Theorietage 2018

76. Workshop über Algorithmen und Komplexität

28. Theorietag über Automaten und Formale Sprachen

Proceedings

Lutherstadt Wittenberg
24.-27.9.2018

Klaus Reinhardt – Ludwig Staiger – Renate Winter
(Hrsg.)

Technical Report No. 18-1 September 2018

Vorwort

Theorietage haben eine lange Tradition bei verschiedenen Fachgruppen der Gesellschaft für Informatik (GI). Die seit 1991 jährlich stattfindenden Theorietage zu Automaten und Formalen Sprachen wurden bereits in den Jahren 2002 und 2009 in Wittenberg organisiert.

Im Jahr 2015 fand der der 70. Workshop über Algorithmen und Komplexität in Speyer zusammen mit dem 25. Theorietag Automaten und Formale Sprachen statt. Dieses Ereignis zum Vorbild nehmend, finden in diesem Jahr beide Tagungen erneut gemeinsam statt. Bereits zum dritten Mal wurde die Leucorea, eine Stiftung des öffentlichen Rechts an der Martin-Luther-Universität Halle-Wittenberg, als Veranstaltungsort in der Lutherstadt Wittenberg gewählt.

Es folgten 43 Teilnehmer aus Deutschland, Österreich, Polen, Russland und Süd Afrika der Einladung. Das wissenschaftliche Programm enthält 4 eingeladene Vorträge, 11 Vorträge zum Workshop über Algorithmen und Komplexität und 16 Vorträge zum Theorietag über Automaten und Formale Sprachen. Die von von den Fachgruppen Algorithmen und Komplexität eingeldenen Vortragen sind:

Henning Fernau und *Wojciech Plandowski.*

Volker Diekert und *Mikhail Volkov*

sind die von der Fachgruppe Automaten und Formalen Sprachen eingeladenen Vortragenden.

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

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

K. Reinhardt

L. Staiger

R. Winter Halle und Wittenberg, im September 2018

Programm des Workshops

Algorithmen und Komplexität

Montag, 24.September 2018, Auditorium Maximum (im Erdgeschoss)

09.25 Uhr: Begrüßung der Teilnehmer zum Workshop

09.30 – 10.00 Uhr: Stefan Walzer:
Practical Constant-Time Retrieval with Polynomially-small Slack

10.00 – 10.30 Uhr: Katrin Casel:
The Minimal Extension of a Partial Solution

10.30 – 11.00 Uhr: Kaffeepause

11.00 – 11.30 Uhr: Niels Grüttemeier:
Triadic Closures with Multiple Relationship Types

11.30 – 12.00 Uhr: Anne-Sophie Himmel:
Listing All Maximal k-Plexes in Temporal Graphs

12.00 – 12.30 Uhr: André Nichterlein:
Exact Algorithms for Finding Well-Connected 2-Clubs in Sparse Real-World Graphs: Theory and Experiments

12.30 – 14.00 Uhr: Mittagspause

14.00 – 14.30 Uhr: Malte Renken:
Finding Vertex Separators on Temporal Unit Interval Graphs

14.30 – 15.00 Uhr: Markus L. Schmid:
Consensus Strings with Small Maximum Distance and Small Distance Sum

15.00 – 15.30 Uhr: Matthias Bentert:
Tree Containment with Soft Polytomies

15.30 – 16.00 Uhr: Kaffeepause

16.00 – 16.30 Uhr: Arne Meier:
Enumeration in Incremental FPT-Time

16.30 – 17.00 Uhr: Holger Spakowski:
The Robustness of LWPP and WPP, with an Application to Graph Reconstruction

17.00 – 17.30 Uhr: Rudolf Freund:
Activation and Blocking of Rules and Graph Control

Programm der eingeladenen Vorträge

Dienstag, 25.September 2018, Auditorium Maximum (im Erdgeschoss)

09.25 Uhr: Begrüßung der Teilnehmer zum Workshop

09.30 – 10.30 Uhr: Henning Fernau:
Komplexitätstheorie bei Formalen Sprachen

10.30 – 11.00 Uhr: Kaffeepause

11.00 – 12.00 Uhr: Wojciech Plandowski:
Word Equations and Compression

12.00 – 13.30 Uhr: Mittagspause

13.30 – 14.30 Uhr: Volker Diekert:
Word Equations in SL(2, Z)

14.30 – 15.30 Uhr: Mikhail Volkov:
Completely Reachable Automata: An Interplay Between Semigroups, Automata, and Trees

15.30 – 16.00 Uhr: Kaffeepause

16.00 – 17.00 Uhr: Open Problem Session, Seminarraum 1.OG

17.15 – 19.00 Uhr: Fachgruppensitzung AFS, Seminarraum 1.OG

Programm des Theorietags

Automaten und Formale Sprachen

Mittwoch, 26.September 2018, Seminarraum 1.OG

09.25 Uhr: Begrüßung der Teilnehmer zum Theorietag

09.30 – 10.00 Uhr: Florin Manea:
The Satisfiability of Word Equations: Decidable and Undecidable Theories

10.00 – 10.30 Uhr: Andreas Malcher:
Iterative Arrays with Bounded Communication

10.30 – 11.00 Uhr: Kaffeepause

11.00 – 11.30 Uhr: Florin Manea: Exact and Approximated Computation of the Locality Number of Words

11.30 – 12.00 Uhr: Erik Paul:
On Ambiguity of Max-Plus Tree Automata

12.00 – 12.30 Uhr: Andreas Maletti:
Composition Hierarchies of Linear Wighted Extended Top-Down Tree Transducers

12.30 – 14.00 Uhr: Mittagspause

14.00 – 14.30 Uhr: Stefan Dück:
Weighted Operator Precedence Languages

14.30 – 15.00 Uhr: Gustav Grabolle:
Multivalued Linear Dynamic Logic

15.00 – 15.30 Uhr: Sven Dziadek:
ω-Pushdown Automata

15.30 – 16.00 Uhr: Kaffeepause

16.00 – 16.30 Uhr: Simon Beier:
Properties and Decidability of Right One-way Jumping Finite Automata

16.30 – 17.00 Uhr: Chris Köcher:
Reachability Questions on Partially Lossy Queue Automata

17.00 – 17.30 Uhr: Stefan Hoffmann:
Überlegungen zur Černý-Vermutung

Donnerstag, 27.September 2018, Seminarraum 1.OG

09.30 – 10.00 Uhr: Jürgen Dassow:
Kuratowski's Closure-Complement Theorem and the Orbit of Closure-Involution Operations

10:00 – 10.30 Uhr: Bianca Truthe:
Networks of Evolutionary Processors with Resources Restricted Filters

10.30 – 11.00 Uhr: Kaffeepause

11.00 – 11.30 Uhr: Johannes Waldmann:
One-Dimensional Tiling Systems and String Rewriting

11.30 – 12.00 Uhr: Rudolf Freund:
Unfair P Systems

12.00 – 12.30 Uhr: Dominikus Heckmann:
Half-Terminal Grammars (HTG): A Formal Two-stage Structured String Derivation and Interpretation System

Inhaltsverzeichnis

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 1–4.

Practical Constant-Time Retrieval with Polynomially-small Slack

Martin Dietzfelbinger[(A)] Stefan Walzer[(B)]

[(A)] martin.dietzfelbinger@tu-ilmenau.de

[(B)] stefan.walzer@tu-ilmenau.de

Zusammenfassung

Let $\mathcal{U}$ be some universe and $E \subseteq \mathcal{U}$ a set of size $m = |E|$, annotated with $f : E \to \{0,1\}$. Our goal is to obtain a data structure $R = \mathsf{construct}(E, f)$ and corresponding algorithm query such that $\mathsf{query}(R, y) = f(y)$ for all $y \in E$. For $y \in \mathcal{U} \setminus E$ the result of $\mathsf{query}(R, y)$ may be arbitrary. R should occupy only $(1+\varepsilon)m$ bits meaning E cannot be stored.

In this paper we present how to achieve constant query time in conjunction with $\varepsilon = O(m^{-\gamma})$ for constant γ, improving on the previous best $\varepsilon = O((\log m)^{-\gamma})$ for constant γ.

Moreover, our techniques are highly practical. In a particular benchmark with $m = 10^7$ we achieve an order-of-magnitude improvement over previous techniques with $\varepsilon = 0.25\%$ instead of $\varepsilon = 3\%$ without significant sacrifices in runtime.

We employ the framework due to [11, 3, 6], where retrieval is connected to solving sparse linear boolean systems. Our technical contribution is to identify a family of sparse linear systems that combine good locality properties with high probabilities of having full rank. We then adapt the LazyGauss technique from [10] and the Method of Four Russians to solve such systems quickly.

1. Introduction

Let $\mathcal{U}$ be some universe and $E \subseteq \mathcal{U}$ an f-annotated set of size $m = |E|$ where $f : E \to \{0,1\}$. Our goal is to obtain a data structure $R = \mathsf{construct}(E, f)$ and corresponding algorithm query such that $\mathsf{query}(R, y) = f(y)$ for all $y \in E$. For $y \in \mathcal{U} \setminus E$ the result of $\mathsf{query}(R, y)$ may be arbitrary. For instance, let $\mathcal{U} = \{\mathtt{a}, \ldots, \mathtt{z}\}^*$ be the universe of strings and $E \subseteq \mathcal{U}$ the set of given names annotated with their gender[1] $f : E \to \{0 = \mathsf{male}, 1 = \mathsf{female}\}$. The query operation must correctly reproduce the correct gender for any given name, but need not be able to distinguish given names from other strings and may return either male or female when presented with a string that is not in E.

Clearly, this problem can be solved with a dictionary data structure: Find a hash function $h : \mathcal{U} \to \{0,1\}^{O(\log(m))}$ that is injective on E and save the hashes $h(y)$ of all female names in a hashtable T. Then we can define $\mathsf{query}(R = (h, T), y) := 1_{h(y) \in T}$. If the fraction of female names in E is $\Theta(1)$ this requires $O(m \log m)$ bits of memory. However, in this paper the goal is $m + \varepsilon m$ bits for a *slack* $\varepsilon = o(1)$.

[1] This example is due to [9].

All retrieval data structures from the literature fit a general framework due to [11, 3, 6]: Pick a random hash function $h : \mathcal{U} \to S$, where $S \subseteq \{0,1\}^n$ is a set of (typically sparse) vectors and $n = (1+\varepsilon)m$. Then solve the linear system $(\langle h(y), \vec{x}\rangle = f(y))_{y \in E}$ for $\vec{x} \in \{0,1\}$. Here, $\langle \cdot, \cdot \rangle$ denotes the scalar product in the vector space $\mathbb{F}_2^n$. If successful, $R = (h, \vec{x})$ forms the retrieval data structure with $\mathsf{query}(R, y) := \langle h(y), \vec{x}\rangle$. Simple pseudo-code is given in Figure 2 and an example in Figure 1. Neglecting the space to save h, the data structure R occupies $(1+\varepsilon)m$ bits.

Input	Hash Values
$(\texttt{Ana}, 1)$	$h(\texttt{Ana}) = (1,3,9)$
$(\texttt{Bea}, 1)$	$h(\texttt{Bea}) = (2,3,4)$
$(\texttt{Cal}, 0)$	$h(\texttt{Cal}) = (3,6,8)$
$(\texttt{Dan}, 0)$	$h(\texttt{Dan}) = (5,8,9)$
$(\texttt{Eli}, 0)$	$h(\texttt{Eve}) = (2,8,9)$
$(\texttt{Fen}, 1)$	$h(\texttt{Fen}) = (1,5,6)$

$$\begin{pmatrix} 1&0&1&0&0&0&0&0&1 \\ 0&1&1&1&0&0&0&0&0 \\ 0&0&1&0&0&1&0&1&0 \\ 0&0&0&0&1&0&0&1&1 \\ 0&1&0&0&0&0&0&1&1 \\ 1&0&0&0&1&1&0&0&0 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Figure 1: The input consists of $m = 6$ names annotated with 1 for "female" and 0 for "male". Each name is mapped via h to 3 numbers that are interpreted as an $n = 9$ bit vector with 1s in the corresponding positions. The resulting linear system has a solution $\vec{x} = (1,0,0,1,0,0,0,0,0)^T$ and $R = (h, \vec{x})$ forms a retrieval data structure for the input.

Algorithm $\mathsf{construct}(E \subseteq \mathcal{U}, f : E \to \{0,1\})$**:**
- pick $h : E \to S \subseteq \{0,1\}^{(1+\varepsilon)|E|}$
- solve $(\langle h(y), \vec{x}\rangle = f(y))_{y \in E}$ for $\vec{x}$
- restart if $\vec{x}$ does not exist
- **return** $R = (h, \vec{x})$

Algorithm $\mathsf{query}(R = (h, \vec{x}), y \in \mathcal{U})$**:**
- **return** $\langle h(y), \vec{x}\rangle$

Figure 2: The general framework for retrieval data structures. The vectors in S are typically sparse. Linear system solver and query algorithm should exploit this.

There are three degrees of freedom in this framework:

(F1) What is the set S of sparse vectors that is the image of h?

(F2) Which solver for the linear system is used?

(F3) How, if at all, do we partition the input to reduce the runtime of the solver?

2. Contribution

Our main contribution is to propose and analyse a new answer to **(F1)** namely using vectors with coefficients within two blocks. More precisely, let $\ell \in \mathbb{N}$ be a *block size* that divides n. Then $v \in S$ is determined by two random block indices $b_1 < b_2 \in \{1, \ldots, n/\ell\}$ and two random patterns $p_1, p_2 \in \{0,1\}^\ell$ as $v = 0^{\ell b_1 - \ell} p_1 0^{\ell(b_2 - b_1) - \ell)} p_2 0^{n - \ell b_2}$.

Concerning **(F2)**, we adapt the LazyGauss approach by [10] to our situation which turns our large sparse linear system into a significantly smaller dense linear system. The remaining system is solved with the Method of Four Russians and broad word programming. Together with a standard answer to **(F3)**, we obtain:

Theorem 2.1 (Main Theorem) *Assume the context of a Word-RAM with word length w and access to fully random hash functions* [2]*. Then, for any* chunk size $C = m^\alpha$ *($0 < \alpha < 1$), there is a retrieval data structure with a slack of $\varepsilon = C^{-\delta} + \frac{\log m}{C} = m^{-\alpha\delta}$ (for some constant δ), construction time $O(\frac{mC^2}{w \log C})$ and query time $O(1)$.*

This is a significant theoretical improvement as previously query cost $O(1)$ required a slack of $\varepsilon = \log(m)^{-\gamma}$ for constant γ and, conversely, a slack of $\varepsilon = m^{-\gamma}$ for constant γ required query cost $O(\log m)$.

Our approach also performs very well in practice as shown in Table 1.

	Slack	Construction [µs/key]	Lookup [ns]
[10] $k = 3$	9%	1.12	210
[10] $k = 4$	3%	1.75	236
⟨this paper⟩	0.24%	2.6	75-125

Table 1: Comparison of our algorithm to the arguably best-so-far results [10]. We achieve much smaller slack with comparable run times.

Literatur

[1] M. AUMÜLLER, M. DIETZFELBINGER, M. RINK, Experimental Variations of a Theoretically Good Retrieval Data Structure. In: *Proc. 17th ESA*. 2009.

[2] G. V. BARD, *The Method of Four Russians*. Springer US, Boston, MA, 2009.

[3] F. C. BOTELHO, *Near-Optimal Space Perfect Hashing Algorithms*. Ph.D. thesis, Federal University of Minas Gerais, 2008.
`http://homepages.dcc.ufmg.br/~fbotelho/en/pub/thesis.pdf`

[4] F. C. BOTELHO, Y. KOHAYAKAWA, N. ZIVIANI, A Practical Minimal Perfect Hashing Method. In: *Proc. 4th WEA*. 2005.

[5] F. C. BOTELHO, R. PAGH, N. ZIVIANI, Simple and Space-Efficient Minimal Perfect Hash Functions. In: *Proc. 10th WADS*. 2007.

[6] F. C. BOTELHO, R. PAGH, N. ZIVIANI, Practical Perfect Hashing in Nearly Optimal Space. *Inf. Syst.* (2013).

[2] For any universe $\mathcal{U}$ and any finite domain D we assume oracle access to fully random functions $(h_i : \mathcal{U} \to D)_{i \in \mathbb{N}}$, meaning we need to only store an index i to describe such a function. This assumption is motivated by the observation that good (pseudo-)randomness is usually not an issue in practice. In our experiments we use MurmurHash.

[7] N. J. CALKIN, Dependent Sets of Constant Weight Binary Vectors. *Combinatorics, Probability and Computing* (1997).
http://journals.cambridge.org/article_S0963548397003040

[8] C. COOPER, On the rank of random matrices. *Random Structures & Algorithms* **16** (2000) 2, 209–232.

[9] M. DIETZFELBINGER, R. PAGH, Succinct Data Structures for Retrieval and Approximate Membership (Extended Abstract). In: *Proc. 35th ICALP (1)*. 2008.

[10] M. GENUZIO, G. OTTAVIANO, S. VIGNA, Fast Scalable Construction of (Minimal Perfect Hash) Functions. In: *Experimental Algorithms - 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings*. 2016, 339–352.
https://doi.org/10.1007/978-3-319-38851-9_23

[11] B. S. MAJEWSKI, N. C. WORMALD, G. HAVAS, Z. J. CZECH, A Family of Perfect Hashing Methods. *Comput. J.* (1996).

[12] E. PORAT, An Optimal Bloom Filter Replacement Based on Matrix Solving. In: *Proc. 4th CSR*. 2009.

[13] D. H. WIEDEMANN, Solving Sparse Linear Equations Over Finite Fields. *IEEE Transactions on Information Theory* (1986).

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 5–5.

The Minimal Extension of a Partial Solution

Katrin Casel[(A)]

[(A)]Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
casel@informatik.uni-trier.de

Abstract

The very general problem of determining the quality of a given partial solution occurs basically in every algorithmic approach which computes solutions in some sense gradually. Pruning search-trees, proving approximation guarantees or the efficiency of enumeration strategies usually requires a suitable way to decide if a partial solution is a reasonable candidate to pursue. Consider for example the classical concept of minimal dominating sets for graphs. The task of finding a maximum cardinality minimal dominating set (or an approximation of it) as well as enumerating all minimal dominating sets naturally leads to solving the following extension problem: Given a graph $G = (V, E)$ and a vertex set $P \subseteq V$, does there exists a *minimal* dominating set S with $P \subseteq S$.

In an attempt to study the nature of such extension tasks, we propose a general, partial-order based framework to express a broad class of what we refer to as *extension problems*. In essence, we consider optimisation problems in NPO with an additionally specified set of presolutions (including the solutions) and a partial order on those. This partial order $\preccurlyeq$ reflects not only the notion of *extension* but also of *minimality* as follows. For a presolution P and a solution S, S *extends* P if $P \preccurlyeq S$. A solution S is *minimal*, if there exists no solution $S' \neq S$ with $S' \preccurlyeq S$. The resulting extension problem is then formally the task to decide for a given presolution P, if there exits a minimal solution S which extends P.

We consider a number of specific problems which can be expressed in this framework. Possibly contradicting intuition, these problems tend to be NP-hard, even for problems where the underlying optimisation problem can be solved in polynomial time. This raises the question of how fixing a presolution causes this increase in difficulty. In this regard, we study the parameterised complexity of extension problems with respect to parameters related to the presolution. We further discuss relaxation of the extension constraint asking only for a solution S which extends some presolution $P' \preccurlyeq P$. Here we do not want just any such presolution P' but we want P' to be as close to P as possible, in the sense that there exits no presolution $P'' \neq P'$ with $P' \preccurlyeq P'' \preccurlyeq P$ which can also be extended. These considerations yield some insight into the difficult aspects of extension problems.

[(A)]Based on joint work with Henning Fernau, Mehdi Khosravian, Jérôme Monnot and Florian Sikora.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 6–6.

Triadic Closures with Multiple Relationship Types

Laurent Bulteau[(A)] Niels Grüttemeier[(B)] Christian Komusiewicz[(B)]
Manuel Sorge[(C)]

[(A)]Laboratoire d'Informatique Gaspard Monge, France
laurent.bulteau@u-pem.fr

[(B)]Philipps-Universität Marburg, Germany
{niegru,komusiewicz}@informatik.uni-marburg.de

[(C)]Warsaw University, Poland
manuel.sorge@gmail.com

Zusammenfassung

In MULTI STRONG TRIADIC CLOSURE we aim to label the edges of an undirected graph $G = (V, E)$ with strong colors $1, \ldots, c$ and a weak color w such that at most k edges are weak and G contains no induced P_3 with two edges of the same strong color class. This problem is a generalisation of the previously studied STRONG TRIADIC CLOSURE, which is the special case where $c = 1$. The different edge colors model different types of relationships between agents in a social network.

This talk summarizes recent results on the classic and parameterized complexity and problem kernelization of MULTI STRONG TRIADIC CLOSURE and further generalizations of the problem. For example we consider the case where for each edge a set of possible strong colors is given.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 7–7.

Listing All Maximal k-Plexes in Temporal Graphs

Matthias Bentert Anne-Sophie Himmel[(A)] Hendrik Molter[(B)]
Marco Morik Rolf Niedermeier René Saitenmacher

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany
{anne-sophie.himmel, matthias.bentert, h.molter, rolf.niedermeier}@tu-berlin.de

{marco.t.morik, r.saitenmacher}@campus.tu-berlin.de

Zusammenfassung

Many real-world networks evolve over time, that is, new contacts appear and old contacts may disappear. They can be modeled as temporal graphs where interactions between vertices (in case of social networks these would represent people) are represented by time-stamped edges. One of the most fundamental problems in (social) network analysis is community detection, and one of the most basic primitives to model a community is a clique. Addressing the problem of finding communities in temporal networks, Viard et al. [TCS 2016] introduced Δ-cliques as a natural temporal version of cliques. Himmel et al. [SNAM 2017] showed how to adapt the well-known Bron-Kerbosch algorithm to enumerate Δ-cliques. We continue this work and improve and extend this algorithm to enumerate temporal k-plexes (notably, cliques are the special case $k = 1$).

We define a Δ-k-plex as a set of vertices with a lifetime, where during the lifetime each vertex has an edge to all but at most $k - 1$ vertices at least once within any consecutive $\Delta + 1$ time steps. We develop a recursive algorithm for enumerating all maximal Δ-k-plexes and perform experiments on real-world social networks that demonstrate the feasibility of our approach. In particular, for the special case of Δ-1-plexes (that is, Δ-cliques), we observe that our algorithm is significantly faster than the previous algorithm by Himmel et al. [SNAM 2017] for enumerating Δ-cliques.

Full Version on Arxiv: https://arxiv.org/abs/1806.10210

[(A)] Supported by the DFG, projects DAMM (NI 369/13) and FPTinP (NI 369/16).

[(B)] Supported by the DFG, project MATE (NI 369/17).

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 8–8.

Exact Algorithms for Finding Well-Connected 2-Clubs in Sparse Real-World Graphs: Theory and Experiments

Christian Komusiewicz[(A)] André Nichterlein[(B)] Rolf Niedermeier[(B)]
Marten Picker[(B)]

[(A)]Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany
komusiewicz@informatik.uni-marburg.de

[(B)]Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{andre.nichterlein,rolf.niedermeier}@tu-berlin.de

Abstract

Finding (maximum-cardinality) "cliquish" subgraphs is a central topic in graph mining and community detection. A popular clique relaxation are 2-clubs: instead of asking for subgraphs of diameter one (these are cliques), one asks for subgraphs of diameter at most two (these are 2-clubs). A drawback of the 2-club model is that it produces hub-and-spoke structures (typically star-like) as maximum-cardinality solutions. Hence, we study 2-clubs with the additional constraint to be well-connected. More specifically, we investigate the algorithmic complexity for three variants of well-connected 2-clubs, all established in the literature: robust, hereditary, and "connected" 2-clubs. Finding these more dense 2-clubs is NP-hard; nevertheless, we develop an exact combinatorial algorithm, extensively using efficient data reduction rules. Besides several theoretical insights we provide a number of empirical results based on an engineered implementation of our exact algorithm. In particular, the algorithm significantly outperforms existing algorithms on almost all (sparse) real-world graphs we considered.

The talk is based on the following paper https://arxiv.org/abs/1807.07516

Parts of this work are based on the last author's master thesis. Work started when all authors were with TU Berlin.

[(A)]CK was partially supported by the DFG, project MAGZ (KO 3669/4-1).

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 9–9.

Finding Vertex Separators on Temporal Unit Interval Graphs

Malte Renken[(A)]

[(A)]TU-Berlin,
10623 Berlin Ernst-Reuter-Platz 7 m.renken@tu-berlin.de

Abstract

Let a number of particles on the real line be given, that may move over time. Assume that two particles can exchange information only if they are within proximity of each other. Our goal is to prevent the spread of information from a given source to a given destination by removal of a minimum number of particles. We investigate this problem by modeling it as a temporal (that is time-varying) graph, obtaining hardness results in general as well as a fixed-parameter-tractable algorithm for appropriately chosen parameters.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 10–13.

Consensus Strings with Small Maximum Distance and Small Distance Sum

Laurent Bulteau[(A)] Markus L. Schmid[(B)]

[(A)]Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454, Marne-la-Vallée, France, laurent.bulteau@u-pem.fr

[(B)]Fachbereich 4 – Abteilung Informatikwissenschaften, Universität Trier, 54286 Trier, Germany, mlschmid@mlschmid.de

Abstract

This is a summary of the results presented in [4], in which the parameterised complexity of consensus string problems (CLOSEST STRING, CLOSEST SUBSTRING, CLOSEST STRING WITH OUTLIERS) is investigated in a more general setting, i. e., with a bound on the maximum Hamming distance *and* a bound on the sum of Hamming distances between solution and input strings.

