfurt am Main, Deutschland
E-Mail: freydenberger@em.uni-frankfurt.de

Daniel Reidenbach
Department of Computer Science
Loughborough University
Loughborough, Leicestershire, LE11 3TU, Großbritannien
E-Mail: D.Reidenbach@lboro.ac.uk

Deskriptive Pattern sind minimale und leicht interpretierbare Generalisierungen von Mengen von Wörtern. Die vorliegende Arbeit konstatiert, daß es unendliche Sprachen gibt, die sich nicht in dieser optimalen Weise approximieren lassen.

---

1 Deskriptive Pattern

Ein *Pattern* ist eine endliche Zeichenkette über einem Variablenalphabet $X = \{x, y, z, x_i \mid i \in \mathbb{N}\}$ und einem Terminalalphabet $\Sigma \supseteq \{\mathtt{a}, \mathtt{b}\}$. Ein solches Pattern α ist *konsistent* mit einer Menge $S \subseteq \Sigma^*$, wenn α auf jedes Wort in S durch eine *Substitution* – d. h. einen Homomorphismus $\sigma : (X \cup \Sigma)^* \to \Sigma^*$, der $\sigma(a) = a$ für jedes $a \in \Sigma$ erfüllt – abgebildet werden kann. So sind beispielsweise die Pattern $\alpha_0 := x$, $\alpha_1 := xyxyx$ und $\alpha_2 := x\mathtt{ab}y$ konsistent mit der Menge $S_0 := \{\mathtt{ababa}, \mathtt{ababbababbab}, \mathtt{babab}\}$. Konsistente Pattern liefern also eine kompakte und leichtverständliche Darstellung von Gemeinsamkeiten der Wörter einer Menge.

Wie obiges Beispiel zeigt, gibt es zu einer Wortmenge im allgemeinen eine Vielzahl von konsistenten Pattern, die intuitiv eine sehr unterschiedliche Güte haben können. Es ist daher notwendig, konsistente Pattern hoher Qualität formal zu fassen. Eine natürliches, einschlägiges Konzept basiert auf der *Patternsprache* $L(\alpha)$ eines Pattern α, d. h. der Menge *aller* Wörter in einem freien Monoid Σ^*, mit denen α konsistent ist. Ein Pattern δ heißt dann *deskriptiv* für eine Wortmenge S, wenn δ konsistent ist mit S und es kein Pattern α gibt, das $S \subseteq L(\alpha) \subset L(\delta)$ erfüllt, dessen Patternsprache also S präziser approximieren könnte, als die Sprache von δ dies bewerkstelligt. Es wird hierbei zwischen NE- und E-deskriptiven Pattern unterschieden. Erstere beschränken die zugrundeliegende Definition von Patternsprachen auf nichtlöschende Substitutionen („NE-Patternsprachen“), während letztere beliebige Substitutionen zulassen („E-Patternsprachen“).

Aufgrund von [Ang80] ist bekannt, daß es zu jeder Sprache ein NE-deskriptives Pattern gibt. [JKS+94] zeigt, daß für jede *endliche* Wortmenge auch ein E-deskriptives Pattern existiert, und fragt, ob dasselbe Ergebnis für *unendliche* Sprachen gilt. Dies möchten wir im folgenden beantworten. Eine ausführlichere Darstellung unserer Resultate findet sich im Tagungsband der Konferenz DLT 2009 (siehe [FR09]).

2 E-deskriptive Pattern für unendliche Sprachen

Da deskriptive Pattern eine minimale Generalisierung einer Wortmenge S sind, kann es sie nur dann *nicht* geben, wenn kein mit S konsistentes Pattern eine „kleinste“, S umfassende Patternsprache erzeugt. Es gibt also genau dann kein deskriptives Pattern für S, wenn jede Patternsprache L mit $L \supseteq S$ in einer unendlichen, echt absteigenden Kette von Patternsprachen enthalten ist.

Unsere Argumentation basiert auf einer spezifischen Folge von Pattern, die wie folgt definiert ist: $\alpha_0 := y^2z^2$ und $\alpha_i := \phi(\alpha_{i-1})$, $i \geq 1$, wobei $\phi : X^* \to X^*$ ein Homomorphismus ist, der durch $\phi(x_i) := x_{i+1}$, $\phi(y) := y^2x_1$ und $\phi(z) := x_1z^2$ gegeben ist. Die solchermaßen definierten Pattern erzeugen eine unendliche, echt absteigende Kette von E-Patternsprachen. Mit Hilfe dieser Kette läßt sich die folgende Erkenntnis gewinnen:

Satz 1 *Zu jedem Alphabet Σ mit $\Sigma \geq 2$ existiert eine unendliche Sprache $L_\Sigma \subset \Sigma^*$, die kein E-deskriptives Pattern besitzt.*

Der Beweis des Satzes basiert auf $L_\Sigma := \bigcup_{i=0}^{\infty} L(\psi(\alpha_i))$, wobei der Homomorphismus $\psi : X^* \to X^*$ durch $\psi(x_i) := x_i$ und $\psi(y) := \psi(z) := x_0$ definiert ist. Für jedes α_i gilt dann $L(\alpha_i) \supseteq L_\Sigma$. Außerdem kann gezeigt werden, daß zu jedem Pattern α mit $L(\alpha) \supseteq L_\Sigma$ ein α_i existiert, das $L(\alpha_i) \subseteq L(\alpha)$ erfüllt, was den Beweis beschließt.

Literatur

[Ang80] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[FR09] D.D. Freydenberger und D. Reidenbach. Existence and nonexistence of descriptive patterns. In *Proc. 13th International Conference on Developments in Language Theory, DLT 2009*, *Lecture Notes in Computer Science* 5583, S. 228–239, 2009.

[JKS+94] T. Jiang, E. Kinber, A. Salomaa, K. Salomaa und S. Yu. Pattern languages with and without erasing. *International Journal of Computer Mathematics*, 50:147–163, 1994.

PC-Systems of Restarting Automata

Marcel Goehring
Fachbereich Elektrotechnik/Informatik
Universität Kassel
34109 Kassel, Germany
E-Mail: goehring@theory.informatik.uni-kassel.de

The idea of parallel communicating (PC for short) components was applied to different kinds of models: PC grammar systems, PC systems of finite automata, PC systems of pushdown automata, etc. Mostly the computational power of such a team is larger than that of the components themselves. On the other hand so-called cooperating distributed Restarting Automata were considered and it has been shown that they are more expressive than their components. I want to present an approach of how to carry the idea of parallel communication over to Restarting Automata and report on some observations made so far.

The model of Restarting Automata was discussed in many publications (see [Ott06]) and a lot of variants have been pointed out. One of these variants are the so-called cooperating distributed systems (CD-Systems for short) of Restarting Automata (see [Mes07], [MO07]), where Restarting Automata work together in a sequential manner. Another approach of cooperation is the idea of parallel communicating components, so-called PC systems, that was already considered for several kinds of models like PC grammar systems ([RS97]), PC systems of finite automata ([MMM02]), PC systems of pushdown automata ([CMMV00]), etc. But how can the idea of parallel communication be carried over to Restarting Automata? What advantages over a single component can be achieved? How much communication is needed? There are a lot of questions that can be asked in this context.

Formally a PC-System of Restarting Automata of type $X \in \{(det-)((w)mon-)RL(W)(W), (det-)((w)mon-)R(R)(W)(W)\}$ is a finite set $\mathcal{M} := (M_i)_{i \in I}$ (I is a set of indices) of Restarting Automata $M_i = (Q_i, \Sigma, \Gamma_i, ¢, \$, q_0^{(i)}, k, \delta_i)$ ($i \in I$) of type X. The communication is realised by communication states included in the set of states Q_i of the components. There are request states, response states and receive states. The communication works as follows: if a component M_i sends a request to another component M_j by entering the request state req_j and the component M_j sends a response to M_i by entering the response state res_i^l, then automaton M_i enters the corresponding receive state rec_j^l, whereby l is some information of constant length. Until the corresponding

request or response is sent the communicating component waits (and is blocked). After a succesful communication both components go on with their local computations. In particular there does not exist any other synchronisation mechanism than the communication (there is no global clock or anything like that).

We show that some languages like the Gladkij language and the copy language can be accepted by a deterministic PC-System of R-automata, although these languages are not growing context sensitive and therefore cannot be accepted by a single det-R-automaton. The exponential language can also be accepted by a deterministic PC-System of R-automata without auxiliary symbols, though it cannot be accepted by any single Restarting Automaton without auxiliary symbols. Moreover we found that PC-Systems of Restarting Automata are as expressive as nonforgetting Restarting Automata (see [MO06], [Mes07]) and CD-Systems of Restarting Automata.

References

[CMMV00] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, and G. Vaszil. Parallel communicating pushdown automata systems. *Int. J. Found. Comput. Sci.*, 11(4):633–650, 2000.

[Mes07] H. Messerschmidt. *CD-Systems of Restarting Automata*. PhD thesis, Universität Kassel, 2007.

[MO06] H. Messerschmidt and F. Otto. On nonforgetting restarting automata that are deterministic and/or monotone. In *Computer Science – Theory and Applications*, volume 3967, pages 247–258. Springer Berlin / Heidelberg, 2006.

[MO07] H. Messerschmidt and F. Otto. Strictly deterministic CD-Systems of restarting automata. In *Fundamentals of Computation Theory*, volume Volume 4639/2007, pages 424–434. Springer Berlin / Heidelberg, 2007.

[MMM02] C. Martín-Vide, A. Mateescu, and V. Mitrana. Parallel finite automata systems communicating by states. *Int. J. Found. Comput. Sci.*, 13(5):733–749, 2002.

[Ott06] F. Otto. Restarting automata. In Zoltán Esik, Carlos Martin-Vide, and Victor Mitrana, editors, *Recent Advances in Formal Languages and Applications*, volume 25 of *Studies in Computational Intelligence*, pages 269–303. Springer, Berlin, 2006.

[RS97] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, vol. 2: linear modeling: background and application*. Springer, New York, NY, USA, 1997.

Branching time model-checking of one-counter processes

Stefan Göller
Fachbereich Mathematik und Informatik
Universität Bremen
Bremen, Germany
E-Mail: goeller@informatik.uni-bremen.de

Markus Lohrey
Institut für Informatik
Universität Leipzig
Leipzig, Germany
E-Mail: lohrey@informatik.uni-leipzig.de.

One-counter processes are pushdown processes which operate only on a unary stack alphabet. We study the computational complexity of model-checking Computation Tree Logic (CTL) over one-counter processes. While a PSPACE upper bound is inherited from the modal μ-calculus for this problem [5], we provide corresponding lower bounds. First, we prove that already over some fixed one-counter process, CTL model-checking is hard for PSPACE. However, if we furthermore restrict the input formulas to have fixed nesting depth of the until operator, we can prove a polynomial time upper bound for this problem. Second, we show that there already exists a fixed CTL formula which model-checking of one-counter processes is PSPACE-hard for. To obtain the latter result, we employ two results from computational complexity: (i) Converting a natural number in chinese remainder presentation into binary presentation is in logspace-uniform NC^1 [2] and (ii) PSPACE is AC^0-serializable [4], see also [6]. We demonstrate that our approach can be used to obtain further results. We show that model-checking CTL's fragment EF is hard for P^{NP}, thus establishing a matching lower bound and answering an open question from [3]. We moreover show that the following problem is hard for PSPACE: Given a one-counter Markov decision process, a set of target states with counter value zero each, and an initial state, to decide whether the probability that the initial state will eventually reach one of the target states is arbitrarily close to 1. This improves a previously known lower bound for every level of the boolean hierarchy shown in [1] Brazdil et al.

References

[1] T. Brazdil, V. Brozek, K. Etessami, A. Kucera, and D. Wojtczak. One-Counter Markov Decision Processes. Technical report, arXiv.org, 2009. http://arxiv.org/abs/0904.2511.

[2] A. Chiu, G. Davida, and B. Litow. Division in logspace-uniform NC^1. *Theoretical Informatics and Applications. Informatique Théorique et Applications*, 35(3):259–275, 2001.

[3] S. Göller, R. Mayr, and A. W. To. On the computational complexity of verifying one-counter processes. In *Proc. of LICS 2009*. IEEE Computer Society Press, 2008. to appear.

[4] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference (San Diego, CA, 1993)*, pages 200–207. IEEE Computer Society Press, 1993.

[5] O. Serre. Parity games played on transition graphs of one-counter processes. In L. Aceto and A. Ingólfsdóttir, editors, *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2006), Vienna (Austria)*, number 3921 in Lecture Notes in Computer Science. Springer, 2006.

[6] H. Vollmer. A generalized quantifier concept in computational complexity theory. Technical report, arXiv.org, 1998. http://arxiv.org/abs/cs.CC/9809115.

Simplifying Regular Expressions. A Quantitative Perspective

Hermann Gruber
Institut für Informatik, Universität Gießen
Arndtstraße 2, D-35392 Gießen, Germany
E-Mail: hermann.gruber@informatik.uni-giessen.de

Stefan Gulan
Fachbereich IV—Informatik, Universität Trier
Campus II, D-54296 Trier, Germany
E-Mail: gulan@uni-trier.de

We propose a new normal form for regular expressions which tightly bounds the ratio of two common size measures for regular expressions. We also give a conversion from regular expressions to ε-NFAs, which implicitly computes this normal form while maintaining an optimal ratio of expression-to-automaton-sizes. This allows us to resolve a problem posed by Ilie and Yu [4].

1 Definitions and Constructions

Regular expressions, *expressions* for brevity, may not contain ε or $\emptyset$ and are otherwise defined as usual with the additional operator ?, where $L(r^?) = \{\varepsilon\} \cup L(r)$. If every subexpression $s^?$ of r satisfies $\varepsilon \notin L(s)$, we call r *mildly simplified*. The number of leaves in the parse of r is denoted $\mathrm{alph}(r)$, the number of nodes $\mathrm{arpn}(r)$; further, let $\mathrm{rpn}(r)$ equal $\mathrm{arpn}(r)$ plus the number ?s occuring in r. Let $\mathrm{alph}(L) = \min\{\mathrm{alph}(r) \mid L(r) = L\}$; $\mathrm{rpn}(L)$ and $\mathrm{arpn}(L)$ are defined accordingly.

The operators $\circ$ and $\bullet$ are defined as: $a^\circ = a$, $(r+s)^\circ = r^\circ + s^\circ$, $r^{?\circ} = r^\circ$, $r^{*\circ} = r^{\circ *}$, if $\varepsilon \notin L(rs)$ then $(rs)^\circ = rs$, else $(rs)^\circ = r^\circ + s^\circ$; $a^\bullet = a$, $(r+s)^\bullet = r^\bullet + s^\bullet$, $(rs)^\bullet = r^\bullet s^\bullet$, $r^{*\bullet} = r^{\bullet\circ *}$, if $\varepsilon \in L(r)$ then $r^{?\bullet} = r^\bullet$, else $r^{?\bullet} = r^{\bullet ?}$. We call $r^\bullet$ the *strong star normal form* of r (cf. [1]).

We construct εNFAs from expressions by graph rewritings (Figs. 1,2), taken from [3], with additional precedences. Let $A(r)$ denote any automaton constructed this way, its size $|A(r)|$ is the combined number of states and transitions.

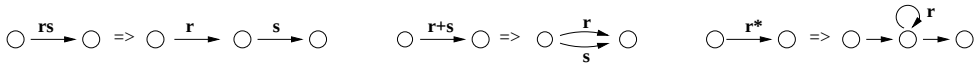


Figure 1: Introducing states/transitions while deconstructing the input in labels.

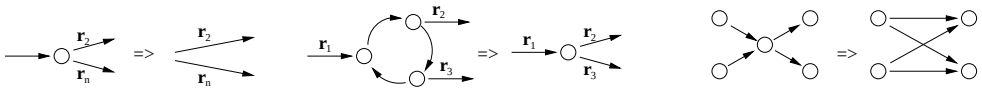


Figure 2: Removing redundant ε-transitions (unlabeled arcs) and incident states.

2 Results

Theorem 2.1. *Any regular language L satisfies* $\mathrm{rpn}(L) \leq 4\,\mathrm{alph}(L) - 1$.

This improves on previous bounds ([2, 4]) of $\mathrm{rpn}(L)$ wrt. $\mathrm{alph}(L)$. The concept of strong star normal form is crucial in the proof. This normal form is implicitly computed upon converting a mildly simplified expression into an εNFA.

Theorem 2.2. *Let r be mildly simplified, then* $A(r) = A(r^\bullet)$.

The precondition poses no severe restriction, since any r can be transformed in linear time into a mildly simplified r', s.t. $L(r) = L(r')$ and $|A(r')| \leq |A(r)|$. The size of an εNFA constructed from such an expression is bounded from above as follows

Theorem 2.3. *Let r be mildly simplified, then* $|A(r)| \leq 4\frac{2}{5}\,\mathrm{alph}(r) + 1$. *This bound is tight for an infinite family of regular languages.*

Finally, we show that for some regular languages, the number of operators makes up for two thirds of even the shortest equivalent expression's size.

Theorem 2.4. *There are regular languages L_i such that* $\mathrm{alph}(L_i) \leq n$ *and* $\mathrm{arpn}(L_i) \geq 3n - 1$.

References

[1] A. Brüggemann-Klein. Regular Expressions into Finite Automata. *Theoretical Computer Science*, 120(2):197–213, 1993.

[2] K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005.

[3] S. Gulan and H. Fernau. An Optimal Construction of Finite Automata from Regular Expressions. In: *FSTTCS 08*, pp. 211–222, Dagstuhl Seminar Proceedings 08004, 2008.

[4] L. Ilie and S. Yu. Follow automata. *Information and Computation* 186(1):140–162, 2003.

On Measuring Non-Recursive Trade-Offs

Hermann Gruber, Markus Holzer, and Martin Kutrib
Institut für Informatik, Universität Giessen,
Arndtstraße 2, 35392 Giessen, Germany
E-Mail: {gruber,holzer,kutrib}@informatik.uni-giessen.de

In computer science in general, and also in the particular field of descriptional complexity, we try to classify problems and mechanisms according to different aspects of their tractability. Often the first distinction we make in such a classification is to check whether a problem admits an effective solution at all. If so, we usually take a closer look and analyze the inherent complexity of the problem. But undecidable problems can also be compared to each other, using the toolkit provided by computability theory. Here, it turns out that most naturally occurring problems are complete at some level of the arithmetic (or analytic) hierarchy. This has been a rather successful approach to understand the nature of many undecidable problems we encounter in various computational settings. As for decision problems, there are conversion problems between different models that cannot be solved effectively. Indeed, they evade solvability *a forteriori* because the size blow-up caused by such a conversion cannot be bounded above by any recursive function. This phenomenon, nowadays known as *non-recursive trade-off*, was first observed by Meyer and Fischer [12] between nondeterministic pushdown automata and finite automata. Previously, it had been known that every deterministic pushdown automaton accepting a regular language can be converted into an equivalent finite automaton of at most triply-exponential size. In contrast, Meyer and Fischer showed that if we replace "deterministic pushdown automaton" with "nondeterministic pushdown automaton," then the maximum size blow-up can no longer be bounded by any recursive function. Since that time there has been a steadily growing list of results where this phenomenon has been observed, e.g., [1, 3, 4, 5, 6, 7, 8, 10, 11, 14, 15, 16]. In [9] a survey is given that also presents a few general proof techniques for proving such results. While it seems to be clear that non-recursive trade-offs usually sprout at the wayside of the crossroads of (un)decidability, in many cases proving such trade-offs apparently requires ingenuity and careful automata constructions. While apparently we cannot get rid of this altogether, here we identify general criteria where non-recursive trade-offs can be directly read off, provided certain basic (un)decidability results about the descriptional systems under consideration are known. The present work aims at making the first steps in paralleling the successful development of the abstract theory of languages, and in building a theory with unified proofs of many non-recursive trade-off results appearing in the literature. We can show the following two results, which allows us to deduce that certain

trade-offs between descriptional systems[1] are non-recursive[2] in a very easy way.

Theorem 1 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems that are effective full trios. If the infiniteness problem for $\mathcal{S}_1$ is not semi-decidable and the infiniteness problem for $\mathcal{S}_2$ is decidable, then the trade-off between $\mathcal{S}_1$ and $\mathcal{S}_2$ is non-recursive.*

Here a descriptional system is called an *effective trio*, if it is effectively closed under λ-free morphism, inverse morphism and intersection with regular languages. If it is also effectively closed under general morphism, we speak of an *effective full trio*.

Theorem 2 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems that are effective trios. If $\mathcal{S}_1$ has a decidable word problem but an undecidable emptiness problem, and $\mathcal{S}_2$ has a decidable emptiness problem, then the trade-off between $\mathcal{S}_1$ and $\mathcal{S}_2$ is non-recursive.*

Besides new proof techniques in this domain, the present work also aims to provide a finer classification of such non-recursive trade-offs, in a similar vein to what has been done in the classification of undecidable problems. We prove bounds on the trade-off function f that serves as a least upper bound for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$. Here, it turns out that the complexity[3] of the problem of the $\mathcal{S}_2$-ness of $\mathcal{S}_1$ descriptors influences the growth rate of f.