1. Problem Definition

Let Σ be a finite alphabet, Σ^* be the set of all strings over Σ, including the empty string ε and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For $w \in \Sigma^*$, $|w|$ is the length of w and, for every i, $1 \leq i \leq |w|$, by $w[i]$, we refer to the symbol at position i of w. For every $n \in \mathbb{N} \cup \{0\}$, let $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$ and $\Sigma^{\leq n} = \bigcup_{i=0}^{n} \Sigma^i$. By $\preceq$, we denote the *substring relation* over the set of strings, i. e., for $u, v \in \Sigma^*$, $u \preceq v$ if $v = xuy$, for some $x, y \in \Sigma^*$. We use the concatenation of sets of strings as usually defined, i. e., for $A, B \subseteq \Sigma^*$, $A \cdot B = \{uv \mid u \in A, v \in B\}$.

For strings $u, v \in \Sigma^*$ with $|u| = |v|$, $\mathsf{d_H}(u, v)$ is the *Hamming distance* between u and v. For a multi-set $S = \{u_i \mid 1 \leq i \leq n\} \subseteq \Sigma^\ell$ and a string $v \in \Sigma^\ell$, for some $\ell \in \mathbb{N}$, the *radius of* S (*w. r. t.* v) is defined by $\mathsf{r_H}(v, S) = \max\{\mathsf{d_H}(v, u) \mid u \in S\}$ and the *distance sum of* S (*w. r. t.* v) is defined by $\mathsf{s_H}(v, S) = \sum_{u \in S} \mathsf{d_H}(v, u)$.[1] Next, we state the problem (r, s)-CLOSEST STRING in full detail, from which we then derive the other considered problems:

(r, s)-CLOSEST STRING
Instance: Multi-set $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$, $\ell \in \mathbb{N}$, and integers $d_r, d_s \in \mathbb{N}$.
Question: Is there an $s \in \Sigma^\ell$ with $\mathsf{r_H}(s, S) \leq d_r$ and $\mathsf{s_H}(s, S) \leq d_s$?

For (r, s)-CLOSEST SUBSTRING, we have $S \subseteq \Sigma^{\leq \ell}$ and an additional input integer $m \in \mathbb{N}$, and we ask whether there is a multi-set $S' = \{s'_i \mid s'_i \preceq s_i, 1 \leq i \leq k\} \subseteq \Sigma^m$ with $\mathsf{r_H}(s, S') \leq d_r$ and $\mathsf{s_H}(s, S') \leq d_s$. For (r, s)-CLOSEST STRING WITH OUTLIERS (or (r, s)-CLOSEST STRING-WO

[1]We slightly abuse notation with respect to the subset relation: for a multi-set A and a set B, $A \subseteq B$ means that $A' \subseteq B$, where A' is the set obtained from A by deleting duplicates; for multi-sets A, B, $A \subseteq B$ is defined as usual. Moreover, whenever it is clear from the context that we talk about multi-sets, we also use the term *set*.

for short) we have an additional input integer $t \in \mathbb{N}$, and we ask whether there is a multi-set $S' \subseteq S$ with $|S'| = k - t$ such that $\mathsf{r_H}(s, S') \leq d_r$ and $\mathsf{s_H}(s, S') \leq d_s$. We also call (r,s)-CLOSEST STRING the *general variant* of CLOSEST STRING, while (r)-CLOSEST STRING and (s)-CLOSEST STRING denote the variants, where the only distance bound is d_r or d_s, respectively; we shall also call them the (r)- and (s)-*variant* of CLOSEST STRING. Analogous notation apply to the other consensus string problems. The problem names that are also commonly used in the literature translate into our terminology as follows: CLOSEST STRING = (r)-CLOSEST STRING, CLOSEST SUBSTRING = (r)-CLOSEST SUBSTRING, CONSENSUS PATTERNS = (s)-CLOSEST SUBSTRING and CLOSEST STRING-WO = (r)-CLOSEST STRING-WO.

For any problem K, by $K(p_1, p_2, \ldots)$, we denote the variant of K parameterised by the parameters $p_1, p_2, \ldots$. For unexplained concepts of parameterised complexity, we refer to the textbooks[5, 6, 9].

The consensus string problems have many natural parameters: the number of input strings k, their length ℓ, the radius bound d_r, the distance sum bound d_s, the alphabet size $|\Sigma|$, the substring length m (in case of (r,s)-CLOSEST SUBSTRING), the number of *outliers* t and *inliers* $k - t$ (in case of (r,s)-CLOSEST STRING-WO). The parameterised complexity (w. r. t. these parameters) of the radius as well as the distance sum variant of CLOSEST STRING and CLOSEST SUBSTRING has been settled by a sequence of papers (see [11, 7, 8, 10, 12] and, for a survey, [3]), except (s)-CLOSEST SUBSTRING with respect to parameter ℓ, which is settled here. The parameterised complexity analysis of the radius variant of CLOSEST STRING WITH OUTLIERS has been started more recently in [2] and, to the knowledge of the authors, the distance sum variant has not yet been considered.

The parameterised complexity of the general variants, where we have a bound on both the radius and the distance sum, has not yet been considered in the literature. While there are obvious reductions from the (r)- and (s)-variants to the general variant, these three variants describe, especially in the parameterised setting, rather different problems.

2. Results

Our main result is that the branching algorithm from [11] (extended in [13] and [2]) can be extended (in a non-trivial way) to obtain the following:

Theorem 2.1 (r,s)-CLOSEST STRING-WO$(d_r, t) \in \mathsf{FPT}$.

In addition to this positive result, we can prove the following hardness results:

Theorem 2.2 (s)-CLOSEST STRING-WO$(d_s, \ell, k-t)$ *and* (s)-CLOSEST SUBSTRING(ℓ, m) *is* $\mathsf{W}[1]$*-hard.*

Finally, by a cross-composition from (r)-CLOSEST STRING into (r)-CLOSEST STRING-WO, we can rule out a polynomial kernel for (r,s)-CLOSEST STRING-WO$(d_r, d_s, \ell, (k-t), |\Sigma|)$.

Theorem 2.3 (r,s)-CLOSEST STRING-WO$(d_r, d_s, \ell, (k-t), |\Sigma|)$ *does not admit a polynomial kernel unless* $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$.

These main results, along with some less complicated extensions of known results for the (r)- and the (s)-variants, yields the following bigger picture of the parameterised complexity of consensus string problems (see [4] for details):

Results for (r,s)-CLOSEST STRING:

k	d_r	d_s	$\|\Sigma\|$	ℓ	Result	Note/Ref.
p	–	–	–	–	FPT	[4]
–	**p**	–	–	–	FPT	[4]
–	–	**p**	–	–	FPT	[4]
–	–	–	2	–	NP-hard	from (r)-variant [10]
–	–	–	–	**p**	FPT	[4]

Results for (r,s)-CLOSEST STRING-WO, including (r)- and (s)-variants:

k	t	$\|\Sigma\|$	ℓ	d_r	d_s	$k-t$	Result	Note/Ref.
p	–	–	–	–	–	–	FPT	[4], Open Prob. in [2]
–	0	2	–	–	–	–	NP-hard	even for d_r-var., but P for d_s-var.
–	**p**	–	**p**	–	–	–	FPT	$d_r \leq \ell$
–	**p**	–	–	**p**	–	–	FPT	Thm. 2.1, and [2] for d_r-var.
–	**p**	–	–	–	**p**	–	FPT	[4]
–	**p**	–	–	–	–	**p**	FPT	$k = t + (k-t)$
–	–	**p**	**p**	–	–	–	FPT	trivial
–	–	**p**	–	$\star$	$\star$	$\star$	Open	param. $\|\Sigma\|$ and some of $d_r, d_s, k-t$
–	–	–	**p**	**p**	**p**	**p**	W[1]-hard	even for d_r-var. [2] and d_s-var. (Thm. 2.2)

Results for (s)-CLOSEST SUBSTRING:

ℓ	k	m	d_s	$\|\Sigma\|$	Result	Reference
–	–	**p**	–	**p**	FPT	trivial
p	–	–	–	**p**	FPT	[13]
p	**p**	–	–	–	FPT	[13]
p	–	–	**p**	–	FPT	[13]
–	–	–	**p**	**p**	FPT	[12]
–	**p**	–	–	**2**	W[1]-hard	[8]
–	**p**	**p**	**p**	–	W[1]-hard	[8]
p	–	**p**	–	–	W[1]-hard	Thm. 2.2

Results for (r,s)-CLOSEST SUBSTRING:

ℓ	k	m	d_r	d_s	$\|\Sigma\|$	Result	Reference
–	–	**p**	–	–	**p**	FPT	[4]
p	**p**	–	–	–	–	FPT	[4]
p	–	–	–	**p**	–	FPT	[4]
p	–	–	–	–	**p**	FPT	[4]
p	–	**p**	**p**	–	–	W[1]-hard	[4], Open Prob. in [13]
–	**p**	–	**p**	**p**	**p**	W[1]-hard	[12]
–	**p**	**p**	**p**	**p**	–	W[1]-hard	[8]

The known kernelisation results can be summarised as follows (the statement "no poly. kernel" is under the assumption that coNP $\not\subseteq$ NP/Poly):

Problem	Kernel size	Reference
(r,s)-CLOSEST STRING(k, d_r)	$\mathrm{O}(k^2 d_r \log k)$	follows from [11]
(r,s)-CLOSEST STRING(d_s)	$\mathrm{O}((d_s)^3 \log d_s)$	follows from [11]
(r,s)-CLOSEST SUBSTRING(ℓ, k)	$\mathrm{O}(\ell k)$	trivial
(r,s)-CLOSEST SUBSTRING(ℓ, d_s)	$\mathrm{O}(\ell d_s)$	trivial
(r,s)-CLOSEST STRING$(d_r, \ell, \lvert\Sigma\rvert)$	no poly. kernel	follows from [1]
(r,s)-CLOSEST SUBSTRING$(k, m, d_r, d_s, \lvert\Sigma\rvert)$	no poly. kernel	follows from [1]
(r)-CLOSEST STRING-WO$(d_r, \ell, t, \lvert\Sigma\rvert)$	no poly. kernel	follows from [1]
(r,s)-CLOSEST STRING-WO$(d_r, d_s, \ell, (k-t), \lvert\Sigma\rvert)$	no poly. kernel	[4]

References

[1] M. BASAVARAJU, F. PANOLAN, A. RAI, M. S. RAMANUJAN, S. SAURABH, On the Kernelization Complexity of String Problems. In: *Proc. 20th International Conference on Computing and Combinatorics, COCOON 2014*. LNCS 8591, 2014, 141–153.

[2] C. BOUCHER, B. MA, Closest string with outliers. *BMC Bioinformatics* **12** (2011), S55.

[3] L. BULTEAU, F. HÜFFNER, C. KOMUSIEWICZ, R. NIEDERMEIER, Multivariate Algorithmics for NP-Hard String Problems. *Bulletin of the EATCS* **114** (2014), 31–73.

[4] L. BULTEAU, M. L. SCHMID, Consensus Strings with Small Maximum Distance and Small Distance Sum. In: *Proc. 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool (UK)*. 2018. To appear.

[5] M. CYGAN, F. FOMIN, L. KOWALIK, D. LOKSHTANOV, D. MARX, M. PILIPCZUK, M. PILIPCZUK, S. SAURABH, *Parameterized Algorithms*. Springer, 2015.

[6] R. G. DOWNEY, M. R. FELLOWS, *Fundamentals of Parameterized Complexity*. Texts in Computer Science, Springer, 2013.

[7] P. A. EVANS, A. D. SMITH, H. T. WAREHAM, On the complexity of finding common approximate substrings. *Theoretical Computer Science* **306** (2003), 407–430.

[8] M. R. FELLOWS, J. GRAMM, R. NIEDERMEIER, On The Parameterized Intractability Of Motif Search Problems. *Combinatorica* **26** (2006), 141–167.

[9] J. FLUM, M. GROHE, *Parameterized Complexity Theory*. Springer, 2006.

[10] M. FRANCES, A. LITMAN, On covering problems of codes. *Theory of Computing Systems* **30** (1997), 113–119.

[11] J. GRAMM, R. NIEDERMEIER, P. ROSSMANITH, Fixed-Parameter Algorithms for CLOSEST STRING and Related Problems. *Algorithmica* **37** (2003), 25–42.

[12] D. MARX, Closest Substring Problems with Small Distances. *SIAM Journal on Computing* **38** (2008), 1382–1410.

[13] M. L. SCHMID, Finding Consensus Strings with Small Length Difference between Input and Solution Strings. *ACM Transactions on Computation Theory* **9** (2017) 3, 13:1–13:18.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 14–14.

Tree Containment with Soft Polytomies

Matthias Bentert[(A)] Josef Malík[(B)] Mathias Weller[(C)]

[(A)]Institut für Softwaretechnik und Theoretische Informatik
TU Berlin
matthias.bentert@tu-berlin.de

[(B)]Department of Theoretical Computer Science
Czech Technical University
josef.malik@fit.cvut.cz

[(C)]CNRS, Laboratoire d'Informatique Gaspard-Monge
Université Paris Est
mathias.weller@u-pem.fr

Zusammenfassung

The Tree Containment problem has many important applications in the study of evolutionary history. Given a phylogenetic network N and a phylogenetic tree T whose leaves are labeled by a set of taxa, it asks if N and T are consistent. While the case of binary N and T has received considerable attention, the more practically relevant variant dealing with biological uncertainty has not. Such uncertainty manifests itself as high-degree vertices ("polytomies") that are "jokers" in the sense that they are compatible with any binary resolution of their children. Contrasting the binary case, we show that this problem, called Soft Tree Containment, is NP-hard, even if N is a binary, multi-labeled tree in which each taxon occurs at most thrice. On the other hand, we reduce the case that each label occurs at most twice to solving a 2-SAT instance of size $O(|T|^3)$. This implies NP-hardness and polynomial-time solvability on reticulation-visible networks in which the maximum in-degree is bounded by three and two, respectively.

The full paper is available at http://drops.dagstuhl.de/opus/volltexte/2018/8835/.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 15–15.

Enumeration in Incremental FPT-Time

Arne Meier[(A)]

[(A)]Leibniz Universität Hannover, Institut für Theoretische Informatik,
Appelstrasse 4, 30167 Hannover,
meier@thi.uni-hannover.de

Abstract

In this talk, we study the relationship of parametrised enumeration complexity classes defined by Creignou et al. (MFCS 2013). Specifically, we introduce two hierarchies (IncFPTa and CapIncFPTa) of enumeration complexity classes for incremental fpt-time in terms of exponent slices and show how they interleave. Furthermore, we define several parametrised function classes and, in particular, introduce the parametrised counterpart of the class of nondeterministic multivalued functions with values that are polynomially verifiable and guaranteed to exist, TFNP, known from Megiddo and Papadimitriou (TCS 1991). We show that TF(para-NP) collapsing to F(FPT) is equivalent to OutputFPT coinciding with IncFPT. This result is in turn connected to a collapse in the classical function setting and eventually to the collapse of IncP and OutputP which proves the first direct connection of classical to parametrised enumeration.

[(A)]Funded by the DFG grant ME 4279/1-2

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 16–17.

The Robustness of LWPP and WPP, with an Application to Graph Reconstruction

Edith Hemaspaandra[(A)] Lane A. Hemaspaandra[(B)]
Holger Spakowski[(C)] Osamu Watanabe[(D)]

[(A)]Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623, USA

[(B)]Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

[(C)]Department of Mathematics and Applied Mathematics
University of Cape Town
Rondebosch 7701, South Africa

[(D)]Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology
Tokyo 152-8552, Japan

We show that the counting class LWPP [1] remains unchanged even if one allows a polynomial number of gap values rather than one. On the other hand, we show that it is impossible to improve this from polynomially many gap values to a superpolynomial number of gap values by relativizable proof techniques.

The first of these results implies that the Legitimate Deck Problem (from the study of graph reconstruction) is in LWPP (and thus low for PP, i.e., $\mathrm{PP}^{\text{Legitimate Deck}} = \mathrm{PP}$) if the weakened version of the Reconstruction Conjecture holds in which the number of nonisomorphic preimages is assumed merely to be polynomially bounded. This strengthens the 1992 result of Köbler, Schöning, and Torán [3] that the Legitimate Deck Problem is in LWPP if the Reconstruction Conjecture holds, and provides strengthened evidence that the Legitimate Deck Problem is not NP-hard.

We additionally show on the one hand that our main LWPP robustness result also holds for WPP, and also holds even when one allows both the rejection- and acceptance- gap-value targets to simultaneously be polynomial-sized lists; yet on the other hand, we show that for the #P-based analog of LWPP the behavior much differs in that, in some relativized worlds, even two target values already yield a richer class than one value does. Despite that nonrobustness result for a #P-based class, we show that the #P-based "exact counting" class $\mathrm{C}_{=}\mathrm{P}$ remains

This work was done in part while Edith and Lane A. Hemaspaandra were visiting the Tokyo Institute of Technology's Department of Mathematical and Computing Sciences, in part while Edith and Lane A. Hemaspaandra were visiting ETH Zürich's Department of Computer Science, and in part while Holger Spakowski was visiting the University of Rochester's Department of Computer Science. This work was presented at MFCS 2018 [2].

unchanged even if one allows a polynomial number of target values for the number of accepting paths of the machine.

References

[1] S. FENNER, L. FORTNOW, S. KURTZ, Gap-Definable Counting Classes. *Journal of Computer and System Sciences* **48** (1994) 1, 116–148.

[2] E. HEMASPAANDRA, L. A. HEMASPAANDRA, H. SPAKOWSKI, O. WATANABE, The Robustness of LWPP and WPP, with an Application to Graph Reconstruction. In: I. POTAPOV, P. SPIRAKIS, J. WORRELL (eds.), *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Leibniz International Proceedings in Informatics (LIPIcs) 117, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, 51:1–51:14. `http://drops.dagstuhl.de/opus/volltexte/2018/9633`

[3] J. KÖBLER, U. SCHÖNING, J. TORÁN, Graph Isomorphism is low for PP. *Computational Complexity* **2** (1992), 301–330.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 18–21.

Sequential Grammars with Activation and Blocking of Rules and Sequential Grammars with Graph Control

Rudolf Freund

Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Vienna, Austria
rudi@emcc.at

Abstract

We introduce the control mechanism of activation and blocking of rules for a general model of sequential grammars and compare it with graph control.

1. Control Mechanisms

In this section we recall the notions and basic results for the general model of sequential grammars equipped with specific control mechanisms as elaborated in [3], based on the applicability of rules, as well as for the new concept of activation and blocking of rules as exhibited in [1] and [2].

1.1. A General Model for Sequential Grammars

A *(sequential) grammar* G_s is a construct $(O, O_T, w, P, \Longrightarrow_{G_s})$ where

- O is a set of *objects*;
- $O_T \subseteq O$ is a set of *terminal objects*;
- $w \in O$ is the *axiom (start object)*;
- P is a finite set of *rules*;
- $\Longrightarrow_{G_s} \subseteq O \times O$ is the *derivation relation* of G_s.
  Each of the rules $p \in P$ induces a relation $\Longrightarrow_p \subseteq O \times O$ with respect to $\Longrightarrow_{G_s}$. A rule $p \in P$ is called *applicable* to an object $x \in O$ if and only if there exists at least one object $y \in O$ such that $(x, y) \in \Longrightarrow_p$; we also write $x \Longrightarrow_p y$. The derivation relation $\Longrightarrow_{G_s}$ is the union of all $\Longrightarrow_p$, i.e., $\Longrightarrow_{G_s} = \cup_{p \in P} \Longrightarrow_p$. The reflexive and transitive closure of $\Longrightarrow_{G_s}$ is denoted by $\overset{*}{\Longrightarrow}_{G_s}$.

Specific conditions on the rules in P define a special type X of grammars which then will be called *grammars of type* X.

The *language generated by* G is the set of all terminal objects that can be derived from the axiom, i.e.,

$$L(G_s) = \left\{ v \in O_T \mid w \overset{*}{\Longrightarrow}_{G_s} v \right\}.$$

The family of languages generated by grammars of type X is denoted by $\mathcal{L}(X)$.

If for every G_s of type X we have $O_T = O$, then X is called a *pure* type, otherwise it is called *extended*; X is called *strictly extended* if for any grammar G_s of type X, $w \notin O_T$ and for all $x \in O_T$, no rule from P can be applied to x.

A type X of grammars is called a *type with unit rules* if for every grammar of type X $G_s = (O, O_T, w, P, \Longrightarrow_G)$ there exists a grammar $G'_s = \left(O, O_T, w, P \cup P^{(+)}, \Longrightarrow_{G'_s}\right)$ of type X such that $\Longrightarrow_{G_s} \subseteq \Longrightarrow_{G'_s}$ and

- $P^{(+)} = \left\{p^{(+)} \mid p \in P\right\}$,
- for all $x \in O$, $p^{(+)}$ is applicable to x if and only if p is applicable to x, and
- for all $x \in O$, if $p^{(+)}$ is applicable to x, the application of $p^{(+)}$ to x yields x back again.

A type X of grammars is called a *type with trap rules* if for every grammar of type X $G_s = (O, O_T, w, P, \Longrightarrow_G)$ there exists a grammar $G'_s = \left(O, O_T, w, P \cup P^{(-)}, \Longrightarrow_{G'_s}\right)$ of type X such that $\Longrightarrow_{G_s} \subseteq \Longrightarrow_{G'_s}$ and

- $P^{(-)} = \left\{p^{(-)} \mid p \in P\right\}$,
- for all $x \in O$, $p^{(-)}$ is applicable to x if and only if p is applicable to x, and
- for all $x \in O$, if $p^{(-)}$ is applicable to x, the application of $p^{(-)}$ to x yields an object y from which no terminal object can be derived anymore.

1.2. Graph-controlled and Programmed Grammars

A *graph-controlled grammar* (with applicability checking) of type X is a construct

$$G_{GC} = \left(G_s, g, H_i, H_f, \Longrightarrow_{GC}\right)$$

where $G_s = (O, O_T, w, P, \Longrightarrow_G)$ is a grammar of type X; $g = (H, E, K)$ is a labeled graph where H is the set of node labels identifying the nodes of the graph in a one-to-one manner, $E \subseteq H \times \{Y, N\} \times H$ is the set of edges labeled by Y or N, $K : H \to 2^P$ is a function assigning a subset of P to each node of g; $H_i \subseteq H$ is the set of initial labels, and $H_f \subseteq H$ is the set of final labels. The derivation relation $\Longrightarrow_{GC}$ is defined based on $\Longrightarrow_{G_s}$ and the control graph g as follows: For any $i, j \in H$ and any $u, v \in O$, $(u, i) \Longrightarrow_{GC} (v, j)$ if and only if

- $u \Longrightarrow_p v$ by some rule $p \in K(i)$ and $(i, Y, j) \in E$ *(success case)*, **or**
- $u = v$, no $p \in K(i)$ is applicable to u, and $(i, N, j) \in E$ *(failure case)*.

The language generated by G_{GC} is defined by

$$L(G_{GC}) = \left\{v \in O_T \mid (w, i) \Longrightarrow^*_{G_{GC}} (v, j),\ i \in H_i, j \in H_f\right\}.$$

If $H_i = H_f = H$, then G_{GC} is called a *programmed grammar*. The families of languages generated by graph-controlled and programmed grammars of type X are denoted by $\mathcal{L}(X\text{-}GC_{ac})$ and $\mathcal{L}(X\text{-}P_{ac})$, respectively. If the set E contains no edges of the form (i, N, j), then the graph-controlled grammar is said to be *without applicability checking*; the corresponding families of languages are denoted by $\mathcal{L}(X\text{-}GC)$ and $\mathcal{L}(X\text{-}P)$, respectively.

As a special variant of graph-controlled grammars we consider those where all labels are final; the corresponding family of languages generated by graph-controlled grammars of type X is abbreviated by $\mathcal{L}\left(X\text{-}GC_{ac}^{allfinal}\right)$. By definition, programmed grammars are just a subvariant where in addition all labels are also initial.

1.3. Grammars with Activation and Blocking of Rules

A *grammar with activation and blocking of rules* (an *AB-grammar*) of type X is a construct

$$G_{AB} = \big(G_s, L, f_L, A, B, L_0, \Longrightarrow_{G_{AB}}\big)$$

where $G_s = (O, O_T, w, P, \Longrightarrow_G)$ is a grammar of type X, L is a finite set of labels with each label having assigned one rule from P by the function f_L, A, B are finite subsets of $L \times L \times \mathbb{N}$, and L_0 is a finite set of tuples of the form $\big(q, Q, \bar{Q}\big)$, $q \in L$, with the elements of $Q, \bar{Q}$ being of the form (l, t), where $l \in L$ and $t \in \mathbb{N}$, $t > 1$.

A derivation in G_{AB} starts with one element $\big(q, Q, \bar{Q}\big)$ from L_0 which means that the rule labeled by q has to be applied to the initial object w in the first step and for the following derivation steps the conditions given by Q as activations of rules and $\bar{Q}$ as blockings of rules have to be taken into account in addition to the activations and blockings coming along with the application of the rule labeled by q. The role of L_0 is to get a derivation started by activating some rule for the first step(s) although no rule has been applied so far, but probably also providing additional activations and blockings for further derivation steps.

A configuration of G_{AB} in general can be described by the object derived so far and the activations Q and blockings $\bar{Q}$ for the next steps. In that sense, the starting tuple $\big(q, Q, \bar{Q}\big)$ can be interpreted as $\big(\{(q,1)\} \cup Q, \bar{Q}\big)$, and we may also simply write $\big(Q', \bar{Q}\big)$ with $Q' = \{(q,1)\} \cup Q$. We mostly will assume Q and $\bar{Q}$ to be non-conflicting, i.e., $Q \cap \bar{Q} = \emptyset$; otherwise, we interpret $\big(Q', \bar{Q}\big)$ as $\big(Q' \setminus \bar{Q}, \bar{Q}\big)$.