Theorem 3 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems. The problem of determining for a given descriptor $D_1 \in \mathcal{S}_1$ whether the language $L(D_1)$ belongs to $\mathscr{L}(\mathcal{S}_2)$, i.e., the $\mathcal{S}_2$-ness of $\mathcal{S}_1$*

[1] A *descriptional system* $\mathcal{S}$ is a recursive set of non-empty finite descriptors, such that each descriptor $D \in \mathcal{S}$ describes a formal language $L(D)$, and if $L(D)$ is recursive (recursively enumerable), then there exists an effective procedure to convert D into a Turing machine that decides (semi-decides) $L(D)$. We always assume that a descriptional system is associated with a reasonable size measure. Here a *complexity (size) measure* for $\mathcal{S}$ is a total, recursive function $c : \mathcal{S} \to \mathbb{N}$ such that for any alphabet A, the set of descriptors in $\mathcal{S}$ describing languages over A is recursively enumerable in order of increasing size, and does not contain infinitely many descriptors of the same size.

[2] Let $\mathcal{S}_1$ be a descriptional systems with complexity measure c_1, and $\mathcal{S}_2$ be descriptional systems with complexity measure c_2. A total function $f : \mathbb{N} \to \mathbb{N}$, with $f(n) \geq n$, is said to be an *upper bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for all $D_1 \in \mathcal{S}_1$ with $L(D_1) \in \mathscr{L}(\mathcal{S}_2)$ there exists a $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \leq f(c_1(D_1))$. If there is no recursive upper bound, the trade-off is said to be *non-recursive*.

[3] In particular we consider the *arithmetic hierarchy* [13], which is defined as follows:

$$\begin{aligned} \Sigma_1 &= \{\, L \mid L \text{ is recursively enumerable} \,\}, \\ \Sigma_{n+1} &= \{\, L \mid L \text{ is recursively enumerable in some } A \in \Sigma_n \,\}, \end{aligned}$$

for $n \geq 1$. Here, a language L is said to be recursively enumerable in some B if there is a Turing machine with oracle B that semi-decides L. Let Π_n be the complement of Σ_n, i.e., $\Pi_n = \{\, L \mid \overline{L} \text{ is in } \Sigma_n \,\}$. Moreover, let $\Delta_n = \Sigma_n \cap \Pi_n$, for $n \geq 1$. Observe that $\Delta_1 = \Sigma_1 \cap \Pi_1$ is the class of all recursive sets. Completeness and hardness are always meant with respect to many-one reducibilities $\leq_m$, if not otherwise stated. Let K denote the *halting set*, i.e., the set of all encodings of Turing machines that accept their own encoding. For any set A define $A' = K^A$ to be the *jump* or *completion* of A, where K^A is the *A-relativized halting set*, which is the set of all encodings of Turing machines with oracle A that accept their own encoding, and define $A^{(0)} = A$ and $A^{(n+1)} = (A^{(n)})'$, for $n \geq 0$. Furthermore we use $\leq_T$ to refer to Turing reducibility.

descriptors, can be solved in Σ_2*, if* both $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are recursive. In case* at least one *descriptional system is not recursive (but recursively enumerable) the problem can be solved in* Σ_3*.*

This theorem can be utilized to prove an upper bound when changing from one system to another one.

Theorem 4 *Let* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *be two descriptional systems. If both* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are recursive, then there is a total function* $f : \mathbb{N} \to \mathbb{N}$ *that serves as an upper bound for the increase in complexity when changing from a descriptor in* $\mathcal{S}_1$ *to an equivalent descriptor in* $\mathcal{S}_2$*, satisfying* $f \leq_T \emptyset''$*. In case* at least one *descriptional system is not recursive (but recursively enumerable) the function* $f : \mathbb{N} \to \mathbb{N}$ *can be chosen to satisfy* $f \leq_T \emptyset'''$*.*

What about lower bounds on the trade-off function f? In fact, we show that there is a relation between the function f and the equivalence problem between $\mathcal{S}_1$ and $\mathcal{S}_2$ descriptors, in the sense that, whenever the former problem becomes easy, the latter is easy too.

Theorem 5 *Let* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *be two descriptional systems and* $f : \mathbb{N} \to \mathbb{N}$ *a total function that serves as an upper bound for the increase in complexity when changing from a descriptor in* $\mathcal{S}_1$ *to an equivalent descriptor in* $\mathcal{S}_2$*. Then we have: If both descriptional systems are recursive and* $f \leq_T \emptyset'$*, then the* $\mathcal{S}_2$*-ness of* $\mathcal{S}_1$ *descriptors is recursive in* $\emptyset'$*. If at least one descriptional system is not recursive (but recursively enumerable) and* $f \leq_T \emptyset''$*, then the* $\mathcal{S}_2$*-ness of* $\mathcal{S}_1$ *descriptors is recursive in* $\emptyset''$*.*

Thus, we can show that only *two* types of non-recursive trade-offs within the recursively enumerable languages exist! First consider the context-free grammars and the right-linear context-free grammars (or equivalently finite automata) as descriptional systems. Thus, we want to consider the trade-off between context-free languages and regular languages. In [12] it was shown that this trade-off is non-recursive. By Theorem 4, one can choose the upper bound function f such that $f \leq_T \emptyset''$. On the other hand, if $f \leq_T \emptyset'$, then by Theorem 5 we deduce that checking regularity for context-free grammars is recursive in $\emptyset'$ and hence belongs to Δ_2. This is a contradiction, because in [2] this problem is classified to be Σ_2-complete. So, we obtain a non-recursive trade-off somewhere in between $\emptyset''$ and $\emptyset'$, that is, $f \leq_T \emptyset''$ but $f \not\leq_T \emptyset'$.

References

[1] I. Borchardt. Nonrecursive tradeoffs between context-free grammars with different constant ambiguity. Master's thesis, Universität Frankfurt, 1992. (in German).

[2] D. F. Cudia. The degree hierarchy of undecidable problems of formal grammars. *Symposium on Theory of Computing (STOC 1970)*, 10–21, 1970.

[3] J. Goldstine, M. Kappes, Ch. M. R. Kintala, H. Leung, A. Malcher, and D. Wotschke. Descriptional complexity of machines with limited resources. *J. UCS*, 8:193–234, 2002.

[4] J. Hartmanis. On the succinctness of different representations of languages. *SIAM J. Comput.*, 9:114–120, 1980.

[5] J. Hartmanis. On Gödel speed-up and succinctness of language representations. *Theoret. Comput. Sci.*, 26:335–342, 1983.

[6] Ch. Herzog. Pushdown automata with bounded nondeterminism and bounded ambiguity. *Theoret. Comput. Sci.*, 181:141–157, 1997.

[7] C. Kapoutsis. From $k+1$ to k heads the descriptive trade-off is non-recursive. *Descriptional Complexity of Formal Systems (DCFS 2004)*, 213–224, 2004.

[8] M. Kutrib. On the descriptional power of heads, counters, and pebbles. *Theoret. Comput. Sci.*, 330:311–324, 2005.

[9] M. Kutrib. The phenomenon of non-recursive trade-offs. *Int. J. Found. Comput. Sci.*, 16:957–973, 2005.

[10] A. Malcher. Descriptional complexity of cellular automata and decidability questions. *J. Autom., Lang. Comb.*, 7:549–560, 2002.

[11] A. Malcher. On the descriptional complexity of iterative arrays. *IEICE Trans. Inf. Syst.*, E87-D:721–725, 2004.

[12] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. Symposium on Switching and Automata Theory (SWAT 1971), 188–191, 1971.

[13] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[14] E. M. Schmidt and Th. G. Szymanski. Succinctness of descriptions of unambiguous context-free languages. *SIAM J. Comput.*, 6:547–553, 1977.

[15] B. Sunckel. On the descriptional complexity of metalinear CD grammar systems. *Descriptional Complexity of Formal Systems (DCFS 2004)*, 260–273, 2004.

[16] L. G. Valiant. A note on the succinctness of descriptions of deterministic languages. *Inform. Control*, 32:139–145, 1976.

Cyclically Repetition-free Words on Small Alphabets

Tero Harju
Turku Centre for Computer Science (TUCS)
Department of Mathematics
University of Turku, Finland
E-Mail: harju@utu.fi

Dirk Nowotka
Institute for Formal Methods in Computer Science (FMI)
Universität Stuttgart, Germany
E-Mail: nowotka@fmi.uni-stuttgart.de

All sufficiently long binary words contain a square but there are infinite binary words having only the short squares 00, 11 and 0101. Recently it was shown by J. Currie that there exist cyclically square-free words in a ternary alphabet except for lengths 5, 7, 9, 10, 14, and 17. We consider binary words all conjugates of which contain only short squares. We show that the number $c(n)$ of these binary words of length n grows unboundedly. In order for this, we show that there are morphisms that preserve cyclically square-free words in the ternary alphabet.

1 Introduction

We shall consider binary ($w \in \{0,1\}^*$) and ternary ($w \in \{0,1,2\}^*$) words. A word u is a *factor* of a word w if there are words w_1 and w_2 such that $w = w_1uw_2$. In this case, u *occurs* in w. Two words u and v are *conjugates* if $u = xy$ and $v = yx$ for some words x and y. The *conjugate class* of a word w consists of the words that are conjugates of w. For a given lexicographic order on words, the conjugate class of any primitive word has a minimal element that is called *Lyndon word*. A nonempty factor u^2 $(= uu)$ of a word w is a *square* in w. The word w is *square-free* if it has no squares in it. Moreover, w is *cyclically square-free* if its conjugates are square-free.

While each binary word $w \in \{0,1\}^*$ of length at least four contains a square, Entringer, Jackson, and Schatz [EJS74] showed that there exists an infinite word with only 5 different squares. Afterwards Fraenkel and Simpson [FS95] showed that there exists an infinite binary word having only the three squares 00, 11, and 0101. We say that a binary word w is *short-squared* if its squares belong to the set $\{00, 11, 0101\}$ – but we do not allow the square 1010.

Theorem 1 (Fraenkel–Simpson) *For each $n \geq 1$, there exists a short-squared binary word of length n.*

n	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$c(n)$	3	2	2	2	1	0	0	0	3	0	1	0	0	0

n	18	19	20	21	22	23	24	25	26	27	28	29
$c(n)$	0	2	1	0	0	0	3	0	0	0	1	0

n	30	31	32	33	34	35	36
$c(n)$	1	0	0	0	0	0	2

Table 1: Curious sequence of numbers of cyclic short-squared binary words up to interchanging 0 and 1.

A simplified proof of Theorem 1 was given by Rampersad, Shallit, and Wang [RSM05] which was still shortened by the present authors in [HN06]. In this paper we consider cyclic words with short squares. The problem was motivated by the following result due to Currie [Cur02].