Given a configuration $\big(u, Q, \bar{Q}\big)$, in one step we can derive $\big(v, R, \bar{R}\big)$, and we also write $\big(u, Q, \bar{Q}\big) \Longrightarrow_{G_{AB}} \big(v, R, \bar{R}\big)$, if and only if

- $u \Longrightarrow_G v$ using the rule r such that $(q,1) \in Q$ and $(q,r) \in f_L$, i.e., we apply the rule labeled by q activated for this next derivation step to u; the new sets of activations and blockings are defined by

$$\begin{aligned} \bar{R} &= \big\{(x,i) \mid (x,i+1) \in \bar{Q},\ i > 0\big\} \cup \{(x,i) \mid (q,x,i) \in B\}, \\ R &= (\{(x,i) \mid (x,i+1) \in Q,\ i > 0\} \cup \{(x,i) \mid (q,x,i) \in A\}) \\ &\quad \setminus\ \big\{(x,i) \mid (x,i) \in \bar{R}\big\} \end{aligned}$$

  (observe that R and $\bar{R}$ are made non-conflicting by eliminating rule labels which are activated and blocked at the same time); **or**
- no rule r is activated to be applied in the next derivation step; in this case we take $v = u$ and continue with $\big(v, R, \bar{R}\big)$ constructed as before provided R is not empty, i.e., there are rules activated in some further derivation steps; otherwise the derivation stops.

The language generated by G_{AB} is defined by

$$L(G_{AB}) = \big\{v \in O_T \mid \big(w, Q, \bar{Q}\big) \Longrightarrow^*_{G_{AB}} \big(v, R, \bar{R}\big) \text{ for some } \big(Q, \bar{Q}\big) \in L_0\big\}.$$

The family of languages generated by AB-grammars of type X is denoted by $\mathcal{L}(X\text{-}AB)$. If the set B of blocking relations is empty, then the grammar is said to be a *grammar with activation of rules* (an *A-grammar* for short) of type X; the corresponding family of languages is denoted by $\mathcal{L}(X\text{-}A)$.

2. AB-Grammars and Graph-controlled Grammars

Theorem 2.1 *For any type X, $\mathcal{L}(X\text{-}AB) \subseteq \mathcal{L}(X\text{-}GC_{ac})$.*

In the case of graph-controlled grammars with all labels being final, for any strictly extended type X with trap rules, we can show an exciting result exhibiting that the power of rule activation is really strong and that the additional power of blocking is not needed.

Theorem 2.2 *For any strictly extended type X with trap rules,*

$$\mathcal{L}\left(X\text{-}GC_{ac}^{allfinal}\right) \subseteq \mathcal{L}(X\text{-}A).$$

As programmed grammars are just a special case of graph-controlled grammars with all labels being final, we immediately infer the following result:

Corollary 2.3 *For any strictly extended type X with trap rules,*

$$\mathcal{L}(X\text{-}P_{ac}) \subseteq \mathcal{L}(X\text{-}A).$$

Combining Theorems 2.1 and 2.2, we infer the following equality:

Corollary 2.4 *For any strictly extended type X with trap rules,*

$$\mathcal{L}\left(X\text{-}GC_{ac}^{allfinal}\right) = \mathcal{L}(X\text{-}A).$$

For example, in the string case the preceding general results yield the following result (where CF denotes the type of context-free string grammars, RE denotes the family of recursively enumerable languages):

Corollary 2.5 $\mathcal{L}(CF\text{-}GC_{ac}) = \mathcal{L}(CF\text{-}P_{ac}) = \mathcal{L}(CF\text{-}A) = RE.$

References

[1] A. ALHAZOV, R. FREUND, S. IVANOV, P systems with activation and blocking of rules. In: S. STEPNEY, S. VERLAN (eds.), *Unconventional Computation and Natural Computation. 17th International Conference, UCNC 2018, Fontainebleau, France, June 25-29, 2018, Proceedings*. Lecture Notes in Computer Science, Springer, 2018, 1–15.

[2] R. FREUND, Control mechanisms for array grammars on Cayley grids. In: J. DURAND-LOSE, S. VERLAN (eds.), *Machines, Computations, and Universality*. Springer, 2018, 1–33.

[3] R. FREUND, M. KOGLER, M. OSWALD, A general framework for regulated rewriting based on the applicability of rules. In: J. KELEMEN, A. KELEMENOVÁ (eds.), *Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday*. Lecture Notes in Computer Science 6610, Springer, 2011, 35–53.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 22–28.

Komplexitätstheorie bei Formalen Sprachen

Henning Fernau[(A)]

[(A)]Universität Trier, Informatikwissenschaften, CIRT
fernau@uni-trier.de

Zusammenfassung

Wir betrachten einige neuere Aspekte dieser alten Beziehung. Wir konzentrieren uns auf Aspekte der parameterisierten Komplexität und auf den Nachweis unterer Schranken.

1. Einleitung

Schon die Betrachtung klassischer Lehrbücher der Theoretischen Informatik wie [15] zeigt, wie eng Komplexitätstheorie und Formale Sprachen zusammenhängen. Wir möchten in diesem Beitrag auf einige modernere Aspekte dieses Zweiklangs aufmerksam machen, fokussierend auf einer mehrparametrischen Analyse der entsprechenden Probleme, die die Ursachen für die Härte von gewissen Berechnungsproblemen aufzeigen möchte. Hierbei setzen wir elementare Kenntnisse aus dem Bereich der parameterisierten Komplexität voraus, siehe [8, 10]. Unter dem *Standardparameter* eines Problems versteht man eine Schranke auf die Größe des Objekts, nach dem gesucht wird. Gerade bei formalsprachlichen Problemen ergibt sich jedoch eine Vielzahl weiterer natürlicher Parameter, wie wir im Folgenden zeigen wollen. Abschließend erörtern wir neuere untere Komplexitätsschranken.

2. Parameter Alphabetgröße

Dieser Parameter mag zunächst verwundern: Wie kann die Härte eines Problems in der Größe des Alphabetes stecken? Es gibt aber Beispiele, bei denen das der Fall ist, und andere, wo das nicht so ist. Wir beschränken uns auf zwei klassische Probleme.

2.1. Editierprobleme

Bei Worteditierproblemen (string-to-string correction, kurz S2S) werden klassischerweise vier Operationen betrachtet:

- **C** Change: Ersetze einen einzelnen Buchstaben durch einen anderen;
- **D** Delete: Lösche einen einzelnen Buchstaben;
- **I** Insert: Füge einen einzelnen Buchstaben ein;
- **S** Swap: Vertausche zwei benachbarte Buchstaben.

Es sei O eine Teilmenge dieser Operationen. O-S2S erhält als Eingaben zwei Wörter M,T und eine Zahl k und fragt, ob es mit höchstens k Operationen aus O möglich ist, S in T zu verwandeln. Wagner [30] hat folgendes schöne Dichotomieergebnis gezeigt:

Satz 2.1 *Gilt $O \notin \{\{\boldsymbol{S},\boldsymbol{D}\},\{\boldsymbol{S},\boldsymbol{I}\}\}$, so ist O-S2S in Polynomzeit lösbar; für $O \in \{\{\boldsymbol{S},\boldsymbol{D}\},\{\boldsymbol{S},\boldsymbol{I}\}\}$ gilt, dass O-S2S NP-schwer ist.*

Die NP-Schwere fußt auf einem Beweis, bei dem die Alphabetgröße unbeschränkt ist. Es blieb 40 Jahre lang offen, ob z.B. für den wichtigen Fall des Binäralphabetes O-S2S für $O \in \{\{\mathbf{S},\mathbf{D}\},\{\mathbf{S},\mathbf{I}\}\}$ doch in Polynomzeit lösbar ist. Dieses wurde abschließend in [17] behandelt:

Satz 2.2 *Für $O \in \{\{\boldsymbol{S},\boldsymbol{D}\},\{\boldsymbol{S},\boldsymbol{I}\}\}$ und jedes feste Alphabet Σ gilt, dass O-S2S-Σ in Polynomzeit lösbar ist.*

Aus der parameterisierten Sicht zeigt der zughörige Beweis, dass O-S2S in XP liegt bei Parameterisierung mit der Alphabetgröße. Die Mitgliedschaft in FPT ist noch ungeklärt. Für die Standardparameterisierung k konnte jedoch Folgendes gezeigt werden [3]:

Satz 2.3 *Für $O \in \{\{\boldsymbol{S},\boldsymbol{D}\},\{\boldsymbol{S},\boldsymbol{I}\}\}$ kann O-S2S in Zeit $O^*(1.62^k)$ gelöst werden.*

Die Probleme, zentrale oder Median-Wörter aufzufinden, kann man als Verallgemeinerungen der Worteditierprobleme begreifen; **S** ist hier keine zulässige Operation, es wird also der Levenshtein-Abstand betrachtet. Hierfür wurde auch zunächst nur für unbeschränkte Alphabete gezeigt, dass diese Probleme NP-schwer sind, später wurde dieses aber auch (sogar) für Binäralphabete nachgewiesen, siehe [9, 19].

2.2. Grammatikbasierte Kompression

In Verallgemeinerung der bekannten Kompressionsverfahren von Lempel und Ziv haben Storer und Szymanski vorgeschlagen (siehe [23, 26]), bei vorgelegtem Wort w eine möglichst kleine kontextfreie Grammatik G anzugeben mit $L(G) = \{w\}$. Hierbei wird die Größe einer Grammatik als Summe der Längen rechter Regelseiten gemessen. Für das zugehörige Entscheidungsproblem SGP konnte man [7, 25] zeigen, dass es NP-vollständig ist. Es blieb sehr lange offen, ob dieses Ergebnis auch für feste Alphabetgrößen gilt. Casel u. a. [6] konnten zeigen:

Satz 2.4 *SGP ist NP-vollständig für Eingabealphabetgrößen ≥ 24.*

Mit Parameter Alphabetgröße sind somit keine FPT-Ergebnisse zu erwarten. Offen bleibt aber z.B., ob SGP für Binäralphabete in Polynomzeit lösbar ist. Ebenfalls ist offen, ob es Näherungsalgorithmen mit konstantem Approximationsfaktor gibt (selbst bei fester Eingabealphabetgröße). Für die Parameterisierung nach der Anzahl der Nichtterminale der Grammatik (oder hier äquivalent nach der Anzahl der Regeln) lässt sich Mitgliedschaft in XP und W[1]-Schwere beweisen. Der entsprechendende Beweis benötigt allerdings beliebig große Eingabealphabete, sodass es offen bleibt, ob FPT-Mitgliedschaft mit der Gesamtalphabetgröße als Parameter gezeigt werden könnte. Für den Standardparameter k (Grammatikgröße) erwähnen wir noch:

Satz 2.5 *SGP kann in Zeit $O^*(f(k))$ gelöst werden für eine Funktion f.*

Natürlich kann man auch andere Größenmaße für Grammatiken betrachten. Ähnliche Fragen zur Beschreibung endlicher Sprachen werden in [14] diskutiert.

3. Fallbeispiel synchronisierende Wörter

Ein Eingabewort w eines deterministischen endlichen Automaten A heißt *synchronisierend*, falls es einen Zustand s_f gibt, sodass A bei Eingabe von $w \in I^*$ nach s_f überführt wird, gleichwohl wo A mit der Abarbeitung beginnt. Es ist leicht festzustellen, ob es überhaupt ein synchronisierendes Wort für A gibt, jedoch NP-schwer zu entscheiden, ob es (bei vorgelegtem A und k) ein synchronisierendes Wort der Länger höchstens k für A gibt.

Parameter	Komplexität	Polynomieller Kern?
k	W[2]-schwer	—
$\|I\|$	NP-vollständig für $\|I\| = 2$	—
k und $\|I\|$	FPT, Laufzeit $O^*(\|I\|^k)$	Nur falls NP $\subseteq$ coNP/poly
q	FPT, Laufzeit $O^*(2^q)$	Nur falls NP $\subseteq$ coNP/poly

Table 1: Zusammenfassung der Ergebnisse für SW; q ist Zustandsanzahl

Satz 3.1 *[11, 29] Es gelten die in Tabelle 1 aufgeführten Ergebnisse.*

Die positiven FPT-Ergebnisse sind durchweg trivial, jedoch schwer zu verbessern, wie das folgende Resultat zeigt.

Satz 3.2 *Falls SETH (starke Exponentialzeithypothese) gilt, so gibt es kein $\varepsilon > 0$, sodass SW in Zeit $O^*((|I| - \varepsilon)^k)$ gelöst werden könnte.*

Wie auch in einem weiteren Beitrag auf diesem Workshop diskutiert, gibt es zahlreiche Anwendungen für dieses Konzept. So werden Verallgemeinerungen dieses Konzeptes auf stochastische Automaten in [27] diskutiert. Eine Diskussion derartiger Begriffe unter dem Blickwinkel der parameterisierten Komplexität steht noch aus.

Die Reichhaltigkeit natürlicher Parameter ist eine der Charakteristika für Automaten- und Wortprobleme. Als ein weiteres Fallbeispiel verweisen wir auf Morphismenprobleme [13]. Auch der bekannte Graphparameter "Baumweite" findet in diesem Kontext Anwendung [21].

4. Untere Schranken

In der letzten Dekade wurden immer mehr Ergebnisse publiziert, die unter in der Regel schwächeren Annahmen (im Vergleich zum "Goldstandard" P vs. NP) untere Schranken für gewisse Probleme zeigen. Satz 3.2 kann als ein Beispiel betrachtet werden. Klassische(re) Automatenprobleme umfassen Schnittleerheit und Universalität. Exemplarisch sei hier folgendes neuere Ergebnis erwähnt.

Satz 4.1 *[12] Das Universalitätsproblem für binäre NEAs mit q Zuständen kann in Zeit $O^*(2^q)$ gelöst werden, aber unter Annahme der ETH (Exponentialzeithypothese) nicht in Zeit $O^*(2^{o(q)})$.*

Aus diesen Überlegungen ergeben sich auch Ergebnisse wie das folgende, an den vorigen Abschnitt anschließende. In der zitierten Arbeit finden sich viele weitere Resultate von ähnlicher Gestalt.

Satz 4.2 *[12] SW bei DEAs mit q Zuständen kann in Zeit $O^*(2^q)$ gelöst werden, aber unter Annahme der ETH (Exponentialzeithypothese) nicht in Zeit $O^*(2^{o(q)})$.*

Für längenbeschränkte Varianten solcher Automatenprobleme ergeben sich typischerweise untere Schranken durch die SETH, wie schon in Satz 3.2.

Satz 4.3 *[12] In Zeit $O^*(|I|^k)$ kann festgestellt werden, ob ein vorgelegter NEA irgendein Wort der Maximallänge k akzeptiert. Umgekehrt gibt es unter SETH kein $\varepsilon > 0$, sodass dieses Problem in Zeit $O^*((|I|-\varepsilon)^k)$ gelöst werden könnte.*

Es sollte klar sein, dass für (S)ETH-basierte untere Schranken andere Arten von Reduktionen benötigt werden als die in der klassischen Komplexitätstheorie üblichen; so wären Reduktionen, die die Größe der Instanzen superlinear wachsen lassen, in der Regel ungeeignet.

Die so genannte "Feinkörnige Komplexität" benutzt gern auch andere Hypothesen als (S)ETH, oft fußend auf wohluntersuchten Problemen wie z.B. k-*Clique*: Gegeben ist ein Graph G, und gefragt wird, ob G eine Clique der Größe k enthält. Nešetřil und Poljak [18] konnten folgendes Ergebnis zeigen:

Satz 4.4 *3ℓ-Clique kann auf Graphen mit n Knoten in Zeit $O(n^{\omega\ell})$ gelöst werden, wobei ω der Exponent der Quadratmatrixmultiplikation ist, d.h., $\omega < 2,373$.*

Leicht schlechtere Zahlen ergeben sich für k-Clique, wenn k nicht durch drei teilbar ist. Die "passende" k-*Clique-Hypothese* behauptet nun, dass es für kein $\varepsilon > 0$ einen möglicherweise sogar randomisierten Algorithmus für k-Clique gibt, der in Zeit $O(n^{\omega k/3-\varepsilon})$ läuft.

Satz 4.5 *[1] Unter Annahme der k-Clique-Hypothese gibt es keinen Parsing-Algorithmus für (feste) kontextfreie Grammatiken G, der für Wörter w der Länge n in Zeit $O(n^{\omega-\varepsilon})$ entscheidet, ob $w \in L(G)$. Hierbei ist $\varepsilon > 0$ beliebig.*

Die erwähnte Schranke kann bekanntermaßen auch erreicht werden, siehe [28, 22]. Diese unteren Schranken übertragen sich naturgemäß auf Verallgemeinerungen wie Boolesche Grammatiken, für deren Parsing ähnliche obere Schranken gelten [20]. Wesentlich ist, dass dieses recht neue Ergebnis für Grammatiken fester Größe gilt, während Lees Resultate [16] eine Abhängigkeit der Grammatikgröße von der Länge des zu parsenden Wortes einforderte. Bringmann und Wellnitz [5] haben den vorigen Satz (sowie [24]) erweitert zu folgendem Ergebnis, das für Computerlinguisten relevant ist.

Satz 4.6 *Unter Annahme der k-Clique-Hypothese gibt es keinen Parsing-Algorithmus für (feste) baumadjungierende Grammatiken G, der für Wörter w der Länge n in Zeit $O(n^{2\omega-\varepsilon})$ entscheidet, ob $w \in L(G)$. Hierbei ist $\varepsilon > 0$ beliebig.*

Die Reduktionen in der feinkörnigen Komplexität haben zumeist einen ganz eigenen Charakter. Von formalsprachlicher Relevanz sind Arbeiten zu Stringproblemen; wir erwähnen nur zwei Arbeiten: [2, 4], die auch andere Grundprobleme als k-Clique benutzen.

Literatur

[1] A. ABBOUD, A. BACKURS, V. V. WILLIAMS, If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In: V. GURUSWAMI (ed.), *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2015, 98–117.

[2] A. ABBOUD, V. V. WILLIAMS, O. WEIMANN, Consequences of Faster Alignment of Sequences. In: J. ESPARZA, P. FRAIGNIAUD, T. HUSFELDT, E. KOUTSOUPIAS (eds.), *Automata, Languages, and Programming - 41st International Colloquium, ICALP, Proceedings, Part I*. LNCS 8572, Springer, 2014, 39–51.

[3] F. N. ABU-KHZAM, H. FERNAU, M. A. LANGSTON, S. LEE-CULTURA, U. STEGE, A Fixed-Parameter Algorithm for String-to-String Correction. *Discrete Optimization* **8** (2011), 41–49.

[4] K. BRINGMANN, M. KÜNNEMANN, Multivariate Fine-Grained Complexity of Longest Common Subsequence. In: A. CZUMAJ (ed.), *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. SIAM, 2018, 1216–1235.

[5] K. BRINGMANN, P. WELLNITZ, Clique-Based Lower Bounds for Parsing Tree-Adjoining Grammars. In: J. KÄRKKÄINEN, J. RADOSZEWSKI, W. RYTTER (eds.), *28th Annual Symposium on Combinatorial Pattern Matching, CPM*. LIPIcs 78, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 12:1–12:14.

[6] K. CASEL, H. FERNAU, S. GASPERS, B. GRAS, M. L. SCHMID, On the Complexity of Grammar-Based Compression over Fixed Alphabets. In: I. CHATZIGIANNAKIS, M. MITZENMACHER, Y. RABANI, D. SANGIORGI (eds.), *International Colloquium on Automata, Languages and Programming, ICALP*. Leibniz International Proceedings in Informatics (LIPIcs) 55, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, 122:1–122:14.

[7] M. CHARIKAR, E. LEHMAN, D. LIU, R. PANIGRAHY, M. PRABHAKARAN, A. SAHAI, A. SHELAT, The smallest grammar problem. *IEEE Transactions on Information Theory* **51** (2005) 7, 2554–2576.

[8] M. CYGAN, F. FOMIN, L. KOWALIK, D. LOKSHTANOV, D. MARX, M. PILIPCZUK, M. PILIPCZUK, S. SAURABH, *Parameterized Algorithms*. Springer, 2015.

[9] C. DE LA HIGUERA, F. CASACUBERTA, Topology of strings: Median string is NP complete. *Theoretical Computer Science* **230** (2000), 39–48.

[10] R. G. DOWNEY, M. R. FELLOWS, *Fundamentals of Parameterized Complexity*. Texts in Computer Science, Springer, 2013.

[11] H. FERNAU, P. HEGGERNES, Y. VILLANGER, A multi-parameter analysis of hard problems on deterministic finite automata. *Journal of Computer and System Sciences* **81** (2015) 4, 747–765.

[12] H. FERNAU, A. KREBS, Problems on Finite Automata and the Exponential Time Hypothesis. *Algorithms* **10** (2017), 24:1–25.

[13] H. FERNAU, M. L. SCHMID, Y. VILLANGER, On the Parameterised Complexity of String Morphism Problems. *Theory of Computing Systems* **59** (2016) 1, 24–51.

[14] H. GRUBER, M. HOLZER, S. WOLFSTEINER, Concise Description of Finite Languages, Revisited. In: H. FERNAU (ed.), *Theorietag*. Technical Report, Computer Science and Mathematics, Trier University 17-1, 2017, 32–36.

[15] J. E. HOPCROFT, J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*. Reading (MA): Addison-Wesley, 1979.

[16] L. LEE, Fast context-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM* **49** (2002) 1, 1–15.

[17] D. MEISTER, Using swaps and deletes to make strings match. *Theoretical Computer Science* **562** (2015), 606–620.

[18] J. NEŠETŘIL, S. POLJAK, On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae* **26** (1985) 2, 415–419.

[19] F. NICOLAS, E. RIVALS, Hardness results for the center and median string problems under the weighted and unweighted edit distances. *Journal of Discrete Algorithms* **3** (2005) 2-4, 390–415.

[20] A. OKHOTIN, Parsing by matrix multiplication generalized to Boolean grammars. *Theoretical Computer Science* **516** (2014), 101–120.

[21] D. REIDENBACH, M. L. SCHMID, Patterns with bounded treewidth. *Information and Computation* **239** (2014) 0, 87–99.

[22] W. RYTTER, Context-free recognition via shortest paths computation: a version of Valiant's algorithm. *Theoretical Computer Science* **143** (1995) 2, 343–352.

[23] W. RYTTER, Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theoretical Computer Science* **302** (2003), 211–222.

[24] G. SATTA, Tree-adjoining Grammar Parsing and Boolean Matrix Multiplication. *Journal Computational Linguistics* **20** (1994) 2, 173–191.

[25] J. A. STORER, *NP-Completeness Results Concerning Data Compression*. Technical Report 234, Dept. Electrical Engineering and Computer Science, Princeton University, USA, 1977.

[26] J. A. STORER, T. G. SZYMANSKI, Data compression via textual substitution. *Journal of the ACM* **29** (1982) 4, 928–951.

[27] N. F. TRAVERS, J. P. CRUTCHFIELD, Exact Synchronization for Finite-State Sources. *Journal of Statistical Physics* **145** (2011) 5, 1181–1201.

[28] L. G. VALIANT, General Context-Free Recognition in Less than Cubic Time. *Journal of Computer and System Sciences* **10** (1975) 2, 308–315.

[29] V. VOREL, A. ROMAN, Parameterized complexity of synchronization and road coloring. *Discrete Mathematics & Theoretical Computer Science* **17** (2015), 283–306.

[30] R. A. WAGNER, On the complexity of the Extended String-to-String Correction Problem. In: *STOC '75: Proceedings of seventh Annual ACM Symposium on Theory of Computing*. ACM Press, 1975, 218–223.

THEORIE-TAGE 2018 K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 29–32.

Equations in SL(2,Z)

Volker Diekert[(A)]

[(A)]Institut für Formale Methoden der Informatik, Universität Stuttgart, Germany

Abstract

There are classical connections between Hilbert's Tenth Problem, WordEquations, and $\mathrm{SL}(2,\mathbb{Z})$. This note relates them to some recent results showing that the existential theory of $\mathrm{SL}(2,\mathbb{Z})$ is in PSPACE.

Introduction[1]

Hilbert's Tenth Problem appears in the publication to Hilbert's famous 1900 address to the International Congress of Mathematicians in Paris. The problem is stated as follows:

> "Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers."

It was only in 1970 when Matiyasevich showed (based on previous work by Davis, Putnam, and Robinson) that Hilbert's Tenth Problem is undecidable [13]. In order to show undecidability he tried first, but in vain, to show that the problem WordEquations is undecidable. A *word equation* is a pair (U,V) where U and V are strings over finite sets of constants A and variables Ω. A *solution* is a mapping $\sigma : \Omega \to A^*$ which is extended to a homomorphism $\sigma : (A \cup \Omega)^* \to A^*$ leaving the constants invariant such that $\sigma(U) = \sigma(V)$ becomes an identity in A^*. A program of a Turing machine can be viewed a string rewriting system, so it was quite tempting to believe that the problem WordEquations is undecidable. In the language of Hilbert WordEquations is the following problem:

> "Given a word equation: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable."

Why did Matiyasevich try to prove that WordEquations is undecidable when he was interested in Hilbert 10? The connection is via the group $\mathrm{SL}(2,\mathbb{Z})$: the special linear group of 2×2-matrices with integral numerical coefficients:

$$\mathrm{SL}(2,\mathbb{Z}) = \{ \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \mid a,b,c,d \in \mathbb{Z} \wedge ad - bc = 1 \}.$$

[1]The same talk with was given in Kyoto, September 5th, 2018 during a DLT 2018 satellite workshop dedicated to Masami Ito's KIJU and Pál Dömösi's 75th birthday

It is not difficult to see that $\mathrm{SL}(2,\mathbb{N}) = \{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \mid a,b,c,d \in \mathbb{N} \wedge ac - bd = 1\}$ forms a free monoid of rank 2 inside $\mathrm{SL}(2,\mathbb{Z})$. This fact has, for example, a nice application to fast randomized pattern matching [8]. The (unique) basis of $\mathrm{SL}(2,\mathbb{N})$ is given by the two matrices $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. It is therefore possible to encode a word equation over $\{a,b\}$ (which captures the general case) as an equation over $\mathrm{SL}(2,\mathbb{Z})$. Each word variable X becomes a matrix $\left(\begin{smallmatrix} X_1 & X_2 \\ X_3 & X_4 \end{smallmatrix}\right)$. The equation (U,V) becomes a polynomial identity over $\mathbb{Z}$ and we add equations $X_1X_4 - X_2X_3 = 1$ to ensure the determinant is 1. Finally, to guarantee that each $\sigma(X_i)$ is non-negative we employ a theorem of Lagrange that every non-negative integer is the sum of four squares. Having shown that Hilbert 10 is undecidable doesn't mean that WordEquations is undecidable. Indeed, a few years later in 1977, Makanin wrote a seminal paper showing that WordEquations is decidable [11]. Thus, the status in 1977 was as follows: WordEquations is decidable, Hilbert 10 is undecidable, but no results whether or not "Equations over $\mathrm{SL}(2,\mathbb{Z})$" is decidable. It took almost 30 more years to give a positive answer in 2006.

$\mathrm{SL}(2,\mathbb{Z})$

The algebraic structure of the group $\mathrm{SL}(2,\mathbb{Z})$ is well-understood. It is amalgamated product of the two cyclic groups $\mathbb{Z}/6\mathbb{Z}$ and $\mathbb{Z}/4\mathbb{Z}$ over their common subgroup $\mathbb{Z}/2\mathbb{Z}$. Possible generators are the matrices $\rho = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 1 \end{smallmatrix}\right)$ and $\tau = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$. We have $\rho^3 = \tau^2 = \left(\begin{smallmatrix} -1 & 0 \\ 0 & -1 \end{smallmatrix}\right) = -1$. Hence ρ and τ have orders 6 and 4 respectively. Since $\mathrm{SL}(2,\mathbb{Z}) = \mathbb{Z}/4\mathbb{Z} \star_{\mathbb{Z}/2\mathbb{Z}} \mathbb{Z}/6\mathbb{Z}$ there is a natural surjective homomorphism $\gamma : \mathrm{SL}(2,\mathbb{Z}) \to \mathbb{Z}/12\mathbb{Z}$ given by $\gamma(\rho) = 2$ and $\gamma(\tau) = 3$. It is not very hard to see and it follows from a well-known classical result of Newman that the kernel of γ is commutator subgroup $\mathrm{SL}(2,\mathbb{Z})'$ [14, Lem. 1]. Moreover, $\mathrm{SL}(2,\mathbb{Z})'$ is a free group of rank 2 with basis $\left\{\tau\rho\tau\rho^2, \tau\rho^2\tau\rho\right\}$. Thus, $\mathrm{SL}(2,\mathbb{Z})$ is virtually free.

In particular, all finitely generated submonoids of free groups appear as finitely generated submonoids in $\mathrm{SL}(2,\mathbb{Z})$. More general: all rational subsets in free groups embed as rational subsets in $\mathrm{SL}(2,\mathbb{Z})$. In the 1980s, Makanin showed that the existential theory of free groups is decidable [12]; and [4] showed that the problem is actually PSPACE-complete in the presence of *rational constraints*. This opened the way to tackle successfully the problem how to solve equations (with rational constraints) over finitely generated virtually free groups: Lohrey and Sénizergues [10] showed a transfer result for certain amalgamated products and HNN extension; and Dahmani and Guirardel [2] extended the decidability result from virtually free groups to hyperbolic groups by showing how to solve *twisted word equations with rational constraints*. Both papers, [10] and [2], are long and use a quite complicated and advanced machinery. Both approaches use [4] as a black box which in turn is a rather technical paper with more than 30 pages.

As a special case of [10, 2] the decidability of the existential theory of $\mathrm{SL}(2,\mathbb{Z})$ was eventually established, but a concrete algorithm or complexity bound was somehow "lost in translation".[2] So, our journey is not over.

In 2013 Jeż [7] surprised the community by presenting an extremely simple and easy to understand $\mathsf{NSPACE}(n \log n)$-algorithm how to solve word equations. His algorithm did more: it provides an effective description of all solutions and, as a side effect, an

[2] Not meant to be confused with the 2003 American romantic comedy-drama playing in a Tokyo hotel.

NSPACE($n \log n$)-algorithm to decide whether there are infinitely many solutions. Soon after the publication, it became clear that his *recompression technique* copes with rational constraints and the presence of an involution [5]: leading to a simpler proof for a more general result with a better complexity with respect to existential theory in free groups than shown in [4]. Actually, (re-)compression turned out to be even more powerful. [1] showed that the solution set for an equation over free group is an effective EDT0L-language. At least for one of the authors this was a highly unexpected and amazingly simple structural description for the set of all solutions. The underlying technique made it possible to derive the same structural result for twisted word equations [3]. This was more demanding. However, as a consequence, [3] still provides a much easier to understand algorithm how to solve equations in virtually free groups than [10] or [2]. Moreover, for the first time a concrete and reasonable complexity bound was established: PSPACE, or more precisely: NSPACE($n^2 \log n$). Within this complexity it is also possible (for a fixed finitely generated virtually free group) to construct for a system of equations (with rational constraints) an effective description of the solution set as an EDT0L-language. The description is given by a finite nondeterministic automaton where the transitions are labeled by endomorphisms over a free monoid. The automaton is empty if and only if the solution set is empty, it has a directed cycle if and only if the solution set is infinite; and, perhaps most importantly, these properties can be decided in PSPACE, respectively in NSPACE($n^2 \log n$).

The construction for virtually free groups relies heavily on Bass-Serre theory [15], but for $\mathrm{SL}(2,\mathbb{Z})$ everything can be made fully explicit, so that no knowledge of Bass-Serre theory is required to understand the procedure.

With that focus another issue becomes important: when dealing $\mathrm{SL}(2,\mathbb{Z})$ we typically use binary notation for the integer coefficients. Thus, the size of a matrix $\left(\begin{smallmatrix} 1 & n \\ 0 & 1 \end{smallmatrix}\right)$ should be $\mathcal{O}(\log n)$ rather than $\Theta(n)$ in unary notation. However, the PSPACE-complexity for $\mathrm{SL}(2,\mathbb{Z})$ was established for unary notation, only. That is: an element of $\mathrm{SL}(2,\mathbb{Z})$ is written as a (perhaps very long) word over the generators ρ and τ and the unary size the matrix is the word length. Thus, a direct application of [3] to $\mathrm{SL}(2,\mathbb{Z})$ yields EXPSPACE, nothing better.

This would have been a sad ending of a success story. Fortunately, $\mathrm{SL}(2,\mathbb{Z})$ enables elementary number theory: Fibonacci numbers and the Euclidean algorithm pop-up when rewriting a matrix in the free monoid $\mathrm{SL}(2,\mathbb{N})$ in its basis. It is therefore possible to encode every matrix in $\mathrm{SL}(2,\mathbb{Z})$ in binary notation as a word over ρ's and τ's with exponents written in binary without increasing the denotational length by more than a linear factor: see the last section in the paper of Gurevich and Schupp [6]. This leads to a formalism where equations are not necessarily written in plain form, but where also exponents in binary notation are allowed. Now, exponential expressions can be encoded as straight-line programs, and that doesn't lead us outside PSPACE, see for example [9].

Conclusion. Forty years after the publication of Makanin's result [11] we knew how to solve equations over $\mathrm{SL}(2,\mathbb{Z})$ in PSPACE. The complexity holds even if we use binary notation for the matrices. Moreover, we may allow rational constraints and we can do more: we find an effective description of the solution set as an EDT0L-language. This result about the special case of the special linear group $\mathrm{SL}(2,\mathbb{Z})$ is quite special: it is stronger than the corresponding results for free monoids and free groups.

References

[1] L. Ciobanu, V. Diekert, and M. Elder. Solution sets for equations over free groups are EDT0L languages. *International Journal of Algebra and Computation*, 26:843–886, 2016. Conference abstract in Proc. ICALP 2015, LNCS 9135.

[2] F. Dahmani and V. Guirardel. Foliations for solving equations in groups: free, virtually free and hyperbolic groups. *J. of Topology*, 3:343–404, 2010.

[3] V. Diekert and M. Elder. Solutions of twisted word equations, EDT0L languages, and context-free groups. In Proc. ICALP 2017, volume 80 of *LIPIcs*, 96:1–96:14, Dagstuhl, Germany, 2017.

[4] V. Diekert, C. Gutiérrez, and Ch. Hagenah. The existential theory of equations with rational constraints in free groups is PSPACE-complete. *Information and Computation*, 202:105–140, 2005. Conference abstract in Proc. STACS 2001, LNCS 2010.

[5] V. Diekert, A. Jeż, and W. Plandowski. Finding all solutions of equations in free groups and monoids with involution. *Information and Computation*, 251:263–286, 2016. Conference abstract in Proc. CSR 2014, LNCS 8476.

[6] Y. Gurevich and P. Schupp. Membership problem for the modular group. *SIAM J. Comput.*, 37:425–459, 2007.

[7] A. Jeż. Recompression: a simple and powerful technique for word equations. In Proc. *STACS*, volume 20 of *LIPIcs*, 233–244, Dagstuhl, Germany, 2013. Journal version in J. ACM 2016 with DOI http://dx.doi.org/10.1145/2743014.

[8] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31:249–260, 1987.

[9] M. Lohrey. *The Compressed Word Problem for Groups*. Springer, 2014.

[10] M. Lohrey and G. Sénizergues. Theories of HNN-extensions and amalgamated products. In Proc. ICALP, LNCS 4052: 504–515, 2006.

[11] G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. Sbornik*, 103:147–236, 1977. English transl. in Math. USSR Sbornik 32 (1977).

[12] G. S. Makanin. Decidability of the universal and positive theories of a free group. *Izv. Akad. Nauk SSSR*, Ser. Mat. 48:735–749, 1984. In Russian; English translation in: *Math. USSR Izvestija, 25*, 75–88, 1985.

[13] Yu. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.

[14] M. Newman. The structure of some subgroups of the modular group. *Illinois J. Math.*, 6:480–487, 1962.

[15] J.-P. Serre. *Trees*. Springer, 1980. French original 1977.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 33–36.

Completely Reachable Automata: An Interplay Between Semigroups, Automata, and Trees

E. A. Bondar and M. V. Volkov

Institute of Natural Sciences and Mathematics
Ural Federal University, Lenina 51, 620000 Ekaterinburg, Russia
bondareug@gmail.com, mikhail.volkov@usu.ru

Zusammenfassung

A complete deterministic finite automaton in which every non-empty subset of the state set occurs as the image of the whole state set under the action of a suitable input word is called completely reachable. We overview recent results on synchronization of completely reachable automata with transition monoids of maximal or minimal possible size.

A *complete deterministic finite automaton* (DFA) is a triple $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, where Q and Σ are finite sets called the *state set* and the *input alphabet* respectively, and $\delta\colon Q \times \Sigma \to Q$ is a total map called the *transition function*. Let Σ^* be the *free monoid* over Σ, that is, the collection of all finite words over Σ, including the empty word. The function δ extends to a function $Q \times \Sigma^* \to Q$ (still denoted by δ) in the usual way: for every $q \in Q$ and $w \in \Sigma^*$, we set $\delta(q, w) := q$ if w is empty and $\delta(q, w) := \delta(\delta(q, v), a)$ if $w = va$ for some word $v \in \Sigma^*$ and some letter $a \in \Sigma$. Thus, via δ, every word $w \in \Sigma^*$ induces a transformation of the set Q. The collection of all transformations of Q that arise this way is called the *transition monoid* of $\mathscr{A}$.

Whenever we deal with a fixed DFA, we suppress the sign of the transition function. This means that we introduce the DFA as the pair $\langle Q, \Sigma\rangle$ rather than the triple $\langle Q, \Sigma, \delta\rangle$ and write $q\,.\,w$ for $\delta(q, w)$ and $P\,.\,w$ for $\{\delta(q, w) \mid q \in P\}$ where P is any non-empty subset of Q.

Given a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$, we say that a non-empty subset $P \subseteq Q$ is *reachable* in $\mathscr{A}$ if $P = Q\,.\,w$ for some word $w \in \Sigma^*$. A DFA is called *completely reachable* if every non-empty subset of its state set is reachable. Our paper [1] lists several motivations for considering completely reachable automata. In particular, such DFAs have appeared in the study of descriptional complexity of formal languages [7, 1] and in relation to synchronizing automata [3, 5]. Recall that a DFA $\mathscr{A} = \langle Q, \Sigma\rangle$ is called *synchronizing* if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state it started: $q\,.\,w = q'\,.\,w$ for all $q, q' \in Q$. This amounts to saying that a singleton is reachable in $\mathscr{A}$, whence every completely reachable automaton is synchronizing.

If $\mathscr{A} = \langle Q, \Sigma\rangle$ is a synchronizing automaton, any word $w \in \Sigma^*$ such that $Q\,.\,w$ is a singleton is called a *reset word* for $\mathscr{A}$. The minimum length of reset words for $\mathscr{A}$ is called the *reset*

Supported by the Russian Foundation for Basic Research, grant no. 16-01-00795, the Ministry of Education and Science of the Russian Federation, project no. 1.3253.2017, and the Competitiveness Enhancement Program of Ural Federal University.

threshold of $\mathscr{A}$. In 1964, Černý [2] constructed for each $n > 1$, a synchronizing automaton $\mathscr{C}_n$ with 2 input letters and n states that has reset threshold $(n-1)^2$. The states of $\mathscr{C}_n$ are the residues modulo n, and the input letters a and b act as follows:

$$0\,.\,a := 1,\ \ m\,.\,a := m \text{ for } 0 < m < n, \quad m\,.\,b := m+1 \pmod{n}.$$

The automata $\mathscr{C}_n$ provide a lower bound for the maximum reset threshold for synchronizing automata with n states. The famous Černý conjecture claims that these automata represent the worst possible case; in other terms, it the claim is that every synchronizing automaton with n states can be reset by a word of length $(n-1)^2$. The conjecture, first stated in the 1960s, resists researchers' efforts for more than 50 years. The best upper bound achieved so far is cubic in n; it is due to Szykuła [9] and is only slightly better than the upper bound $\frac{n^3-n}{6}$ established by Pin [8] and Frankl [4] approx. 35 years ago.

The Černý Conjecture has proved to be hard in general, and therefore, a natural strategy consists in considering its restrictions to some special classes of DFAs. We refer to the survey [10] and the chapter [6] of the forthcoming "Handbook of Automata Theory" for an overview of this research direction. Here we consider synchronization of completely reachable automata. It is known (see [7, 3]) that the automata $\mathscr{C}_n$ in the Černý series are completely reachable so that the lower bound $(n-1)^2$ for the reset threshold of automata with n states persists in the class of completely reachable automata.

Don [3, Conjecture 18] has formulated the following conjecture: if a completely reachable automaton $\mathscr{A}$ has n states, then every subset of size k can be reached in $\mathscr{A}$ via a word of length at most $n(n-k)$. It is easy to see that if Don's conjecture holds, then every completely reachable automaton satisfies the Černý conjecture. At the moment, it is not even known whether there exists a constant C such that in an arbitrary completely reachable DFA with n states, every non-empty subset can be reached by a word of length n^C. So far, some progress has been achieved for DFAs whose transition monoid is equal to the full monoid of transformations of the state set [5]. Clearly, automata with the latter property are completely reachable but the Černý automata $\mathscr{C}_n$ with $n \geq 3$ are not in this family of completely reachable automata since the full transformation monoid on a set with more than 2 elements requires at least 3 generators. Nevertheless, there exists a series of n-state automata $\mathscr{V}_n$, $n \geq 3$, in this class with reset threshold $\frac{n(n-1)}{2}$ [5, Theorem 4]. The state set of $\mathscr{V}_n$ is $\{0,\dots,n-1\}$ and the input alphabet consists of n letters $a_1,\dots,a_n$. The transition function is defined as follows:

$$\begin{cases} i\,.\,a_j := i & \text{for } 0 \leq i,j < n,\ i \neq j,\ i \neq j+1,\\ i\,.\,a_i := i-1 & \text{for } 0 < i \leq n-1,\\ i\,.\,a_{i+1} := i+1 & \text{for } 0 \leq i < n-1,\\ 0\,.\,a_n = 1\,.\,a_n := 0,\ i\,.\,a_n := i & \text{for } 2 \leq i \leq n-1. \end{cases}$$

Simply speaking, every letter a_i for $i \leq n-1$ swaps the states i and $i-1$ and fixes the other states. The letter a_n brings both 0 and 1 to 0 and fixes the other states. For an illustration, the automaton $\mathscr{V}_5$ is shown in Fig. 1.

On the other hand, it is shown in [5, Lemmas 5 and 6] that if $\mathscr{A} = \langle Q, \Sigma\rangle$ is an arbitrary n-state DFA whose transition monoid contains all transformations of the set Q, then for every proper non-empty subset $P \subset Q$, there exist a word $w \in \Sigma^*$ of length at most $2n-2$ and a set $P' \subseteq Q$ such that $|P| < |P'|$ and $P'\,.\,w = P$. From this, it readily follows that, first, every subset

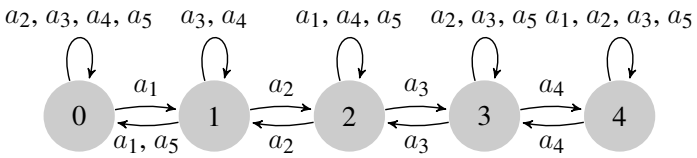


Figure 1: The automaton $\mathscr{V}_5$

of size k can be reached in $\mathscr{A}$ via a word of length at most $2(n-2)(n-k)$ and, second, the reset threshold of $\mathscr{A}$ does not exceed $2n^2-6n+5$.

Thus, for the reset threshold of DFAs with full transition monoid, we have lower and upper bounds with the same order of magnitude, namely, $\Theta(n^2)$. For follow-up work, one direction is to refine the bounds with respect to the constants that do not match yet. Also, a lower bound for the reset threshold of such automata with a fixed number of letters is of interest, since the number of letters in the automata $\mathscr{V}_n$ is equal to the number of states.

Now we switch to completely reachable automata being in a sense the extreme opposites of DFAs with full transition monoids, namely, on completely reachable automata whose transition monoids have minimal possible size. Clearly, if a completely reachable automaton has n states, then its transition monoid has at least 2^n-1 elements since each non-empty subset must occur as the image of a transformation induced by an input word. A completely reachable automaton with n states and exactly 2^n-1 elements in its transition monoid is called *minimal*. In [1] we have given a classification of minimal completely reachable automata: they are parameterized by full binary trees satisfying certain subordination conditions.

Recall that a binary tree is *full* if each its vertex v either is a leaf or has exactly two children. We refer to the left/right child of v as the *son/daughter* of v. If Γ is a tree and v is its vertex, Γ_v stands for the subtree of Γ rooted at v. A *homomorphism* between trees is a map between their vertex sets that preserves the roots, the parent–child relation and the genders of non-root vertices. If u and v are vertices of a tree Γ, we say that u *subordinates* v if there is a 1-1 homomorphism $\Gamma_u \to \Gamma_v$. A *respectful tree* is a full binary tree such that:
• if a male vertex has a nephew, the nephew subordinates his uncle;
• if a female vertex has a niece, the niece subordinates her aunt.
For instance, each *perfect* binary tree (that is, a full binary tree in which all leaves have the same depth, see Fig. 2) is respectful.

In [1, Section 4] we describe a construction that, given an arbitrary respectful tree Γ with n leaves, produces a minimal completely reachable automaton $\mathscr{A}(\Gamma)$ with n states and $2n-2$ input letters (one for each non-root vertex of Γ). Conversely, for every minimal completely reachable automaton $\mathscr{A}$ with n states, there exists a respectful tree Γ with n leaves such that the automata $\mathscr{A}$ and $\mathscr{A}(\Gamma)$ are *syntactically equivalent*, that is, their transition monoids coincide. Here we add the following result to this characterization of minimal completely reachable automata: for every minimal completely reachable automaton, its reset threshold is equal to the minimum length of a path from the root to a leaf in the corresponding respectful tree. In particular, the reset threshold of minimal completely reachable automaton with n states does not exceed $\log_2 n$, and this bound is attained for the minimal completely reachable automata corresponding to perfect binary trees.

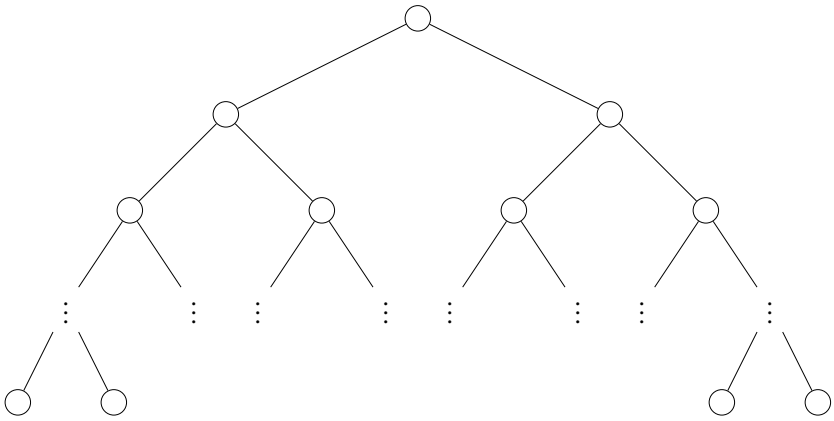


Figure 2: Perfect binary tree

Literatur

[1] E. A. BONDAR, M. V. VOLKOV, Completely Reachable Automata. In: C. CÂMPEANU, F. MANEA, J. SHALLIT (eds.), *Descriptional Complexity of Formal Systems, 18th Int. Conf., DCFS 2016*. LNCS 9777, Springer, 2016, 1–17.

[2] J. ČERNÝ, Poznámka k homogénnym eksperimentom s konečnými automatami. *Matematicko-fyzikalny Časopis Slovenskej Akadémie Vied* **14** (1964) 3, 208–216. (in Slovak).

[3] H. DON, The Černý Conjecture and 1-Contracting Automata. *Electr. J. Comb.* **23** (2016) 3, P3.12.

[4] P. FRANKL, An extremal problem for two families of sets. *European J. Combinatorics* **3** (1982), 125–127.

[5] F. GONZE, V. V. GUSEV, B. GERENCSÉR, R. M. JUNGERS, M. V. VOLKOV, On the Interplay Between Babai and Černý's Conjectures. In: É. CHARLIER, J. LEROY, M. RIGO (eds.), *Developments in Language Theory – 21st Int. Conf., DLT 2017*. LNCS 10396, Springer, 2017, 185–197.

[6] J. KARI, M. VOLKOV, Černý's conjecture and the Road Coloring Problem. In: J.-E. PIN (ed.), *Handbook of Automata Theory*. chapter 15, EMS Publishing House. (in print).

[7] M. I. MASLENNIKOVA, Reset complexity of ideal languages. In: M. BIELIKOVÁ, G. FRIEDRICH, G. GOTTLOB, S. KATZENBEISSER, R. ŠPÁNEK, G. TURÁN (eds.), *Current Trends in Theory and Practice of Computer Science, 38th Int. Conf., SOFSEM 2012. Vol. II*. Inst. Comp. Sci. Acad. Sci. Czech Republic, 2012, 33–44.

[8] J.-E. PIN, On two combinatorial problems arising from automata theory. *Ann. Disc. Math.* **17** (1983), 535–548.

[9] M. SZYKUŁA, Improving the Upper Bound on the Length of the Shortest Reset Word. In: R. NIEDERMEIER, B. VALLÉE (eds.), *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018*. LIPIcs 96, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2018, 56:1–56:13.

[10] M. V. VOLKOV, Synchronizing automata and the Černý conjecture. In: C. MARTÍN-VIDE, F. OTTO, H. FERNAU (eds.), *Proc. 2nd International Conference on Language and Automata Theory and Applications*. LNCS 5196, Springer, 2008, 11–27.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 37–41.

The Satisfiability of Word Equations: Decidable and Undecidable Theories

Joel D. Day[(A)] Vijay Ganesh[(B)] Paul He[(B)] Florin Manea[(A)]
Dirk Nowotka[(A)]

[(A)]Kiel University, Germany,
{jda,fpa,flm}@informatik.uni-kiel.de

[(B)]Waterloo University, Canada
vijay.ganesh@uwaterloo.ca,paul.he@edu.uwaterloo.ca

Abstract

The study of word equations (or the existential theory of equations over free monoids) is a central topic in mathematics and theoretical computer science. The problem of deciding whether a word equation has a solution was shown to be decidable by Makanin in the late 1970s, and since then considerable work has been done on this topic. Recently, this decidability question has gained critical importance in the context of string SMT solvers for security analysis. Further, many extensions (e.g., quantifier-free word equations with linear arithmetic over the length function) and fragments (e.g., restrictions on the number of variables) of this theory are important from a theoretical point of view, as well as for program analysis applications. Motivated by these considerations, we prove several new results and thus shed light on the boundary between decidability and undecidability for many fragments of the first order theory of word equations and their extensions.

1. Overview

A *word equation* is a formal equality $U = V$, where U and V are words (called the left and right side of the equation respectively) over an alphabet $A \cup X$; $A = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots\}$ is the alphabet of *constants* or *terminals* and $X = \{x_1, x_2, x_3, \ldots\}$ is the set of *variables*. A *solution* to the equation $U = V$ is a morphism $h : (A \cup X)^* \to A^*$ that acts as the identity on A and satisfies $h(U) = h(V)$; h is called the assignment to the variables of the equation. For instance, $U = x_1\mathtt{ab}x_2$ and $V = \mathtt{a}x_1x_2\mathtt{b}$ define the equation $x_1\mathtt{ab}x_2 = \mathtt{a}x_1x_2\mathtt{b}$, whose solutions are the morphisms h with $h(x_1) = \mathtt{a}^k$, for $k \geq 0$, and $h(x_2) = \mathtt{b}^\ell$, for $\ell \geq 0$. An equation is *satisfiable* (in A^*) if it admits a solution $h : (A \cup X)^* \to A^*$. A set (or system) of equations is satisfiable if there exists an assignment of the variables of the equations in this set that is a solution for all equations. In logical terms, word equations are often investigated as fragments of the first order theory $\mathsf{FO}(A^*, \cdot)$ of strings. Karhumäki et al. [18] showed that deciding the satisfiability of a system of word equations, that is, checking the truth of formulas from the existential theory Σ_1 of $\mathsf{FO}(A^*, \cdot)$, can be reduced to deciding the satisfiability of a single (more complex) word equation that encodes the respective system.