Theorem 2 (Currie) *There exists a cyclically square-free ternary word w of length n if and only if $n \notin \{5, 7, 9, 10, 14, 17\}$.*

A word w is *cyclically short-squared* if its conjugates are all short-squared. We shall show that there are unboundedly long cyclically short-squared binary words.

The exception list of lengths for cyclically short-squared binary words is much more extensive than the list for cyclically square-free ternary words given by Currie. Indeed, it is an open problem to characterize the set L_{cyc} of lengths n for which there exists a cyclically short-squared binary word of length n. Also, even for each length $n \in L_{\mathrm{cyc}}$ there seems to be only a small number of solutions as seen from Table 1 where $c(n)$ denotes the number of conjugate classes of cyclically short-squared binary words of length n, i.e., $c(n)$ is the number of cyclically short-squared binary Lyndon words having length n.

Example 3 *Let us consider some examples of cyclically short-squared binary words. We choose the ordering $1 \prec 0$ for the alphabet for our own convenience.*

The Lyndon representatives of length $n = 12$ are the following three words:

$$111001011000\,,$$

$$111000101100\,,$$

$$111000110010\,.$$

The Lyndon representatives of length $n = 24$ are the following words:

$$111001011001110001011000\,,$$

$$111001011100011001011000\,,$$

$$111000110010111000101100\,.$$

There are, however, only two Lyndon representatives of length $n = 36$:

$$111001011001110001100101110001011000\,,$$

$$111001011100010110011100011001011000\,.$$

Despite of Table 1 suggesting a shrinking number of cyclic short-squared binary words when the length grows, we will show

Theorem 4 *The function $c(n)$ is unbounded:*

$$\limsup_{n\to\infty} c(n) = \infty\ .$$

A mapping $\xi\colon X^* \to Y^*$ is called *morphism* if $\xi(uv) = \xi(u)\xi(v)$, and ξ is called *uniform morphism* if additionally we have for some k that $|\xi(a)| = k$ for all $a \in X$.

Consider now a uniform morphism $\xi\colon \{0,1,2\}^* \to \{0,1\}^*$ that takes cyclic square-free ternary words to cyclic short-squared binary words. Such a morphism can be easily found. Let u and v be two different cyclic square-free ternary words of the same length. Then $\xi(u)$ and $\xi(v)$ are two different cyclic short-squared binary words of the same length. Hence, Theorem 4 follows from the next result which we shall show. Let $c_3(n)$ denote the number of cyclically square-free ternary Lyndon words of length n w.r.t. some fixed order.

Theorem 5 *The function $c_3(n)$ is unbounded:*

$$\limsup_{n\to\infty} c_3(n) = \infty\ .$$

We also state the following conjecture.

Conjecture 6 *There exists an integer N such that $c(n) > 0$ for all $n \geq N$.*

References

[Cur02] J. D. Currie. There are ternary circular square-free words of length n for $n \geq 18$. *Electron. J. Combin.*, 9(1):Note 10, 7 pp. (electronic), 2002.

[EJS74] R. C. Entringer, D. E. Jackson, and J. A. Schatz. On nonrepetitive sequences. *J. Combin. Theory, Ser. A*, 16:159–164, 1974.

[FS95] A. S. Fraenkel and J. Simpson. How many squares must a binary sequence contain? *Electronic J. Combin.*, 2(#R2 (electronic)), 1995.

[HN06] T. Harju and D. Nowotka. Binary words with few squares. *Bull. EATCS*, 89:164–166, 2006.

[RSM05] N. Rampersad, J. Shallit, and Wang M.-w. Avoiding large squares in infinite binary words. *Theoret. Comput. Sci.*, 339:19–34, 2005.

Learning Residual Finite-State Automata Using Observation Tables

Anna Kasprzik
FB IV Theoretische Informatik, University of Trier, D-54286 Trier
E-Mail: kasprzik@informatik.uni-trier.de

Keywords: Regular language, NFA, residual language, observation table, grammatical inference

In the area of grammatical inference the problem of how to infer – or "learn" – a description of a formal language (e.g., a grammar or an automaton) algorithmically from given examples or other kinds of information is considered. A range of conceivable learning settings have been formulated and based on those quite an amount of learning algorithms have been developed. One of the best studied classes with respect to this conception of learnability is the class of regular languages.

A significant part of the algorithms that have been developed, of which Angluin's seminal algorithm L^* for regular string languages [1] was one of the first, use the concept of an *observation table*. In an observation table the rows are labeled by elements from some set that can be taken as prefixes, the columns are labeled by elements from some set that can be taken as suffixes, and each cell of the table contains a Boolean value indicating the membership status of the concatenation of the two labeling elements with respect to the language L the learner is trying to infer (if known from the available information). If the table fulfils certain conditions then we can immediately derive a deterministic finite-state automaton (DFA) from it, and if the given information is sufficient then this DFA is equivalent to the minimal DFA $\mathcal{A}_L$ for L. The states of $\mathcal{A}_L$ correspond exactly to the equivalence classes of L under the syntactic congruence relation on which the Myhill-Nerode theorem (see for example [2]) is based. The elements labeling the rows of the table from which $\mathcal{A}_L$ is derived can be seen as representatives for these equivalence classes, and the elements labeling the columns as experiments by which it is possible to prove that two of those representatives belong to different equivalence classes and should thus represent different states of $\mathcal{A}_L$.

Unfortunately there is a price to pay for the uniqueness of the minimal DFA: It can have exponentially more states than a minimal non-deterministic finite-state automaton (NFA) for the same language, so that it seems worth to consider the question if we should not rather try to infer an automaton of the latter kind. Lemay and Terlutte [3] introduce a special case of NFA, so-called residual finite-state automata (RFSAs), where each state represents a residual language

of the language that is recognized by the automaton. A residual language of a language L with respect to some element u is the set of all suffixes v such that $uv \in L$. And fortunately, there is a unique minimal RFSA $\mathcal{R}_L$ for every regular string language L. Lemay and Terlutte present several learning algorithms for RFSAs (see [6, 5, 4]), which, however, all work by adding or deleting states in an initial automaton according to the information contained in a given sample.

We define a new learning algorithm for regular string languages based on the concept of an observation table by first using an arbitrary existing appropriate algorithm (see for example [9]) in order to establish a table for the reversal[1] of the unknown language L that is to be inferred and then showing that it is possible to derive the minimal RFSA $\mathcal{R}_L$ for L itself from this table after some necessary modifications. By thus exploiting the complementarity of prefixes and suffixes, viz. residual languages and equivalence classes under the syntactic congruence relation, we would like to emphasize the universality of the concept of an observation table which can be used as a basis for the formulation of any kind of inference algorithm (i.e., in any kind of learning setting) founded on the Myhill-Nerode theorem without having to resort to a complicated or very specialized underlying mechanism such as an automaton. The ultimate goal is to illuminate the different aspects of the area of Grammatical Inference and the notions used in it from as many angles as possible in order to eventually be able to set it on a better, even more general theoretical foundation.

References

[1] Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2) (1987) 87–106

[2] Hopcroft, J.E., Ullmann, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Longman (1990)

[3] Denis, F., Lemay, A., Terlutte, A.: Residual finite state automata. In: Proceedings of STACS. Volume 2010 of LNCS. Springer (2001) 144–157

[4] Denis, F., Lemay, A., Terlutte, A.: Some classes of regular languages identifiable in the limit from positive data. In: Grammatical Inference: Algorithms and Applications. Volume 2484 of LNCS. Springer (2002) 269–273

[5] Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using rfsa. In: Proceedings of ALT. Volume 2225 of LNCS. Springer (2001) 348–363

[6] Denis, F., Lemay, A., Terlutte, A.: Learning regular languages using non-deterministic finite automata. In: Proceedings of ICGI. Volume 1891 of LNCS. Springer (2000) 39–50

[7] de la Higuera, C.: Grammatical inference. Unpublished manuscript.

[8] Kasprzik, A.: Learning residual finite-state automata using observation tables. Technical Report 09-3, University of Trier (2009)

[9] Kasprzik, A.: Meta-algorithm genmodel: Generalizing over three learning settings using observation tables. Technical Report 09-2, University of Trier (2009)

[1] The *reversal* $\overline{w}$ of a string $w \in \Sigma^*$ for some alphabet of symbols Σ is defined inductively by $\overline{\varepsilon} := \varepsilon$ and $\overline{aw} := \overline{w}a$ for $a \in \Sigma$, $w \in \Sigma^*$. The reversal of a language $X \subseteq \Sigma^*$ is defined as $\overline{X} := \{\overline{w} | w \in X\}$. Note that due to the duality of left and right congruence the reversal of a regular string language is a regular language as well.

A New Equational Description of an Infinite Hierarchy within $\mathbf{DA}$

Manfred Kufleitner Alexander Lauser
Institut für Formale Methoden der Informatik
Universität Stuttgart
Universitätsstr. 38, 70569 Stuttgart, Germany
E-Mail: {kufleitner,lauser}@fmi.uni-stuttgart.de

We discuss different characterizations of an infinite hierarchy within the variety of finite monoids **DA**, among them a novel equational description.

Inspired by the dot-depth hierarchy of star-free languages, Weil and the first author studied the quantifier alternation hierarchy within FO^2 over finite words, i. e., the first-order logic of the linear order using only two different names for variables [KW09]. They defined the fragments FO^2_m which consist of all FO^2-formulas that have at most m quantifier alternations and showed that the class of languages definable by such formulas constitute a variety of languages. Let $\mathbf{FO}^2_m$ be the corresponding variety of monoids.

On their way, they considered two families of finite monoids $\mathbf{R}_m$ and $\mathbf{L}_m$ for $m \geq 1$ and showed (among other results, some of which we recall below) that these families form an infinite hierarchy of varieties, and that their union is **DA** (we refer to the surveys [TT02, DGK08] for various characterizations of **DA**). It turned out that the hierarchy could be defined algebraically using the so-called Mal'cev-product. We assume that the reader is familiar with the basic notations from semigroup theory [Pin86]. The Mal'cev-product is denoted by $\mathbf{W} \circledcirc{m} \mathbf{V}$ for a variety of finite semigroups $\mathbf{W}$ and a variety of finite monoids $\mathbf{V}$. It consists of all finite monoids M, such that there is a relational morphism $\tau\colon M \to N$ for some $N \in \mathbf{V}$ with the property, that for all idempotents e of N the semigroup $\tau^{-1}(e)$ is in $\mathbf{W}$.

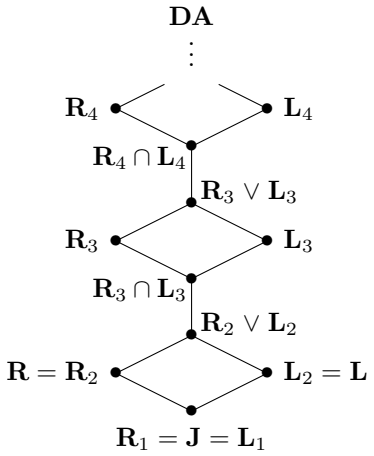


Figure 1: The hierarchy.

We set $\mathbf{R}_1 = \mathbf{L}_1 = \mathbf{J}$, the variety of finite $\mathcal{J}$-trivial monoids, and inductively we define $\mathbf{R}_{m+1} = \mathbf{K} \circledcirc{m} \mathbf{L}_m$ and $\mathbf{L}_{m+1} = \mathbf{D} \circledcirc{m} \mathbf{R}_m$ for $m \geq 1$. Here, $\mathbf{K}$ (resp. $\mathbf{D}$) is the variety defined by the equation $x^\omega y = x^\omega$ (resp. $yx^\omega = x^\omega$).

Another algebraic characterization of the varieties $\mathbf{R}_m$ and $\mathbf{L}_m$ is given as maximal elements of intervals of varieties [KW] which have been studied earlier [TW97]. Combining these two results yields finite defining sets of equations for the two varieties. Here, we give a single equation for $\mathbf{R}_m$ (resp. $\mathbf{L}_m$). As the varieties themselves, the equations are built up inductively. We fix the set of variables $\{x_1, x_2, \ldots\}$. We start by setting $P_2 = (x_2x_1)^\omega$, $Q_2 = P_2x_2$ and $P_{m+1} = (x_{m+1}\overline{P_m})^\omega$, $Q_{m+1} = P_{m+1}x_{m+1}\overline{Q_m}$ for $m \geq 2$, where $\overline{w}$ denotes the reversal of the word w. We are now ready to state our main theorem.

Theorem 1 *For $m \geq 2$ we have $M \in \mathbf{R}_m$ if and only if M satisfies the equation $P_m = Q_m$, and symmetrically, $M \in \mathbf{L}_m$ if and only if M satisfies the equation $\overline{P_m} = \overline{Q_m}$.*

Next, we sketch various characterizations and interesting relations of the hierarchy to other results. It has been shown, that the variety of languages $\mathcal{R}_m$ corresponding to $\mathbf{R}_m$ is equal to

$\mathcal{L}_{m-1}^{det} = \mathcal{L}_{m-1}^{vdet}$, starting with $\mathcal{R}_1$ and $\mathcal{L}_1$ equal to the piecewise-testable languages [KW09]. Here, $\mathcal{L}^{det}$ (resp. $\mathcal{L}^{vdet}$) is a language-theoretic operation, denoting the (visibly) deterministic closure of $\mathcal{L}$. Of course, a dual description of $\mathcal{L}_m$ in terms of (visibly) co-deterministic closure could be given. A major contribution of the paper is an almost tight classification of $\mathbf{FO}_m^2$ within the hierarchy: It is shown that $\mathbf{R}_m \vee \mathbf{L}_m \subseteq \mathbf{FO}_m^2 \subseteq \mathbf{R}_{m+1} \cap \mathbf{L}_{m+1}$ which yields decidability of the alternation depth in FO^2 within one unit.

Furthermore, they draw connections to fragments of the unary temporary logic TL and to certain classes of so-called rankers which are basically sequences of instructions of the form "go to the next a" or "go to the last b". As a combinatorial tool, a generating family of congruences for the variety $\mathbf{R}_m$ (resp. $\mathbf{L}_m$) is introduced.

Complementing this work, Lodaya et al. gave characterizations of the hierarchy in terms of so-called unambiguous fragments of an interval temporal logic [LPS08].

Future work

One natural question one may ask is whether the join of $\mathbf{R}_m$ and $\mathbf{L}_m$ coincides with the meet of $\mathbf{R}_{m+1}$ and $\mathbf{L}_{m+1}$. In fact it can be shown, that $\mathbf{R}_2 \vee \mathbf{L}_2 \subsetneq \mathbf{FO}_2^2 \subseteq \mathbf{R}_3 \cap \mathbf{L}_3$ [KW09]. This result relies on an equational description for $\mathbf{R}_2 \vee \mathbf{L}_2$ due to Almeida and Azevedo [AA89]. For the higher levels of the hierarchy it is not known whether the containment is strict. Another open problem is the question where exactly $\mathbf{FO}_m^2$ is located in the hierarchy. The authors of [KW09] conjecture that $\mathbf{R}_m \vee \mathbf{L}_m$ and $\mathbf{R}_{m+1} \cap \mathbf{L}_{m+1}$ do not coincide for $m \geq 2$ and that moreover $\mathbf{FO}_m^2 = \mathbf{R}_{m+1} \cap \mathbf{L}_{m+1}$. In order to attack these problems, it would be useful to have good algebraical descriptions (in particular equations) for the join and the meet of a given level.

Another line of research would be to consider other structures than finite words and check the statements in these cases. Possible candidates are, e. g., traces or trees; and for all these one can try to generalize to their infinite counterparts.

References

[AA89] Jorge Almeida and Assis Azevedo. The join of the pseudovarieties of $\mathcal{R}$-trivial and $\mathcal{L}$-trivial monoids. *J. Pure Appl. Algebra*, 60(2):129–137, 1989.

[DGK08] Volker Diekert, Paul Gastin, and Manfred Kufleitner. A survey on small fragments of first-order logic over finite words. *IJFCS*, 19(3):513–548, 2008.

[KW] Manfred Kufleitner and Pascal Weil. On the lattice of sub-pseudovarieties of **DA**. To appear.

[KW09] Manfred Kufleitner and Pascal Weil. On FO^2 quantifier alternation over words. In *MFCS 2009, Proceedings*, volume 5734 of *LNCS*, pages 513–524, 2009.

[LPS08] Kamal Lodaya, Paritosh K. Pandya, and Simoni S. Shah. Marking the chops: An unambiguous temporal logic. In *IFIP TCS*, 2008.

[Pin86] Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford Academic, 1986.

[TT02] Pascal Tesson and Denis Thérien. Diamonds are forever: The variety **DA**. In *Semigroups, Algorithms, Automata and Languages*, pages 475–500, 2002.

[TW97] Peter Trotter and Pascal Weil. The lattice of pseudovarieties of idempotent semigroups and a non-regular analogue. *Algebra Universalis*, 37(4):491–526, 1997.

Decidability Questions on Cellular Automata Accepting Bounded Languages[1]

Martin Kutrib
Institut für Informatik
Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
E-Mail: kutrib@informatik.uni-giessen.de

Andreas Malcher
Institut für Informatik
Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
E-Mail: malcher@informatik.uni-giessen.de

Cellular automata are one-dimensional arrays of interconnected interacting finite automata. We investigate one of the weakest classes, the real-time one-way cellular automata, and impose an additional restriction on their inter-cell communication by bounding the number of allowed uses of the links between cells. Moreover, we consider the devices as acceptors for *bounded* languages in order to explore the borderline at which non-trivial decidability problems of cellular automata classes become decidable. It is shown that even devices with drastically reduced communication, that is, each two neighboring cells may communicate only constantly often, accept bounded languages that are not semilinear. If the number of communications is at least logarithmic in the length of the input, several problems are undecidable. The same result is obtained for classes where the total number of communications during a computation is linearly bounded.

[1]Summary of a paper presented at DCFS 2009, Magdeburg (M. Kutrib, A. Malcher: Bounded languages meet cellular automata with sparse communication. In: Jürgen Dassow, Giovanni Pighizzini, Bianca Truthe (Eds.), Descriptional Complexity of Formal Systems (DCFS 2009, Magdeburg), Magdeburg, 2009, 211–222).

Verallgemeinerte Periodizität und Primitivität von Wörtern, deren Wurzeln und Abstände - Fakten und offene Fragen

Gerhard Lischke
Institut für Informatik, Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 1-4, D-07743 Jena, Germany
E-Mail: lischke@minet.uni-jena.de

Zusammenfassung

Die bekannten Begriffe der Periodizität und Primitivität von Wörtern werden verallgemeinert, so daß jeweils sechs verschiedene Abstufungen entstehen. Bekannte und auch neuere Resultate zu den Mengen dieser Wörter werden zusammengestellt und zahlreiche noch offene Fragen herausgearbeitet. So ist z. B. die 1991 von Dömösi, Horváth und Ito aufgestellte Hypothese, daß Q - die Menge aller primitiven Wörter über festem nichttrivialen Alphabet - nicht kontextfrei ist, noch immer unbewiesen. Gleiches gilt auch für die entsprechende Frage hinsichtlich der verallgemeinerten Begriffe. Hinsichtlich ihrer Einordnung in das System kontextueller Sprachen nach Marcus sind die Beziehungen klar: Genau drei der sechs verschiedenen Mengen primitiver Wörter sind externe kontextuelle Sprachen, keine davon aber eine interne kontextuelle Sprache.

Das kürzeste Wort r, so daß $p = r^n$ für eine natürliche Zahl n, ist die Wurzel von p. Hinsichtlich unserer Verallgemeinerungen ergeben sich somit sechs verschiedene Arten von Wurzeln, deren gegenseitige Beziehungen angegeben werden. Offen bleibt aber z. B. noch, ob es ein Wort p gibt, dessen sechs verschiedenartige Wurzeln paarweise verschieden sind, und welches gegebenenfalls das kleinste derartige Wort ist.

Für die Kompliziertheit der Entscheidung jeder der hier eingeführten Mengen ist quadratische Zeit eine untere Schranke. Diese ist auch optimal für die Mengen der primitiven und der stark primitiven Wörter, während dies für die anderen Mengen noch offen ist. Definieren wir die Wurzel einer Sprache L als die Menge der Wurzeln aller Elemente von L, so erhalten wir sechs verschiedene Wurzelmengen von L. Für diese wurde ein Lückensatz bewiesen (Für nahezu beliebige monotone unbeschränkte Funktionen t und f existieren Sprachen L, so daß $L \in$1-TIME(O(t)), aber keine der Wurzelmengen von L ist in TIME(f).), aus dem gefolgert werden kann, daß es reguläre Sprachen gibt, deren Wurzelmengen durchweg nicht einmal kontext-sensitiv sind.

Ausgehend von der in der Codierungstheorie benutzten Hamming-Distanz definieren wir den Abstand zwischen Wörtern und Sprachen und den maximalen Abstand von Wörtern gegebener

Länge und einer Sprache und zeigen, daß der Abstand eines beliebigen Wortes von jeder der Mengen primitiver oder verallgemeinert primitiver Wörter höchstens 1 ist. In umgekehrter Richtung, d.h. von primitiven Wörtern zu den Mengen der in verschiedener Art periodischen Wörter, hängt dieser Maximalabstand von der Kardinalzahl des zugrunde liegenden Alphabets und zahlentheoretischen Eigenschaften der Wortlänge ab. Mit Ausnahme der vier allgemeinsten Fälle periodischer Wörter bei zweibuchstabigem Alphabet kann dieser Abstand für alle anderen Fälle exakt angegeben werden. Für die Ausnahmefälle wird eine bisher unbewiesene Vermutung geäußert. Abschließend untersuchen wir die Mengen möglicher Abstände zwischen den in verschiedener Hinsicht periodi-schen Wörtern, wo es z.Z. noch die meisten offenen Fragen gibt, und geben eine Formel für die genaue Anzahl periodischer Wörter und halbperiodischer Wörter fester Länge an.