The existential theory of word equations has been studied for decades in mathematics and theoretical computer science with a particular focus on the decidability of the satisfiability of logical formulae defined over word equations. Quine [27] proved in 1946 that the first-order theory of word equations is equivalent to the first-order theory of arithmetic, which is known to be undecidable. In order to solve Hilbert's tenth problem in the negative [14], Markov later showed a reduction from word equations to Diophantine equations (see [21, 22] and the references therein), in the hopes that word equations would prove to be undecidable. However, Makanin [22] proved in 1977 that the satisfiability of word equations *is* in fact decidable. Though Markov's approach was unsuccessful, similar ones, based on extended theories of word equations, can also be explored. Matiyasevich [25] showed in 1968 a reduction from the more powerful theory of word equations with linear length constraints (i.e., linear relations between word lengths) to Diophantine equations. Whether this theory is decidable remains a major open problem. More than a decade after Makanin showed that the satisfiability of word equations is decidable, the focus shifted towards identifying the complexity of solving word equations. Plandowski [26] showed in 1999 that this problem is in PSPACE. Recently, in a series of papers (see specifically e.g., [15, 16]), Jez applied a new technique called recompression to word equations. This lead to, ulimately, a proof that the satisfiability of word equations can be decided in linear space. However, there is a mismatch between this upper bound and the known lower bound: solving word equations is NP-hard.

In recent years, deciding the satisfiability of systems of word equations has also become an important problem in fields such as formal verification and security where string solvers such as HAMPI [19], CVC4 [3], Stranger [30], ABC [2], Norn [1], S3P [28] and Z3str3 [4] have become more popular. However, in practice more functionality than just word equations is required, so solvers often extend the theory of word equations with certain functions (e.g., linear arithmetic over the length, replace-all, extract, reverse, etc.) and predicates (e.g., numeric-string conversion predicate, regular-expression membership, etc.). Most of these extensions are not expressible by word equations, in the sense introduced by Karhumäki et al. [18], and some of them lead to undecidable theories. On the one hand, regular (or rational) constraints or constraints based on involutions (allowing to model the mirror image, or, when working with equations in free groups, inverse elements), are not expressible, see [6, 18], but adding them to word equations preserves the decidability [8]. As mentioned above, whether the theory of word equations enhanced with a length function is decidable is still a major open problem. On the other hand, the satisfiability of word equations extended with a *replace-all* operator was shown to be undecidable in [20], and the same holds when a numeric-string conversion predicate is added (see the Appendix). Due to this very complex and fuzzy picture, none of the solvers mentioned above has a complete algorithm.

Our Contributions: In this setting, our work aims to provide a better understanding of the boundary between extensions and restrictions of the theory of word equations for which satisfiability is decidable and, respectively, undecidable.

Firstly, we present a series of undecidability results for the Σ_1-fragment of $\mathsf{FO}(A^*, \cdot)$ extended with simple predicates or functions. In the main result on this topic, we show that extending Σ_1 with constraints imposing that a string is the morphic image of another one also leads to an undecidable theory. These results are related to the study of theories of quantifier-free word equations constrained by very simple relations, see, e.g., [6, 13]. While our results do

not settle the decidability of the theory of word equations with length constraints, they enforce the intuitive idea that enhancing the theory of word equations with predicates providing little control on the combinatorial structure of the solutions of the equation leads to undecidability.

We further explore the border between decidability and undecidability when considering formulae over word equations allowing at most one quantifier alternation. We show that checking the truth of an arbitrary Σ_2-formula is equivalent to, on the one hand, checking the truth of a $\exists^*\forall^*$-quantified terminal-free formula, or, on the other hand, to a single $\exists^*\forall^*$-quantified inequation whose sides contain at most two terminals. Since the Inclusion of Pattern Languages problem (see [5, 11, 17]) can be reformulated as checking the truth of a single $\exists^*\forall^*$-quantified inequation whose sides contain at most two terminals and are variable disjoint, and it is undecidable, we obtain a clear image of the simplest undecidable classes of Σ_2-formulae. Consequently, we consider decidable cases. Complementary to the above, we show that the satisfiability in an arbitrary free monoid A^* of quantifier free *positive* formulae over word equations (formulae obtained by iteratively applying only conjunction and disjunction to word equations of the form $U = V$), in which we have at most one terminal $\mathtt{a} \in A$ (appearing zero or several times) and no restriction on the usage of variables, enhanced with linear length constraints, is decidable, and, moreover, NP-complete. The decidability is preserved when considering positive Σ_2-formulae of this kind, as opposed to the case of arbitrary Σ_2 terminal-free formulae, mentioned above. Moreover, if we allow negated equations in the quantifier-free formulae (so arbitrary Σ_1-formulae) with at most one terminal, and length constraints, we obtain a decidable theory if and only if the general theory of equations with length constraints is decidable. Putting together these results, we draw a rather precise border between the decidable and undecidable subclasses of the Σ_2-fragment over word equations, defined by restrictions on the number of terminals allowed to occur in the equations and the presence or absence of inequations. As a corollary, we can show that deciding the truth of arbitrary formulae from the positive Σ_2-fragment of $\mathsf{FO}(A^*, \cdot)$ (i.e., $\exists^*\forall^*$ quantified positive formulae), without length constraints, is decidable. The resulting proof follows arguments partly related to those in [23, 9]. This result is strongly related to the work of [27, 12, 10], in which it was shown that the validity of sentences from the positive Π_2-fragment of $\mathsf{FO}(A^*, \cdot)$ (i.e., the quantifier alternation is $\forall^*\exists^*$) is undecidable, as well as to the results of [29] in which it was shown that the truth of arbitrarily quantified positive formulae over equations is decidable *over an infinite alphabet of terminals*.

We then extend our approach in a way partly motivated by the practical aspects of solving word equations. Most equations that can be successfully solved by string solvers (e.g., Z3str3) must be in *solved form* [12], or must not contain *overlapping variables* [31]. In a sense, this suggests that in practice it is interesting to find equations with restricted form that can be solved in reasonable time. We analyse, from a theoretical point of view, one of the simplest classes of equations that are not in solved form or contain equations with overlapping variables, namely strictly regular-ordered equations (each variable occurs exactly once in each side, and the order in which the variables occur is the same). We show that the satisfiability of such equations, even when enhanced with various predicates, is decidable. In particular we show that when extended with regular constraints (given by DFAs), linear length constraints, abelian equivalence constraints (two variables should be substituted for abelian-equivalent words), subword constraints (one variable should be a (scattered) subword of another), and $Eq_{\mathtt{a}}$ constraints (two variables should have the same number of occurrences of a letter $\mathtt{a}$), the satisfiability problem remains NP-complete. Thus, there is hope that they can be solved reasonably fast by string solvers

based on, e.g., SAT-solvers. This line of results is also related to the investigations initiated in [24, 7], in which the authors were interested in the complexity of solving equations of restricted form. In the most significant result of [7], it was shown that deciding the satisfiability of strictly regular-ordered equations (with or without regular constraints) is NP-complete, which makes this class of word equations one of simplest known classes of word equations that are hard to solve. Although these results regard a very restricted class of equations, they might provide some insights in tackling harder classes, such as, e.g., quadratic equations.

References

[1] P. A. ABDULLA, M. F. ATIG, Y. CHEN, L. HOLÍK, A. REZINE, P. RÜMMER, J. STENMAN, Norn: An SMT Solver for String Constraints. In: *Proc. CAV 2015*. LNCS 9206, 2015, 462–469.

[2] A. AYDIN, L. BANG, T. BULTAN, Automata-Based Model Counting for String Constraints. In: *Proc. CAV 2015*. LNCS 9206, 2015, 255–272.

[3] C. BARRETT, C. L. CONWAY, M. DETERS, L. HADAREAN, D. JOVANOVIĆ, T. KING, A. REYNOLDS, C. TINELLI, CVC4. In: *Proc. CAV 2011*. LNCS 6806, 2011, 171–177.

[4] M. BERZISH, V. GANESH, Y. ZHENG, Z3str3: A string solver with theory-aware heuristics. In: *Proc. FMCAD 2017*. IEEE, 2017, 55–59.

[5] J. BREMER, D. D. FREYDENBERGER, Inclusion problems for patterns with a bounded number of variables. *Inf. Comput.* **220** (2012), 15–43.

[6] J. R. BÜCHI, S. SENGER, Definability in the existential theory of concatenation and undecidable extensions of this theory. *Z. für math. Logik Grundlagen d. Math.* **47** (1988), 337–342.

[7] J. D. DAY, F. MANEA, D. NOWOTKA, The Hardness of Solving Simple Word Equations. In: *Proc. MFCS 2017*. LIPIcs 83, 2017, 18:1–18:14.

[8] V. DIEKERT, A. JEŻ, W. PLANDOWSKI, Finding all solutions of equations in free groups and monoids with involution. *Inf. Comput.* **251** (2016), 263–286.

[9] V. DIEKERT, M. LOHREY, Existential and Positive Theories of Equations in Graph Products. *Theory Comput. Syst.* **37** (2004) 1, 133–156.

[10] V. G. DURNEV, Undecidability of the positive $\forall\exists$-theory of a free semigroup. *Sib. Math. J.* **36.5** (1995), 917–929.

[11] D. D. FREYDENBERGER, D. REIDENBACH, Bad news on decision problems for patterns. *Information and Computation* **208** (2010) 1, 83–96.

[12] V. GANESH, M. MINNES, A. SOLAR-LEZAMA, M. C. RINARD, Word Equations with Length Constraints: What's Decidable? In: *HVC 2012, Revised Selected Papers*. LNCS 7857, 2013, 209–226.

[13] S. HALFON, P. SCHNOEBELEN, G. ZETZSCHE, Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In: *Proc. LICS 2017*. IEEE Computer Society, 2017, 1–12.

[14] D. HILBERT, Mathematische probleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* **1900** (1900), 253–297.

[15] A. JEŻ, Recompression: a simple and powerful technique for word equations. In: *Proc. STACS 2013*. LIPIcs 20, 2013, 233–244.

[16] A. JEŻ, Word Equations in Nondeterministic Linear Space. In: *Proc. ICALP 2017*. LIPIcs 80, 2017, 95:1–95:13.

[17] T. JIANG, A. SALOMAA, K. SALOMAA, S. YU, Decision Problems for Patterns. *J. Comput. Syst. Sci.* **50** (1995) 1, 53–63.

[18] J. KARHUMÄKI, F. MIGNOSI, W. PLANDOWSKI, The expressibility of languages and relations by word equations. *Journal of the ACM (JACM)* **47** (2000) 3, 483–505.

[19] A. KIEZUN, V. GANESH, P. J. GUO, P. HOOIMEIJER, M. D. ERNST, HAMPI: a solver for string constraints. In: *Proc. ISSTA 2009*. ACM, 2009, 105–116.

[20] A. W. LIN, P. BARCELÓ, String solving with word equations and transducers: towards a logic for analysing mutation XSS. In: *ACM SIGPLAN Notices*. 51, ACM, 2016, 123–136.

[21] M. LOTHAIRE, *Combinatorics on Words*. Addison-Wesley, 1983.

[22] G. S. MAKANIN, The problem of solvability of equations in a free semigroup. *Sbornik: Mathematics* **32** (1977) 2, 129–198.

[23] G. S. MAKANIN, Decidability of the Universal and Positive Theories of a Free Group. *Mathematics of the USSR-Izvestiya* **25** (1985) 1, 75.

[24] F. MANEA, D. NOWOTKA, M. L. SCHMID, On the Solvability Problem for Restricted Classes of Word Equations. In: *Proc. DLT 2016*. LNCS 9840, 2016, 306–318.

[25] Y. V. MATIYASEVICH, A connection between systems of words-and-lengths equations and Hilbert's tenth problem. *Zapiski Nauchnykh Seminarov POMI* **8** (1968), 132–144.

[26] W. PLANDOWSKI, Satisfiability of word equations with constants is in PSPACE. In: *Proc. FOCS 1999*. IEEE, 1999, 495–500.

[27] W. V. QUINE, Concatenation as a basis for arithmetic. *J. Symb. Log.* **11** (1946) 4, 105–114.

[28] M.-T. TRINH, D.-H. CHU, J. JAFFAR, Progressive Reasoning over Recursively-Defined Strings. In: *Proc. CAV 2016*. LNCS 9779, 2016, 218–240.

[29] J. M. VAZENIN, B. V. ROZENBLAT, Decidability of the positive theory of a free countably generated semigroup. *Math. USSR Sb.* **44.1** (1983), 109–116.

[30] F. YU, M. ALKHALAF, T. BULTAN, STRANGER: An Automata-based String Analysis Tool for PHP. In: *Proc. TACAS 2010*. LNCS 6015, 2010.

[31] Y. ZHENG, V. GANESH, S. SUBRAMANIAN, O. TRIPP, M. BERZISH, J. DOLBY, X. ZHANG, Z3str2: an efficient solver for strings, regular expressions, and length constraints. *Formal Methods in System Design* **50** (2017) 2-3, 249–288.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 42–42.

Iterative Arrays with Bounded Communication

Andreas Malcher[(A)]

[(A)]Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
andreas.malcher@informatik.uni-giessen.de

Zusammenfassung

Iterative arrays with restricted internal inter-cell communication are investigated. A quantitative measure for the communication is defined by counting the number of uses of the links between cells and it is differentiated between the sum of all communications of an accepting computation and the maximum number of communications per cell occurring in accepting computations. The computational complexity of both classes of devices is investigated and put into relation. In addition, a strict hierarchy depending on the maximum number of communications per cell is established. Finally, it is shown that almost all commonly studied decidability questions are not semidecidable for iterative arrays with restricted communication and, moreover, it is not semidecidable as well whether a given iterative array belongs to a given class with restricted communication.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 43–46.

Exact and Approximated Computation of the Locality Number of Words

Joel D. Day[(A)] Pamela Fleischmann[(A)] Florin Manea[(A)]
Markus L. Schmid[(B)]

[(A)]Kiel University, Germany,
{jda,fpa,flm}@informatik.uni-kiel.de

[(B)]Trier University, Germany,
mlschmid@mlschmid.de

Abstract

We investigate the problem of computing, for a given word, its locality number, a recently introduced structural complexity measure for words with algorithmic applications in matching patterns with variables. We answer the open question whether the locality number can be efficiently computed by showing the decision variant of the problem to be NP-complete. Additionally, we present a pathwidth-based approximation algorithm and improve on the known exact algorithms.

1. Overview

In this work, we consider the *locality* of words (also called strings), which is a structural parameter that has been first introduced in [5]. More precisely, a word is k-local if there exists an order of the symbols occurring in that word such that, if we *mark* the symbols in the respective order (which is called a *marking sequence*), at each step there are at most k contiguous blocks of marked symbols in the word. This k is called the *marking number* of that marking sequence. The *locality number* of a word is the smallest k for which that word is k-local, or, in other words, the minimum marking number over all marking sequences. For example, the marking sequence $\sigma = (\mathtt{x}, \mathtt{y}, \mathtt{z})$ marks the word $\alpha = \mathtt{xyxyzxz}$ as follows (marked blocks are illustrated by overlines): $\mathtt{xyxyzxz}$, $\overline{\mathtt{x}}\mathtt{y}\overline{\mathtt{x}}\mathtt{yz}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxy}}\mathtt{z}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxyzxz}}$; thus, the *marking number* of σ is 3. In fact, all marking sequences for α have a marking number of 3, except $(\mathtt{y}, \mathtt{x}, \mathtt{z})$, for which it is 2: $\mathtt{x}\overline{\mathtt{y}}\mathtt{x}\overline{\mathtt{y}}\mathtt{zxz}$, $\overline{\mathtt{xyxy}}\mathtt{z}\overline{\mathtt{xz}}$, $\overline{\mathtt{xyxyzxz}}$. Thus, the locality number of α, denoted by $\mathsf{loc}(\alpha)$, is 2. The locality number of a word describes how many separate (or isolated) marked regions must at least be maintained in exploring the word; thus, it can be interpreted as a complexity measure (if we associate some cost or resource per marked region).

The original motivation for the concept of locality comes from pattern matching [5]. A *pattern* is a word that consists of *terminal symbols* (e. g., $\mathtt{a}, \mathtt{b}, \mathtt{c}$), treated as constants, and *variables* (e. g., $x_1, x_2, x_3, \ldots$). A pattern is mapped to a word by substituting the variables by strings of terminals. For example, $x_1x_1\mathtt{bab}x_2x_2$ can be mapped to $\mathtt{acacbabcc}$ or $\mathtt{ccbabaa}$ by the substitutions $(x_1 \to \mathtt{ac}, x_2 \to \mathtt{c})$ and $(x_1 \to \mathtt{c}, x_2 \to \mathtt{a})$, respectively. If a pattern α can be mapped

to a given string of terminals w, we say that α matches w. Deciding whether a given pattern matches a given word is one of the most important problems that arise in the study of patterns with variables. Many applications appear in domains like combinatorics on words (word equations [13], unavoidable patterns [15]), pattern matching (generalized function matching [1]), language theory (pattern languages [2]), learning theory (inductive inference [2, 6, 16, 17], PAC-learning [14]), database theory (extended conjunctive regular path queries [3]), as well as in practice, e.g., extended regular expressions with backreferences [11, 12], used in programming languages like Perl, Java, Python, etc. Unfortunately, the *matching problem* is NP-complete [2] in general. This is especially bad for some computational tasks on patterns which implicitly solve the matching problem and are therefore also intractable. For instance, this is the case of the task of finding descriptive patterns for a set of strings [8], which is useful in the context of learning theory. Consequently, a thorough analysis [5, 7, 9, 10, 18, 19] of the complexity of the matching problem has shown that some classes of patterns, defined by restricting structural parameters, can be matched in polynomial time.

All these "tractability parameters" have a unifying theory: in [18] it has been shown that any class of patterns with bounded treewidth (for suitable graph representations) can be matched in polynomial-time and virtually all structural parameters that lead to tractability investigated in [5, 7, 18, 19] are also bounding the treewidth (the efficiently matchable classes investigated in [4] are one of the rare cases with an unbounded treewidth). However, this theory is only suitable for proving tractability of certain classes of patterns, while actually computing and algorithmically exploiting the treewidth of a pattern is rather problematic (see the discussion in [7, 18]). The tractability parameters, on the other hand, are formulated in a way that gives some kind of obvious dynamic programming approach, e. g., the *scope coincidence degree*, investigated in [7, 18], is just the maximum number of variables that can be "active" at the same time (i. e., they have already occurred and will occur again) when moving through the pattern from left to right (thus, bounding the number of factors that have to be stored by a dynamic programming algorithm). Similarly, for the locality number, the corresponding marking sequence can be seen as an instruction for a dynamic programming algorithm: assigning the variables in the order given by the marking sequence, the number of factors that we have to keep track of is bounded by the locality number (see [5] for details). Another strong point for the usefulness of most of these parameters (in contrast to the treewidth) is that they are all string parameters with simple definitions that can be computed very easily. The notable exception is the locality number, which has also a simple string-based definition, but seems to be much more difficult to compute. In this sense, the locality number is the string parameter that covers best the treewidth of a string.

Independent of its algorithmic application described above, the locality is also an interesting structural parameter of words in its own right, the combinatorial properties of which deserve further investigation. In combinatorics on words, it is not unusual to define classes of words by imposing that a certain condition holds for a word while cancelling iteratively some letters of that word (e. g., correctly parenthesised words can be defined in this way). In this regard, k-local words are those words for which there exists an ordering of their letters, such that if we iteratively cancel all the occurrences of the letters according to this ordering, we have at each step at most k maximal blocks of cancelled letters. In this context, connections between 1- and 2-local words and correctly parenthesised words were established in [5].

For the algorithmic application, as well as for the further combinatorial investigation of the

concept of locality, the arguably most important problem is the computation of this parameter (and corresponding marking sequences). This paper is devoted to the investigation of this problem. A start in this direction has been done by [5], where it is shown that the k-locality of a word α can be checked in time $n^{\mathrm{O}(k)}$ (or, in parameterised complexity terms, the problem is in XP with respect to the parameter k). However, the focus of [5] is on using marking sequences for matching patterns with variables, while it has been left open if computing the locality number is a computationally hard problem.

Our Contribution – We first investigate the natural decision variant of the problem of computing the locality number, i. e., deciding, for a given word α and number k, whether $\mathsf{loc}(\alpha) \leq k$. We show that this problem is NP-complete, which identifies the computation of the locality number as a computationally hard problem and answers the main open question from [5]. A natural next step is to investigate the approximability of this problem, i. e., given a word, compute a marking sequence with small marking number (note that for algorithmic applications we require good marking sequences rather than the exact locality number). In this regard, we first demonstrate that the obvious greedy approach of choosing the next symbol to be marked according to some kind of greedy strategy (e. g., the symbol that yields the smallest number of marked blocks) does not yield an approximation algorithm for many natural choices of greedy strategies. We then show that, for a natural graph-representation for words, the width of an optimal path-decomposition of the graph representation of a word α ranges between $\mathsf{loc}(\alpha)$ and $2\,\mathsf{loc}(\alpha)$. This directly leads to a meta-theorem, which allows positive approximation results for computing marking sequences to be derived from the numerous results on the approximation of path-decompositions. More precisely, we can show that there is a factor-2 approximation algorithm for computing marking sequences that runs in fpt-time (with respect to the locality number), and that there is also a polynomial-time algorithm that approximates marking sequences within factor $\mathrm{O}(\sqrt{\log(\mathsf{opt})}\log(n))$ of the optimum. The questions whether there are polynomial-time constant-factor approximation algorithms for marking sequences or whether computing the locality number is fixed-parameter tractable (with respect to the bound on the locality number) are left open and assumed to be rather challenging. Note that the former is a longstanding open question for path-decompositions.

The connection between pathwidth and locality number is not strong enough to use this reduction for exact algorithms. Thus, we also discuss exact algorithms for computing the locality number (and a corresponding marking sequence) of a word. We show that if the number ℓ of different letters that occur in a word is logarithmic with respect to its length then we can exactly solve the problem in polynomial time. In the general case, considering ℓ as a parameter, an $\mathrm{O}^*(\ell!)$ fpt-algorithm is obvious, but we can also devise a more sophisticated fpt-algorithm with running-time $\mathrm{O}^*(2^\ell)$. Finally, it follows that the pattern matching problem with variables, parameterised by the locality number of the pattern, even when an optimal marking sequence is an explicit part of the input, is W[1]-hard.

References

[1] A. AMIR, I. NOR, Generalized function matching. *J. Discrete Algoritms* **5** (2007), 514–523.

[2] D. ANGLUIN, Finding patterns common to a set of strings. *J. Comput. Syst. Sci.* **21** (1980), 46–62.

[3] P. BARCELÓ, L. LIBKIN, A. W. LIN, P. T. WOOD, Expressive Languages for Path Queries over Graph-Structured Data. *ACM Trans. Database Syst.* **37** (2012).

[4] J. DAY, P. FLEISCHMANN, F. MANEA, D. NOWOTKA, M. L. SCHMID, On Matching Generalised Repetitive Patterns. In: *Proc. DLT*. LNCS, 2018. To appear.

[5] J. D. DAY, P. FLEISCHMANN, F. MANEA, D. NOWOTKA, Local Patterns. In: *Proc. FSTTCS*. LIPIcs 93, 2017, 24:1–24:14.

[6] T. ERLEBACH, P. ROSSMANITH, H. STADTHERR, A. STEGER, T. ZEUGMANN, Learning one-variable pattern languages very efficiently on average, in parallel, and by asking queries. *Theoret. Comput. Sci.* **261** (2001), 119–156.

[7] H. FERNAU, F. MANEA, R. MERCAŞ, M. L. SCHMID, Pattern Matching with Variables: Fast Algorithms and New Hardness Results. In: *Proc. STACS*. LIPIcs 30, 2015, 302–315.

[8] H. FERNAU, F. MANEA, R. MERCAŞ, M. L. SCHMID, Revisiting Shinohara's Algorithm for Computing Descriptive Patterns. *Theoretical Computer Science* **733** (2016), 44–54.

[9] H. FERNAU, M. L. SCHMID, Pattern matching with variables: A multivariate complexity analysis. *Inf. Comput.* **242** (2015), 287–305.

[10] H. FERNAU, M. L. SCHMID, Y. VILLANGER, On the Parameterised Complexity of String Morphism Problems. *Theory Comput. Syst.* **59** (2016) 1, 24–51.

[11] D. D. FREYDENBERGER, Extended Regular Expressions: Succinctness and Decidability. *Theory Comput. Syst.* **53** (2013), 159–193.

[12] J. E. F. FRIEDL, *Mastering Regular Expressions*. 3rd edition, O'Reilly, Sebastopol, CA, 2006.

[13] J. KARHUMÄKI, W. PLANDOWSKI, F. MIGNOSI, The Expressibility of Languages and Relations by Word Equations. *J. ACM* **47** (2000), 483–505.

[14] M. KEARNS, L. PITT, A polynomial-time algorithm for learning k-variable pattern languages from examples. In: *Proc. COLT*. 1989, 57–71.

[15] M. LOTHAIRE, *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

[16] Y. K. NG, T. SHINOHARA, Developments from enquiries into the learnability of the pattern languages from positive data. *Theoret. Comput. Sci.* **397** (2008), 150–165.

[17] D. REIDENBACH, Discontinuities in pattern inference. *Theoret. Comput. Sci.* **397** (2008), 166–193.

[18] D. REIDENBACH, M. L. SCHMID, Patterns with bounded Treewidth. *Inf. Comput.* **239** (2014), 87–99.

[19] T. SHINOHARA, Polynomial Time Inference of Pattern Languages and Its Application. In: *Proc. IBM MFCS*. 1982, 191–209.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 47–49.

On Ambiguity of Max-Plus Tree Automata

Erik Paul[A]

[A]Institute of Computer Science, Leipzig University, 04109 Leipzig, Germany
epaul@informatik.uni-leipzig.de

Abstract

A max-plus tree automaton is a finite-state machine which assigns real numbers to trees. In this talk, we consider decidability results for max-plus tree automata which are linked to the ambiguity of the respective automata. The ambiguity of a max-plus tree automaton is determined by the maximum number of valid runs there exist on each tree. For finitely ambiguous max-plus tree automata, we consider the equivalence, unambiguity, sequentiality, and finite sequentiality problems.