On the monadic quantifier alternation hierarchy over texts

Christian Mathissen
Institut für Informatik, Universität Leipzig
04009 Leipzig
E-Mail: mathissen@informatik.uni-leipzig.de

The fundamental result of Büchi and Elgot [Büc60, Elg61] states that the class of regular languages of finite words and the class of languages definable in monadic second-order logic (MSO) coincide. A consequence of its proof is that a language of finite words, considered as labeled linear orders, is definable in MSO iff it is definable in the existential fragment of MSO, that is the quantifier alternation hierarchy collapses. Even more, it does not make a difference if we consider existential MSO over a linear order or a successor relation only.

In contrast, Fagin [Fag75] showed that connectivity separates existential MSO from universal monadic second-order logic for the class of graphs. Furthermore, Matz, Schweikardt and Thomas [MST02] could show that for the class of grids and for the class of graphs the monadic quantifier alternation hierarchy is in fact strict, answering a question of Fagin.

In this workshop contribution we will consider classes of finite structures equipped with two linear orders, so-called texts and the monadic quantifier alternation hierarchy over their successor relations. Texts have been introduced by Ehrenfeucht and Rozenberg [ER90]. A number of authors [EtPR94, HtP96, HtP97] have investigated classes of text languages such as the families of context-free, equational or recognizable text languages and developed a language theory. In particular, the result of Büchi and Elgot on the coincidence of recognizable and MSO-definable languages in MSO has been extended to texts [HtP97]. Following Ehrenfeucht and Rozenberg we will first define an infinite set of operations on texts which arise form their modular decomposition. For each subset of operations we consider the class of texts closed under the latter and containing the one point texts. We show that if the set of operations is finite and contains the two binary operations, then existential MSO is less expressive than the full MSO but the alternation hierarchy collapses at a finite level. In contrast to this we prove that if we permit all operations and hence consider the class of all texts, then the alternation hierarchy is strict.

References

[Büc60] J.R. Büchi. Weak second–order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66 – 92, 1960.

[Elg61] C.C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the AMS*, 98:21–51, 1961.

[ER90] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures. I and II. *Theoretical Computer Science*, 70:277–342, 1990.

[EtPR94] A. Ehrenfeucht, P. ten Pas, and G. Rozenberg. Context-free text grammars. *Acta Informatica*, 31(2):161–206, 1994.

[Fag75] R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[HtP96] H.J. Hoogeboom and P. ten Pas. Text languages in an algebraic framework. *Fundamenta Informaticae*, 25(3):353–380, 1996.

[HtP97] H.J. Hoogeboom and P. ten Pas. Monadic second-order definable text languages. *Theory of Computing Systems*, 30:335–354, 1997.

[MST02] O. Matz, N. Schweikardt, and W. Thomas. The monadic quantifier alternation hierarchy over grids and graphs. *Information and Computation*, 179(2):356–383, 2002.

On the Supports of Recognizable Timed Series

Karin Quaas
Institut für Informatik
Universität Leipzig
04009 Leipzig, Germany
E-Mail: quaas@informatik.uni-leipzig.de

Recently, the model of *weighted timed automata*, introduced by Alur et al. [3] and Behrmann et al. [4], has gained interest within the real-time community. Weighted timed automata are timed automata [2] extended with a weight function assigning weights to both the transitions and the states of the automaton. In this way, they can be used to model continuous consumption of resources. In the last years, a number of decision problems have been investigated, most of them concerning the reachability problem under some optimization aspect [4, 3, 1, 8, 6], model-checking [7, 9], and weighted timed games [10, 7].

In previous works, we built a bridge to the theory of classical weighted automata and introduced a general model of weighted timed automata defined over a semiring and a family of weight functions [11]. In this model, a weighted timed automaton recognizes a *timed series*, i.e., a function mapping to each timed word a weight taken from the semiring. We were able to generalize two of the most fundamental theorems in theoretical computer science, namely the Kleene-Schützenberger Theorem and the Theorem of Büchi and Elgot, to the weighted timed setting [11, 12]. Continuing in this spirit, the aim of the work is to investigate the *support* and *cut languages* of recognizable timed series. The support of a timed series consists of all timed words which are not mapped to zero. Within the theory of weighted automata, supports have been extensively studied (see eg. [5, 13]). Here, we present results concerning supports of recognizable timed series that also lead to the decidability of weighted versions of classical decidability problems as e.g. the emptiness or the universality problem.

We also want to investigate the recognizability of so-called cut languages. These are sets of timed words which are assigned a weight smaller than (or greater than, respectively) a given value. Both supports and cut languages may provide the real-time community with nice applications in the analysis of real-time systems. For instance, we may be interested in whether the set of timed words whose weight under a weighted timed automaton is not exceeding a given value satisfies a specification. Problems like this can be solved using an automata-theoretic approach by using methods presented in this work.

References

[1] Y. Abdeddaïm and O. Maler. Preemptive job-shop scheduling using stopwatch automata. In *TACAS*, volume 2280 of *LNCS*, pages 113–126. Springer, 2002.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *HSCC 2001*, volume 2034 of *LNCS*, pages 49–62. Springer, 2001.

[4] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC 2001*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.

[5] J. Berstel and C. Reutenauer. Rational series and their languages. Current online version of the book of the same name from 1988. February 2007.

[6] P. Bouyer, T. Brihaye, V. Bruyère, and J.-F. Raskin. On the optimal reachability problem on weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, October 2007.

[7] P. Bouyer, T. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Inf. Process. Lett.*, 98(5):188–194, 2006.

[8] P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):3–23, 2008.

[9] P. Bouyer and N. Markey. Costs are expensive! In *FORMATS*, volume 4763 of *LNCS*, pages 53–68. Springer, 2007.

[10] T. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *FORMATS*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.

[11] M. Droste and K. Quaas. A Kleene-Schützenberger Theorem for Weighted Timed Automata. In *FoSSaCS*, volume 4962 of *LNCS*, pages 142–156. Springer, 2008.

[12] K. Quaas. Weighted Timed MSO Logics. In *DLT 2009*, volume 5583 of *LNCS*, pages 419–430. Springer, 2009.

[13] J. Sakarovitch. Rational and recognisable power series. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of Weighted Automata*, chapter 4, pages 103–169. Springer, 2009.

Unterschiede zwischen der Mehrdeutigkeit von nicht-löschenden und löschenden Homomorphismen

Johannes C. Schneider
Fachbereich Informatik, Technische Universität Kaiserslautern,
Postfach 3049, 67653 Kaiserslautern, Germany
E-Mail: jschneider@informatik.uni-kl.de

Wir betrachten in dieser Arbeit Zeichenketten über dem unendlichen Alphabet $\mathbb{N} := \{1, 2, \ldots\}$, im Folgenden *Pattern* genannt, z. B. $\alpha = 1 \cdot 2 \cdot 11 \cdot 42 \cdot 42$, und Homomorphismen, die Pattern auf Wörter über einem Zielalphabet Σ abbilden. Ein solcher Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ heißt *eindeutig* bzgl. eines Patterns $\alpha \in \mathbb{N}^*$, wenn kein Homomorphismus $\tau : \mathbb{N}^* \to \Sigma^*$ existiert mit $\tau(\alpha) = \sigma(\alpha)$ und $\tau(x) \neq \sigma(x)$ für ein Zeichen x in α; ansonsten nennen wir σ *mehrdeutig*.

Die Frage nach der Ein- bzw. Mehrdeutigkeit von Homomorphismen wurde erstmals in [FRS06] untersucht. Dabei bietet sie nicht nur in sich selbst eine interessante Theorie, sondern hat auch direkte Auswirkungen auf die Lernbarkeit von Patternsprachen (s. [Rei08]). Neuere Erkenntnisse in [Sch09] zeigen große Unterschiede zwischen Homomorphismen, die Zeichen löschen dürfen und solchen, die jedes Zeichen auf ein nicht-leeres Wort abbilden. Wir fassen im Folgenden einige der markantesten Unterschiede zwischen der Mehrdeutigkeit von nicht-löschenden und löschenden Homomorphismen zusammen und stellen im Zuge dessen auch neue Resultate vor.

Wir halten zunächst fest, dass die Menge der Pattern, für die ein eindeutiger nicht-löschender Homomorphismus existiert, eine echte Teilmenge der Pattern mit eindeutigem, löschenden Homomorphismus ist.

Unterschied 1 *Sei Σ beliebiges Alphabet. Es existiert ein Pattern α, so dass jeder nicht-löschende Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ mehrdeutig bzgl. α ist, es aber einen eindeutigen, löschenden Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ bzgl. α gibt.*

Auch die Wahl des Zielalphabets hat unterschiedliche Auswirkungen – je nachdem, welche Art von Homomorphismus man betrachtet.

Unterschied 2 *Für alle Alphabete Σ_1, Σ_2 mit mind. zwei Buchstaben gilt: Es existiert genau dann ein eindeutiger, nicht-löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma_1^*$ bzgl. α, wenn ein eindeutiger, nicht-löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma_2^*$ bzgl. α existiert.*
Seien Σ_1, Σ_2 beliebige Alphabete mit $|\Sigma_1| < |\Sigma_2|$. Es existiert ein Pattern α, so dass jeder löschende Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma_1^*$ mehrdeutig bzgl. α ist, jedoch ein eindeutiger, löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma_2^*$ bzgl. α existiert.

Wir können also Pattern mit eindeutigem, löschenden Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ nicht unabhängig von der Größe von Σ charakterisieren.

Auch bezüglich der *moderaten Mehrdeutigkeit* unterscheiden sich nicht-löschende und löschende Homomorphismen. Ein Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ ist moderat mehrdeutig bzgl. eines Patterns $\alpha = i_1 \cdot i_2 \cdot \ldots \cdot i_n$, $i_k \in \mathbb{N},$, wenn für alle Homomorphismen $\tau : \mathbb{N}^* \to \Sigma^*$ mit $\tau(\alpha) = \sigma(\alpha)$ und alle $k = 1, 2, \ldots, n$ gilt, dass $\tau(i_k)$ ein bestimmtes Teilwort von $\sigma(i_k)$ erzeugt, falls $\sigma(i_k) \neq \varepsilon$.

Unterschied 3 *Sei Σ ein beliebiges Alphabet mit mind. zwei Buchstaben. Es existiert genau dann ein moderat mehrdeutiger nicht-löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ bzgl. $\alpha \in \mathbb{N}^*$, wenn ein eindeutiger nicht-löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ bzgl. $\alpha \in \mathbb{N}^*$ existiert.*

Es existiert ein Pattern α, für das kein eindeutiger, löschender Homomorphismus $\sigma : \mathbb{N}^* \to \Sigma^*$ existiert, jedoch ein moderat mehrdeutiger, löschender Homomorphismus $\tau : \mathbb{N}^* \to \Sigma^*$.

Es sei $\mathrm{UNAMB}^{nl}_{\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : \mathbb{N}^* \to \Sigma^*$ ist ein eindeutiger nicht-löschender Hom. bzgl. $\alpha\}$ und $\mathrm{UNAMB}^{l}_{\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : \mathbb{N}^* \to \Sigma^*$ ist ein eindeutiger löschender Hom. bzgl. $\alpha\}$. Damit können wir einen weiteren Unterschied ausmachen:

Unterschied 4 *Sei Σ ein beliebiges Alphabet mit mindestens zwei Buchstaben. Sei α beliebig. Dann ist entweder $|\mathrm{UNAMB}^{nl}_{\Sigma}(\alpha)| = 0$ oder $|\mathrm{UNAMB}^{nl}_{\Sigma}(\alpha)| = \infty$. Jedoch existieren $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{N}^*$ mit $|\mathrm{UNAMB}^{l}_{\Sigma}(\alpha_1)| = 0$, $|\mathrm{UNAMB}^{l}_{\Sigma}(\alpha_2)| = m \in \mathbb{N}$ und $|\mathrm{UNAMB}^{l}_{\Sigma}(\alpha_3)| = \infty$.*

Man beachte, dass zu jedem Alphabet Σ solche Pattern α_1, α_2, α_3 existieren. Nicht zuletzt wegen der Abhängigkeit von Σ stellt die Untersuchung der Mehrdeutigkeit von löschenden Homomorphismen $\sigma : \mathbb{N}^* \to \Sigma^*$ eine reizvolle, vielfältige Aufgabe dar, zumal es bisher keine Charakterisierung der Pattern mit eindeutigem Homomorphismus (über einem bestimmten Zielalphabet) gibt.

Literatur

[FRS06] D.D. Freydenberger, D. Reidenbach und J.C. Schneider. Unambiguous morphic images of strings. *International Journal of Foundations of Computer Science*, 17:601–628, 2006.

[Rei08] D. Reidenbach. Discontinuities in pattern inference. *Theoretical Computer Science*, 397:166–193, 2008.

[Sch09] J.C. Schneider. Unambiguous erasing morphisms in free monoids. In *Proc. SOFSEM 2009: Theorie and Practice of Computer Science*, Band 5404 der *Lecture Notes in Computer Science*, Seiten 473–484, 2009.

Ziel basierte akzeptierende Netzwerke evolutionärer Prozessoren mit regulären Filtern

Bianca Truthe
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
39106 Magdeburg
E-Mail: bianca.truthe@ovgu.de

In dieser Arbeit wird eine neue Variante akzeptierender Netzwerke – Ziel basierte akzeptierende Netzwerke genannt – eingeführt. In einem Ziel basierten akzeptierenden Netzwerk evolutionärer Prozessoren ist jeder Knoten mit einer regulären Sprache – der Zielmenge – ausgestattet. Sobald ein Knoten ein Wort enthält, das zu seiner Zielmenge gehört, wird das Eingabewort vom Netzwerk akzeptiert. Auf diese Weise sind keine weiteren Ausgabeknoten erforderlich. Es wird gezeigt, dass konventionelle akzeptierende Netzwerke mit regulären Filtern und Ziel basierte akzeptierende Netzwerke mit regulären Filtern die gleichen Sprachen akzeptieren. Jedoch kann die Anzahl der Prozessoren, die zum Akzeptieren einer Sprache nötig sind, reduziert werden, wenn man Ziel basierte Netzwerke verwendet.

1 Einleitung

Netzwerke von Sprachprozessoren wurden von E. Csuhaj-Varjú und A. Salomaa eingeführt ([6]). Solch ein Netzwerk kann als Graph angesehen werden, bei dem jeder Knoten Regeln und Wörter hat, die er entsprechend den Regeln ableitet, und die nach dem Passieren gewisser Filter über die Kanten zu anderen Knoten gelangen.

Von Punktmutationen in der Biologie inspiriert, haben J. Castellanos, C. Martin-Vide, V. Mitrana und J. Sempere in [4] Netzwerke evolutionärer Prozessoren eingeführt. Dabei sind die verwendeten Regeln Ersetzen eines Buchstabens durch einen anderen, Einfügen eines Buchstabens und Löschen eines Buchstabens. Ergebnisse zu Netzwerken evolutionärer Prozessoren wurden beispielsweise in [4], [5], [3], [1] veröffentlicht.

Die Erzeugungskraft von Netzwerken mir regulären Filtern und evolutionären Prozessoren, bei denen nur zwei Knoten-Arten vorkommen, wurde in [8] und [2] untersucht.

Akzeptierende Netzwerke evolutionärer Prozessoren mit regulären Filtern wurden von J. Dassow und V. Mitrana in [7] eingeführt. Weitere Ergebnisse zu solchen Netzwerken wurden in [9] veröffentlicht.

In der vorliegenden Arbeit wird eine neue Variante von akzeptierenden Netzwerken (Ziel basierten akzeptierenden Netzwerken) vorgestellt. Es wird gezeigt, dass zu jedem konventionellem akzeptierenden Netzwerk ein Ziel basiertes Netzwerk existiert, das die gleiche Sprache akzeptiert, und dass umgekehrt zu jedem Ziel basiertem akzeptierenden Netzwerk ein konventionelles existiert, das die gleiche Sprache akzeptiert. Damit sind beide Arten von akzeptierenden Netzwerken gleichmächtig. Ziel basierte Netzwerke können jedoch mit weniger Prozessoren auskommen.

Wir zeigen weiterhin, dass jede kontextabhängige Sprache von einem Ziel basierten Netzwerk mit regulären Filtern und genau einem Prozessor (einem Substitutionsprozessor) akzeptiert werden kann. Jede rekursiv aufzählbare Sprache kann von einem Ziel basierten Netzwerk mit genau einem Einfügeprozessor und genau einem Substitutions- oder Löschprozessor akzeptiert werden.

Literatur

[1] A. Alhazov, C. Martín-Vide, Yu. Rogozhin, On the number of nodes in universal networks of evolutionary processors. *Acta Inf.* **43** (2006), 331–339.

[2] A. Alhazov, J. Dassow, C. Martín-Vide, Yu. Rogozhin, B. Truthe, On Networks of Evolutionary Processors with Nodes of Two Types. *Fundamenta Informaticae* **91** (2009) 1, 1–15.

[3] J. Castellanos, P. Leupold, V. Mitrana, On the size complexity of hybrid networks of evolutionary processors. *Theor. Comput. Sci.* **330** (2005), 205–220.

[4] J. Castellanos, C. Martín-Vide, V. Mitrana, J. Sempere, Solving NP-complete problems with networks of evolutionary processors. In: *Proc. IWANN*, Lecture Notes in Computer Science 2084, Springer-Verlag, Berlin, 2001, 621–628.

[5] J. Castellanos, C. Martín-Vide, V. Mitrana, J. Sempere, Networks of evolutionary processors. *Acta Informatica* **38** (2003), 517–529.

[6] E. Csuhaj-Varjú, A. Salomaa, Networks of parallel language processors. In: Gh. Păun, A. Salomaa (eds.), *New Trends in Formal Language Theory*. Lecture Notes in Computer Science 1218, Springer-Verlag, Berlin, 1997, 299–318.

[7] J. Dassow, V. Mitrana, Accepting Networks of Non-Increasing Evolutionary Processors. In: I. Petre, G. Rozenberg (eds.), *Proceedings of NCGT 2008. Workshop on Natural Computing and Graph Transformations. September 8, 2008, Leicester, UK*. University of Leicester, 2008, 29–41.

[8] J. Dassow, B. Truthe, On the Power of Networks of Evolutionary Processors. In: J. Durand-Lose, M. Margenstern (eds.), *Machines, Computations and Universality, MCU 2007, Orléans, France, September 10–13, 2007, Proceedings*. Lecture Notes in Computer Science 4664, Springer, 2007, 158–169.

[9] V. Mitrana, B. Truthe, On Accepting Networks of Evolutionary Processors with at Most Two Types of Nodes. In: A. H. Dediu, A. M. Ionescu, C. Martín-Vide (eds.), *Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2009, Proceedings*. Lecture Notes in Computer Science 5457, Springer, 2009, 588–600.

[10] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[11] B. Truthe, On small accepting networks of evolutionary processors with regular filters. In: J. Dassow, B. Truthe (eds.), *Colloquium on the Occasion of the 50th Birthday of Victor Mitrana. Proceedings*. Otto-von-Guericke-Universität Magdeburg, Germany, 2008, 37–52.

Gewichtete Automaten und Ableitungskomplexität

Johannes Waldmann
Fakultät für Informatik, Mathematik und Naturwissenschaften
Hochschule für Technik, Wirtschaft und Kultur Leipzig
D-04251 Leipzig, Germany
http://www.imn.htwk-leipzig.de/~waldmann/

Die *Ableitungskomplexität* eines terminierenden Ersetzunggsystems ist die maximale Länge von Ableitungen als Funktion der Startgröße. Das *Wachstum* eines gewichteten Automaten ist das maximale Gewicht als Funktion der Eingabegröße. Ein gewichteter Automat heißt *kompatibel* mit einem Ersetzungssystem, falls jeder Ableitungsschritt das Gewicht reduziert. Die Existenz eines kompatiblen Automaten beweist somit Termination des Systems und jede Schranke für das Automatenwachstum ergibt damit eine Schranke für die Ableitungskomplexität. Wir wollen solche Komplexitätszertifikate mit Methoden der Constraint-Programmierung automatisch finden.