A max-plus automaton is a finite automaton with transition weights in the real numbers. To each word, it assigns the maximum weight of all accepting paths on the word, where the weight of a path is the sum of the path's transition weights. Max-plus automata and their min-plus counterparts are weighted automata [21, 20, 13, 3, 5] over the max-plus or min-plus semiring. Under varying names, max-plus and min-plus automata have been studied and employed many times in the literature. They can be used to determine the star height of a language [8], to decide the finite power property [22, 23] and to model certain timed discrete event systems [6, 7]. Additionally, they appear in the context of natural language processing [14].

For practical applications, the decidable properties of an automaton model are usually of great interest. Typical problems considered include the emptiness, universality, inclusion, equivalence, sequentiality, finite sequentiality, and unambiguity problems. We consider the last four of these problems for *finitely ambiguous* automata, which are automata in which the number of accepting paths for every word is bounded by a global constant. If there is at most one accepting path for every word, the automaton is called *unambiguous*. It is called *deterministic* or *sequential* if for each pair of a state and an input symbol, there is at most one valid transition into a next state. It is known [11] that finitely ambiguous max-plus automata are strictly more expressive than unambiguous max-plus automata, which in turn are strictly more expressive than deterministic max-plus automata.

Let us quickly recall the considered problems and the related results. The equivalence problem asks whether two automata are *equivalent*, which is the case if the weights assigned by them coincide on all words. In general, the equivalence problem is undecidable [12] for max-plus automata, but for finitely ambiguous max-plus automata it becomes decidable [24, 9]. The sequentiality problem asks whether for a given automaton, there exists an equivalent deterministic automaton. This was shown to be decidable by Mohri [14] for unambiguous max-plus automata. The unambiguity problem asks whether for a given automaton, there exists an equivalent unambiguous automaton. This problem is known to be decidable for finitely ambiguous

[A]This work was supported by Deutsche Forschungsgemeinschaft (DFG), Graduiertenkolleg 1763 (QuantLA).

and even *polynomially ambiguous* max-plus automata [11, 10]. In conjunction with Mohri's results, it follows that the sequentiality problem is decidable for these classes of automata as well. Finally, the finite sequentiality problem asks whether for a given automaton, there exist finitely many deterministic automata whose pointwise maximum is equivalent to the given automaton. Note that the class of automata which possess a finitely sequential representation is expressively incomparable to the class of unambiguous max-plus automata, and it is strictly less expressive than the class of finitely ambiguous automata [11]. It is know that the finite sequentiality problem is decidable for finitely ambiguous max-plus automata [1].

In this talk, we consider these four problems for finitely ambiguous max-plus tree automata, which are max-plus automata that operate on trees instead of words. In the form of *probabilistic context-free grammars*, max-plus tree automata are commonly employed in natural language processing [19]. Recently, we were able to show that the equivalence, unambiguity, and sequentiality problems are decidable for finitely ambiguous max-plus tree automata [18], and that the finite sequentiality problem is decidable for unambiguous max-plus tree automata.

Our approach to the decidability of the equivalence problem uses ideas from [9]. We reduce the equivalence problem to the same decidable problem, namely the decidability of the existence of an integer solution for a system of linear inequalities [15]. However, instead of the inductive argument used in [9], we employ Parikh's theorem [16, Theorem 2], an idea suggested by Mikołaj Bojańczyk. Our solution of the equivalence problem can be applied to weighted logics. In [17], a fragment of a weighted logic is shown to have the same expressive power as finitely ambiguous weighted tree automata. Over the max-plus semiring, equivalence is decidable for formulas of this fragment due to our results.

The decidability of the unambiguity problem employs ideas from [11]. Here, we show how the *dominance property* can be generalized to max-plus tree automata. To show the decidability of the sequentiality problem, we first combine results from [4] and [14] to show the decidability of this problem for unambiguous max-plus tree automata, and then combine this result with the decidability of the unambiguity problem.

For the finite sequentiality problem, we use ideas from [2] and generalize the *fork property* to the *tree fork property*. In comparison to the fork property, the proof of sufficiency of the tree fork property is much more involved due to the non-linear structure of trees.

References

[1] S. BALA, Which Finitely Ambiguous Automata Recognize Finitely Sequential Functions? In: K. CHATTERJEE, J. SGALL (eds.), *Proc. MFCS*. Springer, 2013, 86–97.

[2] S. BALA, A. KONIŃSKI, Unambiguous Automata Denoting Finitely Sequential Functions. In: A. DEDIU, C. MARTÍN-VIDE, B. TRUTHE (eds.), *Proc. LATA*. LNCS 7810, Springer, 2013, 104–115.

[3] J. BERSTEL, C. REUTENAUER, *Rational Series and Their Languages*. Springer, 1988.

[4] M. BÜCHSE, J. MAY, H. VOGLER, Determinization of Weighted Tree Automata Using Factorizations. *Journal of Automata, Languages and Combinatorics* **15** (2010) 3/4, 229–254.

[5] M. DROSTE, W. KUICH, H. VOGLER, *Handbook of Weighted Automata*. Springer, 2009.

[6] S. GAUBERT, Performance evaluation of (max,+) automata. *IEEE T. Automat. Contr.* **40** (1995) 12, 2014–2025.

[7] S. GAUBERT, J. MAIRESSE, Modeling and analysis of timed Petri nets using heaps of pieces. *IEEE T. Automat. Contr.* **44** (1999) 4, 683–697.

[8] K. HASHIGUCHI, Algorithms for Determining Relative Star Height and Star Height. *Inf. Comput.* **78** (1988) 2, 124–169.

[9] K. HASHIGUCHI, K. ISHIGURO, S. JIMBO, Decidability of The Equivalence Problem for Finitely Ambiguous Finance Automata. *IJAC* **12** (2002) 3, 445–461.

[10] D. KIRSTEN, S. LOMBARDY, Deciding Unambiguity and Sequentiality of Polynomially Ambiguous Min-Plus Automata. In: S. ALBERS, J. M. MARION (eds.), *Proc. STACS*. LIPIcs 3, LZI, 2009, 589–600.

[11] I. KLIMANN, S. LOMBARDY, J. MAIRESSE, C. PRIEUR, Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.* **327** (2004) 3, 349–373.

[12] D. KROB, The Equality Problem for Rational Series with Multiplicities in the tropical Semiring is Undecidable. *IJAC* **4** (1994) 3, 405–426.

[13] W. KUICH, A. SALOMAA, *Semirings, Automata, Languages*. Springer, 1986.

[14] M. MOHRI, Finite-State Transducers in Language and Speech Processing. *Comput. Linguist.* **23** (1997) 2, 269–311.

[15] G. L. NEMHAUSER, L. A. WOLSEY, *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988.

[16] R. J. PARIKH, On Context-Free Languages. *Journal of the Association for Computing Machinery* **13** (1966) 4, 570–581.

[17] E. PAUL, On Finite and Polynomial Ambiguity of Weighted Tree Automata. In: S. BRLEK, C. REUTENAUER (eds.), *Proc. DLT*. LNCS 9840, Springer, 2016, 368–379.

[18] E. PAUL, The Equivalence, Unambiguity and Sequentiality Problems of Finitely Ambiguous Max-Plus Tree Automata are Decidable. In: K. G. LARSEN, H. L. BODLAENDER, J.-F. RASKIN (eds.), *Proc. MFCS*. LIPIcs 83, LZI, 2017, 53:1–53:13.

[19] S. PETROV, Latent Variable Grammars for Natural Language Parsing. In: *Coarse-to-Fine Natural Language Processing*. chapter 2, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, 7–46.

[20] A. SALOMAA, M. SOITTOLA, *Automata-Theoretic Aspects of Formal Power Series*. Springer, 1978.

[21] M.-P. SCHÜTZENBERGER, On the definition of a family of automata. *Inform. Control* **4** (1961) 2-3, 245 – 270.

[22] I. SIMON, Limited Subsets of a Free Monoid. In: *Proc. FOCS*. IEEE Computer Society, 1978, 143–150.

[23] I. SIMON, Recognizable Sets with Multiplicities in the Tropical Semiring. In: M. CHYTIL, L. JANIGA, V. KOUBEK (eds.), *Proc. MFCS*. LNCS 324, Springer, 1988, 107–120.

[24] A. WEBER, Finite-Valued Distance Automata. *Theor. Comput. Sci.* **134** (1994) 1, 225–251.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 50–53.

Composition Closure of Linear Weighted Extended Top-Down Tree Transducers

Zoltán Fülöp[(A)] Andreas Maletti[(B)]

[(A)]Department of Foundations of Computer Science, University of Szeged
Árpád tér 2, H-6720 Szeged, Hungary
fulop@inf.u-szeged.hu

[(B)]Department of Mathematics and Computer Science, Universität Leipzig
PO Box 100 920, 04009 Leipzig, Germany
maletti@informatik.uni-leipzig.de

Abstract

Linear weighted extended top-down tree transducers with regular look-ahead and weights from a semiring are formal models of tree transformations that are used in syntax-based statistical machine translation. The composition hierarchies of some restricted versions of such weighted tree transducers (also without regular look-ahead) are considered. In particular, combinations of the restrictions of ε-freeness (all rules consume input), nondeletion, and strictness (all rules produce output) are considered. The composition hierarchy is shown to be finite for all but one ε-free variant of these weighted transducers over any commutative semiring.

Linear extended top-down tree transducers (l-xt) were introduced (under a different name) and investigated already in [1]. We present them in the framework of synchronous grammars [2] since this framework is better known in syntax-based statistical machine translation, where these transducers are applied, and since we utilize some results of [4, 11], where the same framework is used. Let us introduce l-xt informally. An l-xt M has a finite set of states and finitely many rules of the form $\langle \ell, q, r \rangle$, where q is a state and the left- and right-hand side ℓ and r are trees, which may also contain state-labeled leaves such that each state in r also occurs in ℓ. Linearity requires that each state occurs at most once both in ℓ and in r. In particular, in ε-rules the left-hand side ℓ is just a state, which means that applications of them consume no part of the input tree. Symmetrically, in non-strict rules the right-hand side r is just a state, which means that applications of them produce no output. The semantics of M is defined by means of synchronous rewriting using the derivation relation $\Rightarrow$. It is defined over sentential forms, which are triples (ξ, L, ζ) consisting of trees ξ and ζ with state-labeled leaves and a set L of links. A link is a pair (u, v) of positions pointing to occurrences of the same state in the trees ξ and ζ, respectively. A rule $\langle \ell, q, r \rangle$ can be applied to a sentential form (ξ, L, ζ) if there is a link $(u, v) \in L$ such that u and v point to an occurrence of the state q. In this case we write $(\xi, L, \zeta) \Rightarrow (\xi', L', \zeta')$,

[(A)]Supported by the NKFI grant no. K 108 448.

[(B)]Supported by the research training group 1763 "QuantLA" of the German Research Foundation (DFG).

where the sentential form (ξ', L', ζ') is obtained by replacing the linked occurrences of q in ξ and ζ by ℓ and r, respectively. In addition, L is updated to include links induced by occurrences of the same state in ℓ and r. The initial sentential form is $(q_0, \{(\varepsilon, \varepsilon)\}, q_0)$, in which q_0 is the initial state of M, and we apply derivation steps until no occurrences of linked states remain; i.e., until we reach a sentential form $(t, \emptyset, u)$. Any remaining (unlinked) state occurrence in the input tree t can then be replaced by an arbitrary tree. An instance of t is obtained by replacing all state occurrences and $I(t)$ is the set of all instances of t. The tree transformation induced by M consists of all pairs (t', u) such that $(q_0, \{(\varepsilon, \varepsilon)\}, q_0) \Rightarrow^* (t, \emptyset, u)$ and $t' \in I(t)$. In order to increase their expressive power, l-xt can be equipped with regular look-ahead, which restricts the instances $I(t)$ such that an unlinked occurrence of a state q can only be replaced by an element of a given regular tree language $c(q)$. Regular look-ahead for classical top-down tree transducers was invented in [3], and we abbreviate 'lx-t with regular look-ahead' by lx-t$^{\mathrm{R}}$. We refer to [4] for the exact definition of lx-t$^{\mathrm{R}}$.

The main computation model of our paper is the weighted lx-t$^{\mathrm{R}}$, abbreviated by lwx-t$^{\mathrm{R}}$, which is able to express quantitative properties of the tree transformations. In such an lwx-t$^{\mathrm{R}}$ a weight from a semiring $\mathbb{K}$ is associated to each rule, and these rule weights are multiplied in a derivation. Provided that several derivations exist, then these derivation weights are summed up. In this manner, an lwx-t$^{\mathrm{R}}$ M induces a weighted tree transformation, which assigns a weight $\|M\|_{\mathbb{K}}(t, u) \in \mathbb{K}$ to each pair (t, u) of trees. It turned out that both lwx-t and lwx-t$^{\mathrm{R}}$ over the probabilistic semiring can serve as formal models of tree transformations which are used in syntax-based statistical machine translation [7, 6]. Their expressive power and other properties are well studied [13, 5, 9, 10, 8, 11].

In this paper we focus on the composition closure of l-wxt and l-wxt$^{\mathrm{R}}$ without ε-rules ($\not\varepsilon$l-wxt and $\not\varepsilon$l-wxt$^{\mathrm{R}}$, respectively) and some of their restricted subclasses (in order to guarantee that all summations remain finite). Our motivation is that complex systems are often specified in terms of compositions of simpler tree transformations [14], which manage only a part of the overall transformation, because they are easier to develop, train, and maintain [7]. More precisely, let $\mathcal{C}$ be a class of weighted tree transformations (e.g. the class of all weighted tree transformations induced by $\not\varepsilon$l-wxt$^{\mathrm{R}}$ obeying some further restrictions). The composition hierarchy of $\mathcal{C}$ is $\mathcal{C} \subseteq \mathcal{C}^2 \subseteq \mathcal{C}^3 \subseteq \cdots$, where $\mathcal{C}^n$ denotes the n-fold composition of $\mathcal{C}$. It is either infinite (i.e., $\mathcal{C}^n \subsetneq \mathcal{C}^{n+1}$ for all n) or finite (i.e., $\mathcal{C}^n = \mathcal{C}^{n+1}$ for some n). In the latter case, the minimal such n is called the collapse level and is interesting since all compositions can be reduced to this length.

Now let us introduce the classes, for which we will investigate the composition hierarchy. The additional standard restrictions we consider are strictness ('s'), which requires that the right-hand side r is not a single state, and nondeletion ('n'), which means that each state in the left-hand side ℓ occurs also in the right-hand side r, in both cases for each rule $\langle \ell, q, r \rangle$ of the $\not\varepsilon$l-wxt$^{\mathrm{R}}$. Thus, for instance $\not\varepsilon$sl-wxt$^{\mathrm{R}}$ abbreviates the expression 'strict $\not\varepsilon$l-wxt$^{\mathrm{R}}$'. The class of all weighted tree transformations induced by certain kind of $\not\varepsilon$l-wxt$^{\mathrm{R}}$ is denoted by typesetter letters so for instance $\not\varepsilon\mathtt{sl\text{-}WXT}^{\mathrm{R}}(\mathbb{K})$ stands for the set of all weighted tree transformations computable by $\not\varepsilon$sl-wxt$^{\mathrm{R}}$ over the semiring $\mathbb{K}$. We consider the composition hierarchies of the classes $\not\varepsilon\mathtt{nsl\text{-}WXT}(\mathbb{K})$, which is also investigated in [12], and $\not\varepsilon\mathtt{sl\text{-}WXT}(\mathbb{K})$, $\not\varepsilon\mathtt{sl\text{-}WXT}^{\mathrm{R}}(\mathbb{K})$, $\not\varepsilon\mathtt{l\text{-}WXT}^{\mathrm{R}}(\mathbb{K})$, and $\not\varepsilon\mathtt{l\text{-}WXT}(\mathbb{K})$. As main result we show that the composition hierarchies of these classes collapse at levels 2, 2, 2, 3, and 4, respectively, for an arbitrary commutative semiring $\mathbb{K}$. We achieve our results by lifting the results [1, Theorem 6.2] and [4, Theorems 26, 31, 33, 34],

where it is shown that the corresponding hierarchies in the unweighted cases collapse at the same levels. To this end, we decompose an $\not\varepsilon$-l-wxt^R into a weighted relabeling that handles all the weights and the nondeterminism, and a BOOLEAN and functional $\not\varepsilon$-l-wxt^R that is also unambiguous. Moreover, we show that we can compose any such relabeling to the right of any l-wxt^R. These two constructions together will allow us to take all $\not\varepsilon$-l-wxt^R in a composition chain into a particularly simple normal form that is very close to the unweighted case. After some additional technical tailoring, we can utilize the mentioned results [1, 4] and lift them to the corresponding weighted tree transducers over any commutative semiring. We note that in [4] the composition hierarchies of some further classes induced by l-wxt^R with ε-rules are shown to be infinite.

References

[1] A. ARNOLD, M. DAUCHET, Morphismes et Bimorphismes d'Arbres. *Theoret. Comput. Sci.* **20** (1982) 1, 33–93.

[2] D. CHIANG, An Introduction to Synchronous Grammars. In: N. CALZOLARI, C. CARDIE, P. ISABELLE (eds.), *Proc. 44th Ann. Meeting ACL*. Association for Computational Linguistics, 2006. Part of a tutorial given with Kevin Knight.

[3] J. ENGELFRIET, Top-down Tree Transducers with Regular Look-ahead. *Math. Systems Theory* **10** (1977) 1, 289–303.

[4] J. ENGELFRIET, Z. FÜLÖP, A. MALETTI, Composition Closure of Linear Extended Top-down Tree Transducers. *Theory Comput. Systems* **60** (2017) 2, 129–171.

[5] Z. FÜLÖP, A. MALETTI, H. VOGLER, Weighted Extended Tree Transducers. *Fundam. Inform.* **111** (2011) 2, 163–202.

[6] J. GRAEHL, K. KNIGHT, J. MAY, Training Tree Transducers. *Comput. Linguist.* **34** (2008) 3, 391–427.

[7] K. KNIGHT, J. GRAEHL, An Overview of Probabilistic Tree Transducers for Natural Language Processing. In: A. F. GELBUKH (ed.), *Proc. 6th Int. Conf. CICLing*. LNCS 3406, Springer, 2005, 1–24.

[8] A. LAGOUTTE, A. MALETTI, Survey: Weighted extended top-down tree transducers — Part III: Composition. In: W. KUICH, G. RAHONIS (eds.), *Proc. Workshop Algebraic Foundations in Computer Science*. LNCS 7020, Springer, 2011, 272–308.

[9] A. MALETTI, Survey: Weighted Extended Top-down Tree Transducers — Part I: Basics and Expressive Power. *Acta Cybernet.* **20** (2011) 2, 223–250.

[10] A. MALETTI, Survey: Weighted Extended Top-down Tree Transducers — Part II: Application in Machine Translation. *Fundam. Inform.* **112** (2011) 2–3, 239–261.

[11] A. MALETTI, The power of weighted regularity-preserving multi bottom-up tree transducers. *Int. J. Found. Comput. Sci.* **26** (2015) 7, 987–1005.

[12] A. MALETTI, Compositions of Tree-to-Tree Statistical Machine Translation Models. In: S. BRLEK, C. REUTENAUER (eds.), *Proc. 20th Int. Conf. DLT*. LNCS 9840, Springer, 2016, 293–305.

[13] A. MALETTI, J. GRAEHL, M. HOPKINS, K. KNIGHT, The Power of Extended Top-down Tree Transducers. *SIAM J. Comput.* **39** (2009) 2, 410–430.

[14] J. MAY, K. KNIGHT, H. VOGLER, Efficient inference through cascades of weighted tree transducers. In: J. HAJIČ, S. CARBERRY, S. CLARK, J. NIVRE (eds.), *Proc. 48th Ann. Meeting ACL*. Association for Computational Linguistics, 2010, 1058–1066.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 54–57.

Weighted Operator Precedence Languages

Manfred Droste[(B)] Stefan Dück[(A,B)] Dino Mandrioli[(C)]
Matteo Pradella[(C,D)]

[(B)]Institute of Computer Science, Leipzig University, D-04109 Leipzig, Germany
{droste,dueck}@informatik.uni-leipzig.de

[(C)]Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italy
{dino.mandrioli,matteo.pradella}@polimi.it

[(D)]IEIIT, Consiglio Nazionale delle Ricerche, via Ponzio 34/5, 20133 Milano, Italy

Extended Abstract

In the long history of formal languages the family of regular languages (RL) has always played a major role: thanks to its simplicity and naturalness, it enjoys many positive mathematical properties which have been thoroughly exploited in disparate practical applications; among them, those of main interest in this paper are the following:

- RL have been characterized in terms of various mathematical logics. Originally, Büchi, Elgot, and Trakhtenbrot [3, 12, 26] independently developed a monadic second order (MSO) logic defining exactly the RL family. This work has been followed by many further results; in particular those that exploited weaker but simpler logics such as first-order, propositional, and temporal ones culminated in the breakthrough of model checking to support automatic verification [20, 13, 4].
- Weighted RL have been introduced by Schützenberger in [24]: by assigning a weight in a suitable algebra to each language word, we may specify several attributes of the word, e.g., relevance, probability, etc. Much research then followed and extended Schützenberger's original work in various directions, cf. the books [11, 23, 15, 2, 9].

Unfortunately, all families with greater expressive power than RL –typically context-free languages (CFL), which are the most widely used family in practical applications– pay a price in terms of algebraic and logic properties and, consequently, of possible tools supporting their automatic analysis. For instance, for CFL, the containment problem is undecidable.

What was not possible for general CFL, however, has been possible for important subclasses of this family, which together we call *structured CFL*. Informally, by this term we denote those CFL where the syntactic tree-structure of their words is immediately "visible" in the words themselves. Two first equivalent examples of such families are parenthesis languages [19], which are generated by grammars whose right hand sides are enclosed within pairs of

[(A)]This work was supported by Deutsche Forschungsgemeinschaft (DFG) Graduiertenkolleg 1763 (QuantLA).

parentheses, and tree-automata [25], which generalize finite state machines (FSM) from the recognition of linear strings to tree-like structures. Among the many variations of parenthesis languages the recent family of *input-driven languages* [21, 27], alias *visibly pushdown languages (VPL)* [1], has received much attention in recent literature. For most of these structured CFL, including VPL, the relevant algebraic properties of RL still hold [1]. One of the most noticeable results has been a characterization of VPL in terms of a MSO logic that is a natural extension of Büchi's original one for RL [16, 1].

This fact has suggested to extend the investigation of weighted RL to various cases of structured languages. The result of such a fertile approach is a rich collection of *weighted logics*, first studied by Droste and Gastin [8], associated with *weighted tree automata* [10] and weighted extensions of VPA (the automata recognizing VPL) [18].

In an originally unrelated way *operator precedence languages (OPL)* have been defined and studied in two phases temporally separated by four decades. In his seminal work [14] Floyd was inspired by the precedence of multiplicative operations over additive ones in the execution of arithmetic expressions. He extended such a relation to the whole input alphabet in such a way that it could drive a deterministic parsing algorithm that builds the syntax tree of any word that reflects the word's semantics.

OPL do not cover all deterministic CFL, but they enjoy a distinguishing property, not possessed by general deterministic CFL, which we can intuitively describe as "*OPL are input driven but not visible*". They can be claimed as *input-driven* since the parsing actions on their words –whether to push or pop– depend exclusively on the input alphabet and on the relation defined thereon, but their structure is *not visible* in their words: e.g, they can include unparenthesized expressions where the precedence of multiplicative operators over additive ones is explicit in the syntax trees but hidden in their frontiers. Furthermore, unlike other structured CFL, OPL include deterministic CFL that are not real-time [17].

This recent remark suggested to resume their investigation systematically at the light of the recent technological advances and related challenges. Such a renewed investigation led to prove their closure under all major language operations [5] and to characterize them, besides Floyd's original grammars, in terms of an appropriate class of pushdown automata (OPA) and in terms of a MSO logic which is a fairly natural but not trivial extension of the previous ones defined to characterize RL and VPL [17]. Thus, OPL enjoy the same nice properties of RL and many structured CFL but considerably extend their applicability by breaking the barrier of visibility and real-time push-down recognition.

In this paper we join the two research fields above, namely we introduce *weighted OPL* and show that they are able to model system behaviors that cannot be specified by means of less powerful weighted formalisms such as weighted VPL. For instance, one might be interested in the behavior of a system which handles calls and returns but is subject to some emergency interrupts. Then it is important to evaluate how critically the occurrences of interrupts affect the normal system behavior, e.g., by counting the number of pending calls that have been preempted by an interrupt. As another example we consider a system logging all hierarchical calls and returns over words where this structural information is hidden. Depending on changing exterior factors like energy level, such a system could decide to log the above information in a selective way.

Our main contributions are the following.

- The model of *weighted OPA*, which have semiring weights at their transitions, significantly increases the descriptive power of previous weighted extensions of VPA, and has desired closure and robustness properties.
- For arbitrary semirings, there is a relevant difference in the expressive power of the model depending on whether it permits assigning weights to pop transitions or not. This is due to the fact that OPL may be non-real-time and therefore OPA may execute several pop moves without advancing their reading heads. For commutative semirings, however, weights on pop transitions do not increase the expressive power of the automata.
- An extension of the classical result of Nivat [22] to weighted OPL. This robustness result shows that the behaviors of weighted OPA without weights at pop transitions are exactly those that can be constructed from weighted OPA with only one state, intersected with OPL, and applying projections which preserve the structural information.
- A weighted MSO logic and, for arbitrary semirings, a Büchi-Elgot-Trakhtenbrot-Theorem proving its expressive equivalence to weighted OPA without weights at pop transitions. As a corollary, for commutative semirings this weighted logic is equivalent to weighted OPA including weights at pop transitions.

Various possibilities arise for future research concerning theory and applications of our new model. Additionally to the published results in [6], the full version of this paper [7] provides all omitted technicalities and more explanatory comments and examples.

Literatur

[1] R. ALUR, P. MADHUSUDAN, Adding Nesting Structure to Words. *J. ACM* **56** (2009) 3, 16:1–16:43.

[2] J. BERSTEL, C. REUTENAUER, *Rational Series and Their Languages*. EATCS Monographs in Theoretical Computer Science 12, Springer, 1988.

[3] J. R. BÜCHI, Weak second-order arithmetic and finite automata. *Z. Math. Logik und Grundlagen Math.* **6** (1960), 66–92.