Die qualitative Aussage "ein Ersetzungssystem R terminiert" (besitzt keine unendlichen Ableitungen) kann ergänzt werden durch quantitative Angabe der *Ableitungskomplexität* dc_R, die die größtmögliche Länge von Ableitungen als Funktion der Startgröße angibt [HL89]:

$$\mathsf{dc}_R : n \mapsto \sup\{k \mid \exists s, t : |s| \leq n \wedge s \to_R^k t\}.$$

Ein $(\mathbb{N}, +, \cdot)$-gewichteter Automat A mit Zustandsmenge Q, Initialzustand i und Finalzustand f heißt *lokal kompatibel* mit R, falls $\forall a \in \Sigma : A(i, a, i) > 0 \wedge A(f, a, f) > 0$ und für alle Regeln $(l \to r) \in R$ gilt:

$$A(i, l, f) > A(i, r, l) \wedge \forall p, q \in Q : A(p, l, q) \geq A(p, r, q).$$

Das impliziert globale Kompatibilität $A(l) > A(r)$ und damit Termination von R [Wal09]

Es ist klar, daß die Kompatibilität eines (unbekannten) Automaten A fixierter Größe mit einem Ersetzungssystem durch ein Constraint-System beschrieben werden kann, das aus Ungleichungen zwischen Polynomen in den (unbekannten) Gewichten der Transitionen von A besteht.

Der folgende Algorithmus bestimmt eine polynomielle Wachstumsschranke für den $\mathbb{N}$-gewichteten Automaten A, falls diese existiert. Wir benutzen dazu dazu den klassischen (Booleschen) Automaten $\mathsf{supp}(A)$ auf der Zustandsmenge von A, der genau die Transitionen enthält, die in A Gewicht > 0 besitzen.

1. Bestimme die starken Zusammenhangskomponenten $S_1, \ldots, S_k$ von $\mathsf{supp}(A)$.

2. Falls ein S_i eine A-Transition mit Gewicht > 1 enthält, dann wächst A exponentiell.
3. Falls ein S_i mehrdeutig ist, dann wächst A exponentiell. Dazu testet man, ob der Kreuzprodukt-Automat $S_i \times S_i$ erreichbare und produktive Zustände (p, q) mit $p \neq q$ enthält. [Sak03]
4. Ansonsten wächst A polynomiell.

Zur Formulierung dieses Algorithmus als Constraint-System benutzen wir (unbekannte) zweistellige Relationen auf der Zustandsmenge des Automaten mit der Bedeutung:

- $E(p, q) \iff p$ und q liegen in einer gemeinsamen Komponente S_i,
- $R(p, q) \iff$ der Zustand (p, q) aus $A \times A$ ist erreichbar,
- $P(p, q) \iff$ der Zustand (p, q) aus $A \times A$ ist produktiv.

Die resultierende Grad-Schranke ist nicht scharf. Verfeinerungen sind möglich [WS91], aber (als Constraint-System) recht aufwendig.

Das resultierende Verfahren enthält die bisher bekannte Methode der "oberen Dreiecksmatrizen" [MSW08] als Spezialfall, bei dem die S_i alle die Größe 1 haben. Wie dort schon ausgeführt, kann man Schranken für Wortautomaten leicht auf pfad-separierte Baumautomaten übertragen, wie sie in Terminationsbeweisen für Termersetzungssysteme benutzt werden.

Literatur

[HL89] Dieter Hofbauer and Clemens Lautemann. Termination proofs and the length of derivations. In Nachum Dershowitz, editor, *RTA*, volume 355 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 1989.

[MSW08] Georg Moser, Andreas Schnabl, and Johannes Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS*, volume 08004 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[Sak03] Jacques Sakarovitch. *Eléments de théorie des automates*. Vuibert, Paris, 2003.

[Wal09] Johannes Waldmann. Automatic termination. In Ralf Treinen, editor, *RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2009.

[WS91] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.

Position-and-Length-Dependent Context-free Grammars

Frank Weinberg
Fachbereich Informatik
Technische Universität Kaiserslautern
67663 Kaiserslautern, Germany
E-Mail: f_weinbe@cs.uni-kl.de

In our presentation we examine the grammar classes that result when the applicability of a production in a context-free or regular grammar is dependent on the position and/or length of the resulting subword in the complete word generated.

We will consider the inclusion hierarchy of the resulting language classes with respect to each other as well as to the classes of the chomsky hierarchy.

Definitions of the equivalent automatons as well as all the proofs have been omitted due to space restrictions.

1 Definitions

Definition 1 *A* APLCFG *is a 5-tuple:* $G = (V, \Sigma, P, apl, S)$*, where*

- (V, Σ, P, S) *is a context-free grammar and*
- $apl : P \to \wp(\mathbb{N} \times \mathbb{N})$ *is a function associating a set of* allowed positions and lengths *to each rule.*

A PLCFG *is called a*

- P+LCFG *if* $\forall p \in P : apl(p) = (S_{p,1} \subseteq \mathbb{N}) \times (S_{p,2} \subseteq \mathbb{N})$,
- PCFG *if* $\forall p \in P : apl(p) = (S_p \subseteq \mathbb{N}) \times \mathbb{N}$,
- LCFG *if* $\forall p \in P : apl(p) = \mathbb{N} \times (S_p \subseteq \mathbb{N})$.

$\{\text{PL}, \text{P+L}, \text{P}, \text{L}\}\{\text{RLING}, \text{LLING}\}$ *are defined analoguous for right resp. left linear grammars.*

A sequence $S = \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n = w \in \Sigma^*$*. is a derivation of* w *in* G*, if*

- $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ *is a derivation of* w *in* (V, Σ, P, S) *and*
- $\forall 1 \leq i < n$*: if* $\alpha_i = \gamma_1 A \gamma_2$ *and* $\alpha_{i+1} = \gamma_1 \beta \gamma_2$*, then* $(gen(\gamma_1), gen(\beta)) \in apl(A \to \beta)$*, where* $gen(\alpha)$ *is the substring of* w *that is generated from* α*.*

2 Results

Theorem 1 *The inclusion hierarchy of the language classes considered in this paper is as depicted in Figure 1. A line or path from Class* C *to class* C' *indicates* $C \subsetneq C'$*, while no path between* C *and* C' *indicates that neither is a subset of the other.*

Especially note that the newly introduced left linear grammars no longer coincide with their right linear counterparts.

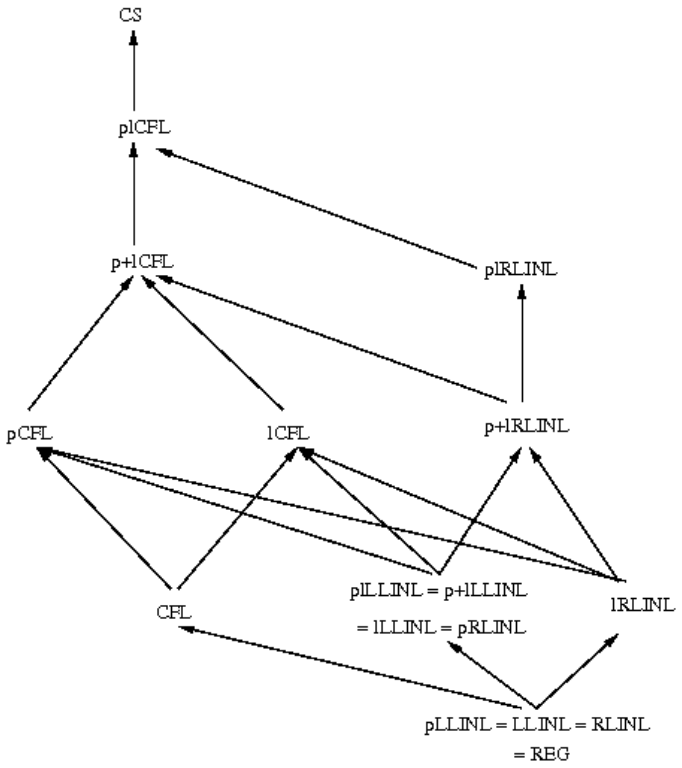


Figure 1: Inclusion hierarchy of the languages considered in this presentation

Theorem 2 *With the exception of* pLLING, *the following properties are not uniformly decidable for G_1 and G_2 from any one of the grammar classes introduced in this presentation.*

1. $L(G_1) = \emptyset$?
2. $L(G_1) = \Sigma^*$?
3. $L(G_1) = L(G_2)$?
4. $L(G_1) \subseteq L(G_2)$?
5. $L(G_1) \cap L(G_2) = \emptyset$?

Theorem 3 *The closure properties depicted in the following table hold:*

	pLCFL	P+LCFL	pCFL	LCFL	CFL	pLRLINL	P+LRLINL	pRLINL	LRLINL	RLINL
complement	-	-	-	-	-	✓	✓	✓	✓	✓
intersection	-	-	-	-	-	✓	✓	✓	✓	✓
union	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
concatenation	-	-	-	✓	✓	-	-	-	-	✓
Kleene star	-	-	-	✓	✓	-	-	-	-	✓
homomorphism	-	-	-	-	✓	-	-	-	-	✓
substitution	-	-	-	-	✓	-	-	-	-	✓

Autorenindex

Liste der Teilnehmer

Bordihn, Henning	Uni Potsdam
Dassow, Jürgen	Uni Magdeburg
Doll, Jens	Context IT GmbH Ahrensburg
Fernau, Henning	Uni Trier
Fichtner, Ina	Uni Leipzig
Freund, Rudolf	TU Wien
Freydenberger, Dominik	Uni Frankfurt
Gebhardt, Andreas	Uni Halle-Wittenberg
Glaßer, Christian	Uni Würzburg
Goehring, Marcel	Uni Kassel
Göller, Stefan	Uni Bremen
Gulan, Stefan	Uni Trier
Hempel, Harald	Uni Jena
Holzer, Markus	Uni Gießen
Hundeshagen, Norbert	Uni Kassel
Kasprzik, Anna	Uni Trier
Kogler, Marian	TU Wien
Kutrib, Martin	Uni Gießen
Lauser, Alexander	Uni Stuttgart
Lischke, Gerhard	Uni Jena
Lohrey, Markus	Uni Leipzig
Malcher, Andreas	Uni Gießen
Manea, Florin	Uni Magdeburg
Mathissen, Christian	Uni Leipzig
Mielke, Jöran	Uni Halle-Wittenberg
Nowotka, Dirk	Uni Stuttgart
Otto, Friedrich	Uni Kassel
Polley, Ronny	Uni Halle-Wittenberg
Quaas, Karin	Uni Leipzig

Reidenbach, Daniel	Loughborough University
Schneider, Johannes	TU Kaiserslautern
Schwarz, Sibylle	FH Zwickau
Staiger, Ludwig	Uni Halle-Wittenberg
Torán, Jacobo	Uni Ulm
Truthe, Bianca	Uni Magdeburg
Wagner, Klaus	Uni Würzburg
Waldmann, Johannes	HTWK Leipzig
Weinberg, Frank	TU Kaiserslautern
Wendlandt, Matthias	Uni Gießen
Winter, Renate	Uni Halle-Wittenberg

Information about the Author(s)

Jöran Mielke, Ludwig Staiger, Renate Winter (Eds.)
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
D-06099 Halle, Germany

E-Mail: {joeran.mielke,ludwig.staiger,renate.winter}@informatik.uni-halle.de
WWW: www.informatik.uni-halle.de