[4] C. CHOFFRUT, A. MALCHER, C. MEREGHETTI, B. PALANO, First-order logics: some characterizations and closure properties. *Acta Inf.* **49** (2012) 4, 225–248.

[5] S. CRESPI REGHIZZI, D. MANDRIOLI, Operator Precedence and the Visibly Pushdown Property. *J. Comput. Syst. Sci.* **78** (2012) 6, 1837–1867.

[6] M. DROSTE, S. DÜCK, D. MANDRIOLI, M. PRADELLA, Weighted Operator Precedence Languages. In: K. G. LARSEN, H. L. BODLAENDER, J. RASKIN (eds.), *Mathematical Foundations of Computer Science, MFCS 2015*. LIPIcs 83, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 31:1–31:15.

[7] M. DROSTE, S. DÜCK, D. MANDRIOLI, M. PRADELLA, Weighted Operator Precedence Languages. *CoRR* **abs/1702.04597** (2017).

[8] M. DROSTE, P. GASTIN, Weighted automata and weighted logics. *Theor. Comput. Sci.* **380** (2007) 1-2, 69–86. Extended abstract in ICALP 2005.

[9] M. DROSTE, W. KUICH, H. VOGLER (eds.), *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science, Springer, 2009.

[10] M. DROSTE, H. VOGLER, Weighted tree automata and weighted logics. *Theor. Comput. Sci.* **366** (2006) 3, 228–247.

[11] S. EILENBERG, *Automata, Languages, and Machines*. Pure and Applied Mathematics 59-A, Academic Press, 1974.

[12] C. C. ELGOT, Decision Problems of Finite Automata Design and Related Arithmetics. *Trans. Am. Math. Soc.* **98** (1961) 1, 21–52.

[13] E. A. EMERSON, Temporal and Modal Logic. In: *Handbook of Theoretical Computer Science, Volume B*. MIT Press, 1990, 995–1072.

[14] R. W. FLOYD, Syntactic Analysis and Operator Precedence. *J. ACM* **10** (1963) 3, 316–333.

[15] W. KUICH, A. SALOMAA, *Semirings, Automata, Languages*. EATCS Monographs in Theoretical Computer Science 6, Springer, 1986.

[16] C. LAUTEMANN, T. SCHWENTICK, D. THÉRIEN, Logics For Context-Free Languages. In: L. PACHOLSKI, J. TIURYN (eds.), *Computer Science Logic, Selected Papers*. LNCS 933, Springer, 1994, 205–216.

[17] V. LONATI, D. MANDRIOLI, F. PANELLA, M. PRADELLA, Operator Precedence Languages: Their Automata-Theoretic and Logic Characterization. *SIAM J. Comput.* **44** (2015) 4, 1026–1088.

[18] C. MATHISSEN, Weighted Logics for Nested Words and Algebraic Formal Power Series. *Logical Methods in Computer Science* **6** (2010) 1. Selected papers of ICALP 2008.

[19] R. MCNAUGHTON, Parenthesis Grammars. *J. ACM* **14** (1967) 3, 490–500.

[20] R. MCNAUGHTON, S. PAPERT, *Counter-free Automata*. MIT Press, Cambridge, USA, 1971.

[21] K. MEHLHORN, Pebbling Mountain Ranges and its Application of DCFL-Recognition. In: *Automata, Languages and Programming, ICALP 1980*. LNCS 85, 1980, 422–435.

[22] M. NIVAT, Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier* **18** (1968), 339–455.

[23] A. SALOMAA, M. SOITTOLA, *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science, Springer, 1978.

[24] M. P. SCHÜTZENBERGER, On the Definition of a Family of Automata. *Inf. Control* **4** (1961) 2-3, 245–270.

[25] J. THATCHER, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journ. of Comp. and Syst.Sc.* **1** (1967), 317–322.

[26] B. A. TRAKHTENBROT, Finite automata and logic of monadic predicates (in Russian). *Doklady Akademii Nauk SSR* **140** (1961), 326–329.

[27] B. VON BRAUNMÜHL, R. VERBEEK, Input-driven languages are recognized in log n space. In: *Proceedings of the Symposium on Fundamentals of Computation Theory*. LNCS 158, Springer, 1983, 40–51.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 58–61.

Bimonoid Weighted Linear Dynamic Logic

Gustav Grabolle[(A)]

[(A)]Gustav Grabolle, Universität Leipzig, Institut für Informatik, PF 100920, D-04009 Leipzig
grabolle@informatik.uni-leipzig.de

Abstract

Linear dynamic logic (LDL) over finite traces was introduced by De Giacomo and Vardi at IJCAI in 2013. LDL combines the syntax of linear temporal logic (LTL) and propositional dynamic logic (PDL), but maintains the intuitiveness of LTL. Satisfiability and validity of LDL formulas is PSPACE-complete. Furthermore, LDL characterizes the complete family of recognizable languages. Very recently, a weighted LDL version with weights in semirings was introduced by Droste and Rahonis. This logic can be used to describe quantitative properties using the syntax of LDL and it has the same expressive power as semiring weighted automata. Here, we want to introduce a weighted LDL for bimonoids to describe quantitative properties over non-distributive structures such as lattices. This logic is a modified version of weighted LDL for semirings and is as expressive as weighted automata if weights are taken from bi-locally finite strong bimonoids.

1. Introduction

The connection between logics and automata is utilized in fields such as program verification or on-the-fly model checking. One of the most prominent logics used for this purpose is linear temporal logic (LTL), which was introduced by Pnueli [9]. LTL is intuitive and its model checking is efficient. Nonetheless, it is less expressive than finite automata and monadic second order logic (MSO). A proposal to close the gap in expressiveness while preserving efficient translations from formulas to automata was given by De Giacomo and Vardi [2] via their linear dynamic logic (LDL). De Giacomo and Vardi showed that LDL and finite automata are equally expressive and that LDL can be translated into finite automata within doubly-exponential time.

Since their introduction weighted finite automata, a generalization of finite automata, became a flourishing field of computer science. They deliver solutions to many theoretical problems and are motivated by applications in fields like language processing and image compression [1, 3]. Instead of deciding whether a computation is successful or not, they return a value of an underlying weight structure. These values are used to model concepts like cost, probability, capacity, or quantity. Weight structures include, but are not limited to fields and semirings [10, 3].

Weighted logics are as important to weighted automata, as non-weighted logics are to automata. Very recently, weighted LDL with weights in semirings was introduced by Droste and Rahonis [4]. However, the operations in semirings are distributive. To model non-distributive

structures we use non-distributive weight structures such as lattices. Lattices have been investigated for a long time and were often used as weight structures for logics like the three valued logic L_3 of Łukasiewicz.

Weighted finite automata over non-distributive weight structures have been investigated by Droste, Stüber, and Vogler [5], who defined weighted finite automata over strong bimonoids, a generalization of semirings and lattices. Multi valued MSO was introduced by Droste and Vogler [6]. Moreover, lattice valued LTL has been studied by Kupferman and Lustig [8].

In this work, we unite the concept of LDL with weighted finite automata over strong bimonoids, by defining multivalued LDL. Our main contributions are:

- We define weighted LDL over strong bimonoids. To this purpose, we adjust the semantics from [4] using a non-recursive iteration operation to fit our non-distributive setting.
- In our main result we prove that weighted LDL and weighted finite automata over bi-locally finite strong bimonoids are equally expressive. We do this by automaton constructions for the operations used in the semantics of weighted LDL.
- We construct an example of a quantitative language over lattices, and we use it to prove that recognizable quantitative languages over bi-locally finite strong bimonoids are not closed under the recursive iteration operation used in [4, 3].

Furthermore, we investigate the behavior of recognizable step functions over lattices under rational operations.

2. Bimonoid weighted quantitative languages

A *monoid* is an algebraic structure $(M, \circ, e)$ where $\circ$ is a binary, associative operation, $e \in M$, and $m \circ e = m = e \circ m$ for all $m \in M$. A monoid is *commutative* if $+$ is a commutative operation.

A *bimonoid* is an algebraic structure $(B, +, \cdot, 0, 1)$ (for short: B) where $(M, +, 0)$ and $(M, \cdot, 1)$ are monoids, and $0 \neq 1$. We call B a *strong* bimonoid if $(B, +, 0)$ is a commutative monoid, and $0 \cdot b = b \cdot 0 = 0$ for all $b \in B$. We say B is *additively locally finite* (*multiplicatively locally finite*), if for every finite $S \subseteq B$ the smallest submonoid of $(B, +, 0)$ (of $(B, \cdot, 1)$ respectively) containing S is finite. A bimonoid B is said to be *bi-locally finite* if it is additively locally finite and multiplicatively locally finite. A bimonoid B is called *locally finite* if for every finite $S \subseteq B$ the smallest subbimonoid of B containing S is finite.

Quantitative languages are the quantitative counterpart to languages (cf. [7, 3]).

Let $(B, +, \cdot, 0, 1)$ be a bimonoid. A *B-valued quantitative language* over Σ is a map $s : \Sigma^* \to B$. The set of all B-valued quantitative languages over Σ is denoted by $B\langle\langle\Sigma^*\rangle\rangle$.

For any language $L \subseteq \Sigma^*$, let $\mathbb{1}_L$ denote the *characteristic quantitative language*. Let $k \in B$ a bimonoid element, $s_1, s_2 \in B\langle\langle\Sigma^*\rangle\rangle$ quantitative languages and $n \in \mathbb{N}_{\geq 1}$ an integer. We will use the usual (cf. [6]) definitions for *scalar product* ($k \cdot s_1 = ks_1$), *sum* ($s_1 + s_2$), *Hadamard product* ($s_1 \odot s_2$), and *Cauchy product* ($s_1 \cdot s_2$). Moreover, if s_1 is proper, we define the *n-th iteration* (s_1^{+n}), the *recursive n-th iteration* ($s_1^{\boxplus n}$), the *iteration* (s_1^{+}), and the *recursive iteration* ($s_1^{\boxplus}$) by

$$(s_1^{+n}, w) = \sum_{w=u_1 \dots u_n} \prod_{1 \le i \le n} (s_1, u_i) \ , \qquad (s_1^{\boxplus n}, w) = \begin{cases} s_1 & \text{if } n = 1 \\ s_1^{\boxplus n-1} \cdot s_1 & \text{otherwise,} \end{cases}$$
$$(s_1^{+}, w) = \sum_{n=1}^{|w|} (s_1^{+n}, w) \ , \qquad (s_1^{\boxplus}, w) = \sum_{n=1}^{|w|} (s_1^{\boxplus n}, w)$$

for all $w \in \Sigma^*$.

A quantitative language is *recognizable* if it is recognized by a weighted finite automaton. For a definition of bimonoid weighted automata we refer the reader to [5]. It was shown in [6] that recognizable quantitative languages over bi-locally finite strong bimonoids are closed under scalar product, sum , Hadamard product, Cauchy product, n-th iteration, and iteration. However, semiring weighted LDL uses the recursive iteration in its semantics and while iteration and recursive iteration coincide over distributive structures, our following result showed that this is not the case for bimonoids.

Lemma 2.1 *Recognizable quantitative languages over bi-locally finite strong bimonoids are not closed under recursive iteration.*

To replace this operation we introduced the *iterative combination* $(s_1 * s_2)$. If s_1 is proper, it is defined by

$$(s_1 * s_2, w) = \sum_{n=1}^{|w|+1} \Big(\sum_{w=u_1 \dots u_n} \big(\prod_{1 \le i \le n-1} (s_1, u_i) \big) \cdot (s_2, u_n) \Big)$$

for all $w \in \Sigma^*$. We proved the closure of recognizable quantitative languages over bi-locally finite strong bimonoids under the iterative combination.

3. Multivalued linear dynamic logic

For the definition of weighted LDL, we follow the definitions of [4]. However, we change the definitions of the semantics of $\langle \rho^{\oplus} \rangle \psi$. Due to Lemma 2.1, this modification was necessary.

Let $(B, +, \cdot, 0, 1)$ be a bimonoid. We define the syntax of *B-weighted LDL formulas* (for short: *B-LDL formulas*) φ over Σ by the grammar

$$\varphi ::= k \mid \psi \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \langle \rho \rangle \varphi$$
$$\rho ::= \phi \mid \varphi? \mid \rho \oplus \rho \mid \rho \cdot \rho \mid \rho^{\oplus}$$

where $k \in B$ is a constant, ψ an LDL formula (cf. [2]) over Σ, and ϕ a propositional formula over Σ. Let φ be a B-LDL formula over Σ and $w \in \Sigma^*$ a word. We define the semantics $[\![\varphi]\!] \in B\langle\!\langle \Sigma^* \rangle\!\rangle'$ of φ inductively on the structure of φ by

$$([\![k]\!], w) = k \ ,$$
$$([\![\psi]\!], w) = (\mathbb{1}_{L(\psi)}, w) \ ,$$
$$([\![\varphi_1 \oplus \varphi_2]\!], w) = ([\![\varphi_1]\!] + [\![\varphi_2]\!], w) \ ,$$
$$([\![\varphi_1 \otimes \varphi_2]\!], w) = ([\![\varphi_1]\!] \odot [\![\varphi_2]\!], w) \ .$$

Now, we define the semantics of $\langle \rho \rangle \varphi$ inductively on the structure of ρ by

$$([\![\langle \phi \rangle \varphi]\!], w) = (\mathbb{1}_{L(\phi)}, w) \cdot ([\![\varphi]\!], w_{\ge 1}) \ ,$$
$$([\![\langle \varphi_1? \rangle \varphi]\!], w) = ([\![\varphi_1]\!] \odot [\![\varphi]\!], w) \ ,$$
$$([\![\langle \rho_1 \oplus \rho_2 \rangle \varphi]\!], w) = ([\![\langle \rho_1 \rangle \varphi]\!] + [\![\langle \rho_2 \rangle \varphi]\!], w) \ ,$$
$$([\![\langle \rho_1 \cdot \rho_2 \rangle \varphi]\!], w) = ([\![\langle \rho_1 \rangle \top]\!] \cdot [\![\langle \rho_2 \rangle \varphi]\!], w) \ ,$$
$$([\![\langle \rho_1^{\oplus} \rangle \varphi]\!], w) = ([\![\langle \rho_1 \rangle \top]\!] * [\![\langle \rho_1 \rangle \varphi]\!], w) \ .$$

A quantitative language s is *B-LDL definable* iff a B-LDL formula φ exists, such that $[\![\varphi]\!] = s$.

In our main result, we showed that B-LDL characterizes the class of recognizable bimonoid weighted quantitative languages over bi-locally finite strong bimonoids.

Theorem 3.1 *Let $(B, +, \cdot, 0, 1)$ be a bi-locally finite strong bimonoid and $s \in B\langle\langle\Sigma^*\rangle\rangle$ a quantitative language. Then, s is B-LDL definable iff s is recognizable by an B-weighted finite automaton.*

References

[1] K. CHATTERJEE, L. DOYEN, T. A. HENZINGER, Quantitative languages. In: M. KAMINSKI, S. MARTINI (eds.), *Computer Science Logic*. Springer, Berlin, Heidelberg, 2008, 385–400.
https://doi.org/10.1007/978-3-540-87531-4_28

[2] G. DE GIACOMO, M. Y. VARDI, Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, 2013, 854–860.
http://dl.acm.org/citation.cfm?id=2540128.2540252

[3] M. DROSTE, W. KUICH, H. VOGLER, *Handbook of Weighted Automata*. 1st edition, Springer, 2009.

[4] M. DROSTE, G. RAHONIS, Weighted linear dynamic logic. In: *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification*. 226, EPTCS, 2016, 149–163.
http://dx.doi.org/10.4204/EPTCS.226.11

[5] M. DROSTE, T. STÜBER, H. VOGLER, Weighted finite automata over strong bimonoids. *Inform. Sci.* **180** (2010) 1, 156 – 166. Special Issue on Collective Intelligence.
http://www.sciencedirect.com/science/article/pii/S0020025509003867

[6] M. DROSTE, H. VOGLER, Weighted automata and multi-valued logics over arbitrary bounded lattices. *Theoret. Comput. Sci.* **418** (2012), 14 – 36.
http://www.sciencedirect.com/science/article/pii/S0304397511009157

[7] W. KUICH, *Semirings and Formal Power Series: Their Relevance to Formal Languages and Automata*, chapter 9. Springer, New York, NY, USA, 1997, 609–677.
http://dl.acm.org/citation.cfm?id=267846.267855

[8] O. KUPFERMAN, Y. LUSTIG, Lattice automata. In: B. COOK, A. PODELSKI (eds.), *Verification, Model Checking, and Abstract Interpretation*. Springer, Berlin, Heidelberg, 2007, 199–213.
https://doi.org/10.1007/978-3-540-69738-1_14

[9] A. PNUELI, The temporal logic of programs. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 1977, 46–57.

[10] M. SCHÜTZENBERGER, On the definition of a family of automata. *Inform. and Control* **4** (1961) 2, 245 – 270.
http://www.sciencedirect.com/science/article/pii/S001999586180020X

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 62–65.

ω-Pushdown Automata

Manfred Droste[(A)] Sven Dziadek[(A,B)] Werner Kuich[(C)]

[(A)]Universität Leipzig, Institut für Informatik, Germany
{droste,dziadek}@informatik.uni-leipzig.de

[(C)]Technische Universität Wien, Institut für Diskrete Mathematik und Geometrie, Austria
werner.kuich@tuwien.ac.at

Abstract

Lautemann, Schwentick, Thérien presented an equivalent logical formalism for pushdown automata over finite words. We present here an equivalent logic for ω-pushdown automata. This automaton model has access to a stack and models context-free properties of infinite words. To prove the equivalence in expressive power of automaton and logic, some kind of normal form for the automaton model, the simple ω-pushdown automaton, is introduced. We prove that simple ω-pushdown automata can recognize all ω-context-free languages; this result may be of independent interest. The second half of the equivalence proof uses similar results recently developed for nested ω-words.

1. Introduction

An extension of the standard model of pushdown automata is given by ω-pushdown automata which can handle infinite words and therefore model infinite processes. In 1977, Cohen, Gold [3] developed fundamental results on ω-pushdown automata and ω-context-free grammars including various recognition modes such as Büchi and Muller acceptance.

Here, we present a new type of ω-pushdown automata, the simple ω-pushdown automaton. This new model does not allow ϵ-transitions and has a very restricted access to the stack. Only three different stack commands are available: the automaton can either pop the topmost symbol, push one symbol or ignore the stack for that transition. We prove that simple ω-pushdown automata still recognize all ω-context-free languages.

Another representation of language classes additionally to automata and grammars are logics. For finite automata, this equivalence dates back to Büchi-Elgot and Trakhtenbrot [2, 4]. Logic provides an intuitive way to describe properties of a system and thus plays an important role in verification. The translation from logics to automata allows to benefit from the algorithmic properties provided by these automata. This equivalence has therefore been extended to numerous automata models. For context-free languages, Lautemann, Schwentick, Thérien [5] presented an equivalent logic over finite words. Here, we extend their results to infinite words to allow the study of reactive systems.

[(B)]Supported by DFG Research Training Group 1763 (QuantLA)

2. Simple ω-Pushdown Automata

We propose a new automaton model, the simple ω-pushdown automaton; this automaton model is the key for the equivalence proof in Section 3. We do not allow ϵ-transitions. Additionally, we restrict the access to the stack to only allow either to keep the stack unaltered, to push one symbol or to pop one symbol.

Let $\mathcal{S}(\Gamma) = (\{\downarrow\} \times \Gamma) \cup \{\#\} \cup (\{\uparrow\} \times \Gamma)$ be the set of *stack commands*.

Definition 2.1 *A* simple ω-pushdown automaton *or ωSPDA denotes a tuple* $M = (Q, \Sigma, \Gamma, T, q_0, F)$ *where*

- Q *is a finite set of states,*
- Σ *and* Γ *are finite input and stack alphabets, respectively,*
- $T \subseteq Q \times \Sigma \times Q \times \mathcal{S}(\Gamma)$ *is a finite set of transitions,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is a finite set of (Büchi-accepting) final states.*

Note that the definition of T does not allow ϵ-transitions.

Let $t = (q, \sigma, q', (\downarrow, A))$; then we call t a *push transitions* and it pushes symbol A onto the stack. Let $t = (q, \sigma, q', \#)$; then we call t an *internal transition* and it leaves the stack unaltered. Finally, let $t = (q, \sigma, q', (\uparrow, A))$; then we call t a *pop transition* which pops one symbol A from the stack.

Runs of the automaton start with an empty stack and are defined to be successful if at least one state in F occurs infinitely often.

For an ωSPDA $M = (Q, \Sigma, \Gamma, T, q_0, F)$, the language *accepted* by M is denoted by $\mathcal{L}(M) = \{w \in \Sigma^\omega \mid \exists \text{ successful run of } M \text{ on } w\}$. A language $L \subseteq \Sigma^\omega$ is called *ωSPDA-recognizable* if there exists an ωSPDA M with $\mathcal{L}(M) = L$.

Example 2.2 *We define an example automaton* $\mathcal{A} = (Q, \Sigma, \{S, B\}, T, S, \{S\})$ *with* $\Sigma = \{a, b\}$, $Q = \{S, M, B\}$ *and the transitions* T *as depicted in Fig. 1. In state* M*, the automaton reads* a *and pushes* B *for every read* a*. For every* B *that is popped from the stack, the automaton reads* b*. When there are no more* B *on the stack, the remaining* S *brings the automaton to start from the beginning. As* S *is the only final state, we have* $\mathcal{L}(\mathcal{A}) = \{a^n b^n \mid n \geq 1\}^\omega$*.*

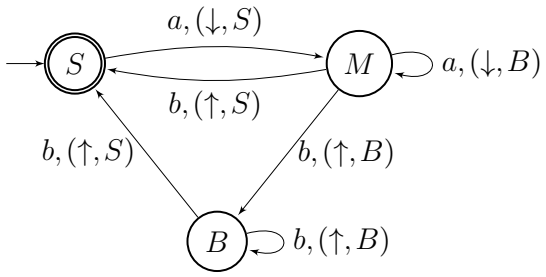


Figure 1: Example 2.2: Automaton

Theorem 2.3 *Every ω-context-free language is ωSPDA-recognizable.*

Proof. Let L be an ω-context-free language. Language L is recognized by some Büchi-accepting ω-context-free grammar $G = (N, \Sigma, P, S, F)$ in quadratic Greibach normal form. We construct an ωSPDA $M = (Q, \Sigma, \Gamma, T, q_0, F)$ with $Q = \Gamma = N$, $q_0 = S$, and

$$T = \{(A, a, B, (\downarrow, C)) \mid A \to aBC \in P\} \cup \tag{1}$$
$$\{(A, a, B, \#) \mid A \to aB \in P\} \cup \tag{2}$$
$$\{(A, a, B, (\uparrow, B)) \mid A \to a \in P, B \in N\} \tag{3}$$

for $a \in \Sigma$ and $A, B, C \in N$.

Intuitively, the variables in the grammar are simulated by states in the automaton. The second variable on the right side of the productions is pushed to the stack to store it for later. Whenever a final production is processed (Eq. (3)), it is checked which non-terminal is waiting on the stack to be processed. □

Example 2.4 *Let $G = (N, \Sigma, P, S, F)$ be a Büchi-accepting ω-context-free grammar with $N = \{S, M, B\}$, $\Sigma = \{a, b\}$, $F = \{S\}$ and P contains the following rules:*

$$S \to aMS$$
$$M \to b \mid aMB$$
$$B \to b$$

Then G is in quadratic Greibach normal form. Note that every non-terminal M derives a string $a^n b^{n+1}$ for $n \in \mathbb{N}$ and thus, $\mathcal{L}(G) = \{a^n b^n \mid n \geq 1\}^\omega$. By the construction from Theorem 2.3, G can be transformed into the ωSPDA in Fig. 1 from Example 2.2. Note that Eq. 3 generates a rule for every non-terminal in the grammar. As there are no transitions that push M onto the stack, we omit its pop-rule here.

3. Logic for ω-Context-Free Languages

The goal of this section is to find a Büchi-type logical formalism that is expressively equivalent to ωSPDA. This extends the work of Lautemann, Schwentick, Thérien [5] who defined a logic for context-free languages over finite words. Their proof uses context-free grammars in a symmetric version of the Greibach normal form. Here we use a direct translation from automata. The logic is composed of a monadic second-order logic together with one dyadic second-order variable that has to define a matching.

Let $w \in \Sigma^\omega$ be an ω-word. The set of all positions of w is $\mathbb{N}$. A binary relation $M \subseteq \mathbb{N} \times \mathbb{N}$ is *matching* (cf. [5]) if

- M is compatible with $<$, i.e., $(i, j) \in M$ implies $i < j$,
- each element i belongs to at most one pair in M,
- M is non-crossing, i.e., $(i, j) \in M$ and $(k, l) \in M$ with $i < k < j$ imply $i < l < j$.

Let Match$(\mathbb{N})$ denote the set of all matchings in $\mathbb{N} \times \mathbb{N}$. The definition of matching corresponds to the idea of Dyck languages and matchings are also important for nested words.

Let V_1, V_2 denote countable and pairwise disjoint sets of first-order and second-order variables. We fix a *matching variable* $\mu \notin V_1 \cup V_2$. Let $\mathcal{V} = V_1 \cup V_2 \cup \{\mu\}$. First-order variables $x \in V_1$ will be interpreted as positions in the ω-words and set variables $X \in V_2$ will be sets of positions.

Definition 3.1 *Let Σ be an alphabet. The set ωMSO(Σ) of matching ω-MSO formulas is defined by the extended Backus-Naur form (EBNF)*

$$\varphi ::= P_a(x) \mid x \leq y \mid x \in X \mid \mu(x,y) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\, \varphi \mid \exists X.\, \varphi$$

where $a \in \Sigma$, $x, y \in V_1$ and $X \in V_2$.

Here, $P_a(x)$ is a unary predicate indicating that the x-th letter of the word is a. Furthermore, $\mu(x,y)$ says that x and y will be matched. The other formulas behave in the expected way.

Definition 3.2 *We let ωML(Σ), the set of formulas of* matching ω-logic *over Σ, be the set of all formulas ψ of the form*

$$\psi = \exists\mu.\, (\varphi \wedge \mathit{MATCHING}(\mu)),$$

where $\varphi \in \omega\mathit{MSO}(\Sigma)$. The predicate $\mathit{MATCHING}(\mu) \in \omega\mathit{MSO}(\Sigma)$ ensures that μ is matching.

A language $L \subseteq \Sigma^\omega$ is *ωML-definable* if there exists a sentence $\psi \in \omega\mathrm{ML}(\Sigma)$ such that $\mathcal{L}(\psi) = L$.

Theorem 3.3 *Let Σ be an alphabet and $L \subseteq \Sigma^\omega$ an ω-language. Then L is ωML-definable if and only if L is ωSPDA-recognizable.*

Out of space restrictions, the proof is omitted here. There are two directions to prove: It is possible to describe the behavior of any ωSPDA as a logical formula. The other direction is done by structural induction. In formulas $\varphi \in \omega\mathrm{MSO}(\Sigma)$, the matching variable μ occurs freely. In the induction, it can therefore be encoded in the word; this encoding corresponds exactly to nested ω-words:

A *nested ω-word* nw over Σ is a pair (w, ν) where $w \in \Sigma^\omega$ is an ω-word and $\nu \in \mathrm{Match}(\mathbb{N})$ is a matching relation over $\mathbb{N}$ (cf. [1]).

Corollary 3.4 *Formulas $\varphi \in \omega\mathit{MSO}(\Sigma)$ are expressively equivalent to regular languages of infinite nested words.*

References

[1] R. ALUR, P. MADHUSUDAN, Visibly pushdown languages. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*. 2004, 202–211.

[2] J. R. BÜCHI, Weak Second-Order Arithmetic and Finite Automata. *Mathematical Logic Quarterly* **6** (1960), 66–92.

[3] R. S. COHEN, A. Y. GOLD, Theory of ω-Languages I: Characterizations of ω-Context-Free Languages. *Journal of Computer and System Sciences* **15** (1977) 2, 169–184.

[4] C. C. ELGOT, Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society* **98** (1961), 21–51.

[5] C. LAUTEMANN, T. SCHWENTICK, D. THÉRIEN, Logics for Context-Free Languages. In: *International Workshop on Computer Science Logic (CSL 1994)*. LNCS 933, Spinger, 1994, 205–216.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 66–69.

Properties and Decidability of Right One-Way Jumping Finite Automata

Simon Beier Markus Holzer

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{simon.beier,holzer}@informatik.uni-giessen.de

Abstract

Right one-way jumping finite automata (ROWJFAs) are jumping automata that process the input in a discontinuous way with the restriction that the input head reads deterministically from left-to-right starting from the leftmost letter in the input and when it reaches the end of the input word, it returns to the beginning and continues the computation. We characterize the family of permutation closed languages accepted by ROWJFAs in terms of Myhill-Nerode equivalence classes. Using this, we investigate closure and non-closure properties. Some interesting decidability questions concerning these automata are considered, too.

1. Introduction

One-way jumping finite automata were recently introduced in [2] as a variant of jumping finite automata [5], which is a machine model for discontinuous information processing. While a jumping automaton is allowed to nondeterministically read letters from anywhere in the input string, not necessarily only from the left of the remaining input, a (right) one-way jumping finite automaton (ROWJFA) moves its head from left-to-right starting from the leftmost letter in the input, reads some symbols, while it jumps over others, and when it reaches the end of the input word, it returns to the beginning and continues the computation, which is executed deterministically. For both automata models a letter is erased when it is read. In a series of papers [1, 3, 4, 7] different aspects mostly of jumping finite automata were investigated, such as, e.g., inclusion relations, closure and non-closure results, decision problems, computational complexity of jumping finite automata problems, etc. For ROWJFAs inclusion relations to well-known formal language families, closure and non-closure results under standard formal language operations were investigated. Nevertheless, a series of problems remained open in [2]. This is the starting point of our investigation.

First, we develop a characterization of (permutation closed) languages that are accepted by ROWJFAs in terms of the Myhill-Nerode relation. It is shown that the permutation closed language L belongs to **ROWJ**, the family of all languages accepted by ROWJFAs, if and only if L can be written as the *finite union* of Myhill-Nerode equivalence classes. The characterization allows us to identify languages that are *not* accepted by ROWJFAs, which are useful to prove non-closure results on standard formal language operations. In this way we solve all of the open problems from [2] on the inclusion relations of ROWJFAs languages to other language families

and on their closure properties. It is shown that the family **ROWJ** is an anti-abstract family of languages.

Then, decision problems for ROWJFAs are considered. It turns out that most problems such as, e.g., emptiness, finiteness, universality, the word problem and variants thereof, closure under permutation, etc., are decidable. Moreover, we show that the containment of a language within the strict hierarchy of ROWJFA permutation closed languages induced by the number of accepting states as well as whether jumping finite automata languages can be accepted by ROWJFAs is decidable, too. On the other hand, we prove that for (linear) context-free languages the corresponding ROWJFA acceptance problem becomes undecidable.

2. Preliminaries

For an alphabet Σ and a language $L \subseteq \Sigma^*$, let $\sim_L$ be the *Myhill-Nerode equivalence relation* on Σ^*. So, for $v, w \in \Sigma^*$, we have $v \sim_L w$ if and only if, for all $u \in \Sigma^*$, the equivalence $vu \in L \Leftrightarrow wu \in L$ holds. For $w \in \Sigma^*$, we call the equivalence class $[w]_{\sim_L}$ *positive* if and only if $w \in L$.

A *right one-way jumping finite automaton*, a ROWJFA for short, is a tuple $A = (Q, \Sigma, R, s, F)$, where Q is the finite *set of states*, Σ is the finite *input alphabet*, $\Sigma \cap Q = \emptyset$, R is a partial function from $Q \times \Sigma$ to Q, $s \in Q$ is the *start state*, and $F \subseteq Q$ is the set of *final states*. The elements of R are referred to as *rules* of A and we write $py \to q \in R$ instead of $R(p, y) = q$. A *configuration* of A is a string in $Q\Sigma^*$. The *right one-way jumping relation*, symbolically denoted by $\circlearrowright$, over $Q\Sigma^*$ is defined as follows. For $p \in Q$ we set

$$\Sigma_{R,p} = \{\, b \in \Sigma \mid pb \to q \in R \text{ for some } q \in Q \,\}.$$

Now, let $pa \to q \in R$, $x \in (\Sigma \setminus \Sigma_{R,p})^*$, and $y \in \Sigma^*$. Then, the ROWJFA A makes a *jump* from the configuration $pxay$ to the configuration qyx, symbolically written as $pxay \circlearrowright qyx$. Let $\circlearrowright^*$ denote the transitive-reflexive closure of $\circlearrowright$. The *language accepted by* A is $L(A) = \{\, w \in \Sigma^* \mid \exists f \in F : sw \circlearrowright^* f \,\}$. Let **ROWJ** be the family of all languages that are accepted by ROWJFAs. Furthermore, for $n \geq 0$, let $\mathbf{ROWJ_n}$ be the class of all languages accepted by ROWJFAs with at most n accepting states.

Let **JFA** be the family of all languages accepted by ordinary jumping finite automata, see [5]. Besides the above mentioned language families let **REG**, **DCF**, **CF**, and **CS** be the families of regular, deterministic context-free, context-free, and context-sensitive languages. Moreover, we are interested in permutation closed language families. These language families are referred to by a prefix **p**. E.,g., **pROWJ** denotes the language family of all permutation closed **ROWJ** languages.

3. Properties of Right One-Way Jumping Finite Automata

We give a characterization for permutation closed languages that are accepted by an ROWJFA in terms of the Myhill-Nerode relation:

Theorem 3.1 *Let L be a permutation closed language and $n \geq 0$. Then, the language L is in $\mathbf{ROWJ_n}$ if and only if the Myhill-Nerode relation $\sim_L$ has at most n positive equivalence classes.*

The following relations were given in [2]: (1) **REG** $\subset$ **ROWJ**, (2) **ROWJ** and **CF** are incomparable, and (3) **ROWJ** $\not\subseteq$ **JFA**. It was stated as an open problem if **JFA** $\subset$ **ROWJ**. The answer is part of the next theorem.

Theorem 3.2 *We have (1)* **ROWJ** $\subset$ **CS**, *(2)* **ROWJ** *and* **DCF** *are incomparable, (3)* **ROWJ** *and* **JFA** *are incomparable, and (4) every language in* **ROWJ** *is semilinear.*

The closure properties of the language families **ROWJ** and **pROWJ** are summarized in Table 1.

Closed under	Language family			
	REG	**pROWJ**	**ROWJ**	**JFA**
Union	yes	no	no	yes
Union with reg. lang.	yes	no	no	no
Intersection	yes	yes	no	yes
Intersection with reg. lang.	yes	no	no	no
Complementation	yes	no	no	yes
Reversal	yes	yes	no	yes
Concatenation	yes	no	no	no
Right conc. with reg. lang.	yes	no	no	no
Left conc. with reg. lang.	yes	no	no	no
Left conc. with prefix-free reg. lang.	yes	no	yes	no
Kleene star or plus	yes	no	no	no
Homomorphism	yes	no	no	no
Inv. homomorphism	yes	yes	no	yes
Substitution	yes	no	no	no
Permutation	no	yes	no	yes

Table 1: Closure properties of **ROWJ** and **pROWJ**. The gray shaded results were proven by us. The non-shaded closure properties for **REG** are folklore. For **ROWJ** the closure/non-closure results can be found in [2] and that for the language family **JFA** in [1, 3, 4, 6].

4. Decidability of Right One-Way Jumping Finite Automata

The word problem, emptiness, finiteness, and universality are decidable for ROWJFAs:

Theorem 4.1 *Let A be an ROWJFA with input alphabet Σ and $w \in \Sigma^*$. Then, it is decidable (i) whether $w \in L(A)$, (ii) whether $L(A)$ is empty, (iii) whether $L(A)$ is finite, and (iv) whether $L(A)$ is universal, that is, $L(A) = \Sigma^*$.*

We can also decide if a word is extendable such that it becomes acceptable by an ROWJFA:

Theorem 4.2 *Let A be an ROWJFA with input alphabet Σ and $w \in \Sigma^*$. Then, it is decidable whether there is a $v \in L(A)$ such that (i) the word w is a prefix of v, (ii) the word w is a suffix of v, (iii) the word w is a factor of v, and (iv) the word w is a sub-word of v.*

It is decidable if an ROWJFA accepts a permutation closed language:

Theorem 4.3 *Let A be an ROWJFA. Then, it is decidable whether $L(A)$ is closed under permutation.*

It was shown in [2] that **JFA** is the family of all permutation closed, semilinear languages. So a language L from **ROWJ** is in **JFA** if and only if L is closed under permutation, which is decidable by the last theorem. Containment in **pROWJ**, or even in the infinite hierarchy induced by the number of accepting states, can be decided for DFAs, JFAs, and ROWJFAs:

Theorem 4.4 *Let A be a DFA, a JFA, or an ROWJFA and $n \geq 0$. Then, it is decidable (i) whether $L(A)$ is in the family* **pROWJ** *and (ii) whether $L(A)$ is in* $\mathbf{pROWJ_n}$.

For one-turn pushdown automata the considered problems become undecidable. Observe, that one-turn pushdown automata accept exactly those languages generated by linear context-free grammars.

Theorem 4.5 *Let A be a one-turn PDA. Then, it* undecidable *(i) whether $L(A)$ is in the family* **ROWJ** *and (ii) whether $L(A)$ is in* **pROWJ**. *For each $n > 0$, it is* undecidable *(iii) whether $L(A)$ is in* $\mathbf{ROWJ_n}$ *and (iv) whether $L(A)$ is in* $\mathbf{pROWJ_n}$.

It is still open if the problems of regularity, disjointness, inclusion, or equivalence are decidable for languages accepted by ROWJFAs.

References

[1] S. BEIER, M. HOLZER, M. KUTRIB, Operational State Complexity and Decidability of Jumping Finite Automata. In: É. CHARLIER, J. LEROY, M. RIGO (eds.), *Proceedings of the* 21*st International Conference on Developments in Language Theory*. Number 10396 in LNCS, Springer, Liège, Belgium, 2017, 96–108.

[2] H. CHIGAHARA, S. FAZEKAS, A. YAMAMURA, One-Way Jumping Finite Automata. *Internat. J. Found. Comput. Sci.* **27** (2016) 3, 391–405.

[3] H. FERNAU, M. PARAMASIVAN, M. L. SCHMID, Jumping Finite Automata: Characterizations and Complexity. In: F. DREWES (ed.), *Proceedings of the* 20*th Conference on Implementation and Application of Automata*. Number 9223 in LNCS, Springer, Umeå, Sweden, 2015, 89–101.

[4] H. FERNAU, M. PARAMASIVAN, M. L. SCHMID, V. VOREL, *Characterization and Complexity Results on Jumping Finite Automata*. http://arxiv.org/abs/1512.00482, 2015.

[5] A. MEDUNA, P. ZEMEK, Jumping Finite Automata. *Internat. J. Found. Comput. Sci.* **23** (2012) 7, 1555–1578.

[6] A. MEDUNA, P. ZEMEK, *Regulated Grammars and Automata*, chapter 17: Jumping Finite Automata. Springer, 2014, 567–585.

[7] V. VOREL, *Basic Properties of Jumping Finite Automata*. http://arxiv.org/abs/1511.08396v2, 2015.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 70–70.

Überlegungen zur Černý-Vermutung

Stefan Hoffmann

Ein endlicher Automat heisst synchronisierend, sofern es ein Wort gibt welches den Automaten von jedem beliebigen Zustand in einen endeutig bestimmten Zustand überführt. Für einen synchronisierenden Automaten stellt sich die Frage nach dem kürzesten derartigen Wort, Cerny[1] vermutete dass dies höchstens quadratisch von der Zustandszahl abhängt. Dies ist aber nach wie vor offen, und die besten Schranken für allgemeine Automaten sind kubisch. Die Beweise basieren meist auf (extremal-)kombinatorischen Überlegungen. Neuere Arbeiten, z.B. von Szykula[2], legen einen mehr (linear-) algebraisch orientierten Ansatz nahe. Um Ideen letzteren Ansatz weiterzuführen und zu verallgemeinern soll es in diesem Vortrag gehen.

References

[1] Ján Černý. Poznámka k. homogénnym experimentom s konecnými automatmi. *Mat. fyz. čas SAV*, 14:208–215, 1964.

[2] Marek Szykula. Improving the upper bound on the length of the shortest reset word. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 71–73.

Reachability Questions on Partially Lossy Queue Automata

Chris Köcher

Technische Universität Ilmenau, Fachgebiet Automaten und Logik
chris.koecher@tu-ilmenau.de

Data structures are possibly the most important concept in computer science. Famous data structures are, e.g., finite memories, counters, and Turing-tapes. But the most fundamental ones are stacks and queues. Although both data structures have the same set of operations (reading and writing of a letter a), there is a big difference from computability's point of view: if we equip finite automata with a stack (these are the well-known pushdown automata) then these models compute exactly the context-free languages. But if we equip a finite automaton with a queue (these are so-called queue automata) we obtain a Turing-complete computation model [4]. This very strong model can be weakened by allowing the queue to forget any part of its content at any time. We call these queues *(fully) lossy queues*.

When studying the similarities and differences between these two types of queues we found it convenient to join them into one common model. These are the so-called *partially lossy queues* (*plq*s for short). This type of queues allows to specify which letters in the queue can be forgotten at any time and which ones are unforgettable. Some algebraic results on partially lossy queues can be found in [10, 9]. In both papers we studied the behavior of partially lossy queues by considering their transformation monoid.

Here, we discuss some reachability questions concerning automata equipped with partially lossy queues, which we call *partially lossy queue automata*. To this end, for a set of transformations T and a language L of queue contents we define the set $\mathsf{post}_T(L)$ of all queue contents after execution of a transformation in T on a partially lossy queue with content in L. In other words, L represents a sets of inputs into the partially lossy queue automaton, T the set of all paths in its control component (this is an NFA labeled with atomic operations), and $\mathsf{post}_T(L)$ the corresponding set of outputs. Then the considered computational problem is defined as follows:

Problem 1 *Given a regular language T of transformations and a regular language L of queue contents, compute* $\mathsf{post}_T(L)$.

Partially lossy queue automata with at least one non-forgettable letter are Turing-complete. Hence, $\mathsf{post}_T(L)$ for these partially lossy queues can be any recursively enumerable language. In this case the language $\mathsf{post}_T(L)$ can be undecidable even if the language of transformations T equals $\{w_1, \ldots, w_n\}^*$ for some transformations $w_1, \ldots, w_n$.

For fully lossy queues, $\mathsf{post}_T(L)$ is a regular language since it is downwards closed under the subword ordering [7]. Though, it is not possible to compute a finite automaton accepting $\mathsf{post}_T(L)$ [11]. But we can construct a Turing machine accepting this language [2, 5]. This is

in some sense optimal since Schnoebelen and Chambart proved in [12, 6] that deciding membership of $\mathsf{post}_T(L)$ is not primitive recursive.

In this paper we consider some restrictions on the language of transformations T. On the one hand, we regard regular languages that are closed under certain commutations of the atomic operations which guarantee the same behavior of the queue. In this case for arbitrary partially lossy queues $\mathsf{post}_T(L)$ is effectively regular. Thereby, an NFA accepting this language can be computed in polynomial time. If we use an on-the-fly construction of this automaton we can decide its membership problem using non-deterministic logarithmic space, only. Additionally, if the language L of queue contents is context-free then $\mathsf{post}_T(L)$ is context-free as well.

On the other hand, we consider transformation languages of the form T^* where T is a finite, so-called *read-write independent* set of transformations. These are sets such that for each two words $s, t \in T$ there is a word $u \in T$ where u and s have the same subsequence of write actions and u and t have the same subsequence of read actions. In this case, we can compute an NFA accepting $\mathsf{post}_{T^*}(L)$ using polynomial space, only. This result also covers the special case w^* for some transformation w which was first studied in [3, 1] for fully reliable and fully lossy queues, respectively. An NFA accepting $\mathsf{post}_{w^*}(L)$ can be computed in polynomial time and, again, its membership problem is in NL.

We may also consider backwards reachability in partially lossy queue automata: for a given language T of transformations and a language L of queue contents, the set $\mathsf{pre}_T(L)$ is the set of all queue contents which can reach contents in L after execution of a transformation in T on a partially lossy queue. Then we regard the following computational problem:

Problem 2 *Given a regular language T of transformations and a regular language L of queue contents, compute* $\mathsf{pre}_T(L)$.

While this problem is the same as the one above for fully reliable queues due to self-duality (cf. [8]), there are some differences for arbitrary partially lossy queues. In this general case for fully lossy queues it is possible to compute an NFA accepting $\mathsf{pre}_T(L)$. Though, the complexity for computing this NFA is, again, not primitive recursive.

For transformation languages T that are closed under the special commutations we mentioned above, we can compute an NFA accepting $\mathsf{pre}_T(L)$ using polynomial space. Similarly, for T^* where T is finite, read-write independent we can also compute an NFA accepting $\mathsf{pre}_{T^*}(L)$ using polynomial space. In particular, for w^* with a transformation w this is possible in polynomial time, again.

References

[1] P. A. ABDULLA, A. COLLOMB-ANNICHINI, A. BOUAJJANI, B. JONSSON, Using Forward Reachability Analysis for Verification of Lossy Channel Systems. *Formal Methods in System Design* **25** (2004) 1, 39–65.

[2] P. A. ABDULLA, B. JONSSON, Verifying Programs with Unreliable Channels. *Information and Computation* **127** (1996) 2, 91–101.

[3] B. BOIGELOT, P. GODEFROID, B. WILLEMS, P. WOLPER, The Power of QDDs. In: *Static Analysis*. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1997, 172–186.

[4] D. BRAND, P. ZAFIROPULO, On Communicating Finite-State Machines. *Journal of the ACM* **30** (1983) 2.

[5] G. CÉCÉ, A. FINKEL, S. PURUSHOTAMAN IYER, Unreliable Channels Are Easier to Verify than Perfect Channels. *Information and Computation* **124** (1996) 1, 20–31.

[6] P. CHAMBART, P. SCHNOEBELEN, The Ordinal Recursive Complexity of Lossy Channel Systems. In: *LICS'08*. IEEE Computer Society Press, 2008, 205–216.

[7] L. H. HAINES, On Free Monoids Partially Ordered by Embedding. *Journal of Combinatorial Theory* **6** (1969) 1, 94–98.

[8] M. HUSCHENBETT, D. KUSKE, G. ZETZSCHE, The Monoid of Queue Actions. *Semigroup forum* **95** (2017) 3, 475–508.

[9] C. KÖCHER, Rational, Recognizable, and Aperiodic Sets in the Partially Lossy Queue Monoid. In: *STACS'18*. LIPIcs 96, Dagstuhl Publishing, 2018, 45:1–45:14.

[10] C. KÖCHER, D. KUSKE, O. PRIANYCHNYKOVA, The Inclusion Structure of Partially Lossy Queue Monoids and Their Trace Submonoids. *RAIRO - Theoretical Informatics and Applications* **52** (2018) 1, 55–86.

[11] R. MAYR, Undecidable Problems in Unreliable Computations. *Theoretical Computer Science* **297** (2003) 1, 337–354.

[12] P. SCHNOEBELEN, Verifying Lossy Channel Systems Has Nonprimitive Recursive Complexity. *Information Processing Letters* **83** (2002) 5, 251–261.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 74–77.

Kuratowski's Complement-Closure Theorem and the Orbit of Closure-Involution Operations

Jürgen Dassow[(A)]

[(A)]Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik,
Postfach 4120, 39016 Magdeburg, Germany
dassow@iws.cs.uni-magdeburg.de

Abstract

For a closure operation c and an involution i defined on a language family $\mathcal{L}$, we define $\mathcal{N}_{c,i}^{\mathcal{L}}(L)$ as the number of languages which can be obtained from L by repeated applications of c and i. The orbit $\mathcal{O}^{\mathcal{L}}(c,i)$ of c and i is defined as the set of all these numbers.

We determine the orbit for some classical language theoretic closure operators and the involution complement. We show that complement and reversal behave very different with respect to some special closure operators. Moreover, we prove that, for an arbitrary closure operator and reversal, some "small" numbers cannot be avoided in the orbit.

1. Introduction

In 1922, Kuratowski proved the following closure-complement theorem (see [3]): *If $(X, \mathcal{T})$ is a topological space and $A \subseteq X$, then at most 14 sets can be obtained from A by repeated applications of the operations topological closure and complement. Furthermore, there is a topological space and a set for which the bound 14 is achieved.*

In 1960, Hammer noticed that such a statement holds in a more general setting; the theorem also holds if - instead of the topological closure - a closure operator on a set X is used (see [2]). In 1984, because the (positive) Kleene-closure is a closure operator, Peleg noticed that the repeated application of (positive) Kleene-closure and complement to a language yields at most 14 (and at most 10, respectively) different languages and there is a language L such that the repeated application of (positive) Kleene-closure and complement gives axactly 14 (10, respectively) different languages (see [4]). In 2009, this statement was refined by Brozoswki, Grant, and Shallit, who proved that, for any numbers $n \in \{4,6,8,10,12,14\}$ ($m \in \{2,4,6,8,10\}$), there are languages L_n (L_m) such that the repeated application of (positive) Kleene-closure and complement to L_n (L_m) yields exactly n (m, respectively) different sets. Moreover, for any number $n \in \{4,6,8,10,12,14\}$ ($m \in \{2,4,6,8,10\}$), the authors gave necessary and sufficient conditions such that the repeated application of (positive) Kleene-closure and complement to a language L yields exactly n (m,respectively) different sets (see [1]).

Thus, in contrast to Kuratowski (and followers), Brozoswki, Grant, and Shallit were not only interested the maximal number of sets which can be obtained by repeated applications of complement and closure; they were interested in the orbit of complement and closure, which consists of all numbers which can be obtained by such applications. In this paper we continue

the determination of the orbit for some closure operators (not necessarily, Kleene-closure) and complement or reversal.

2. Definitions

In the sequel, X is an alphabet and $\mathcal{L}_X$ is the family of all languages over X.

An operator c is called a closure operator on $\mathcal{L}_X$, if the following four conditions are satisfied:

- For all sets $L \in \mathcal{L}_X$, $c(L) \in \mathcal{L}_X$.
- For all sets $L \in \mathcal{L}_X$, $L \subseteq c(L)$.
- For all sets $L \in \mathcal{L}_X$ and $K \in \mathcal{L}_X$, the inclusion $L \subseteq K$ implies $c(L) \subseteq c(K)$.
- For all sets $L \in \mathcal{L}_X$, $c(c(L) = c(L)$, i. e., c is idempotent.

We say that i is an involution on $\mathcal{L}_X$, if $i(L) \in \mathcal{L}_X$ and $i(i(L)) = L$ hold for every set $L \in \mathcal{L}_X$.

Let c be a closure operator on $\mathcal{L}_X$ and i an involution on $\mathcal{L}_X$. Then, for $L \in \mathcal{L}_X$, we define the orbit $\mathcal{O}_{c,i}^{\mathcal{L}_X}(L)$ as the set of all sets which can be obtained from L by repeated applications of c and i and set

$$\mathcal{N}_{c,i}^{\mathcal{L}_X}(L) = \text{card}(\mathcal{O}_{c,i}^{\mathcal{L}_X}(L)).$$

We mention that, by the fourth condition for a closure operator and the second condition for an involution it is sufficient to consider the sequences

$$L, c(L), i(c(L)), c(i(c(L))), i(c(i(c(L)))), \ldots \text{ and } L_i(L), c(i(L)), i(c(i(L))), c(i(c(i(L)))), \ldots,$$

in order to determine $\mathcal{O}_{c,i}^{X}(L)$, i. e., it is sufficient to consider alternating applications of the two operators.

Moreover, we define the orbit of the pair (c, i) as

$$\mathcal{O}^{\mathcal{L}_X}(c, i) = \{n \mid \mathcal{N}_{c,i}^{\mathcal{L}_X}(L) = n \text{ for some } L \in \mathcal{L}_X\}$$

and Kuratowski's number of (c, i) as

$$\mathcal{K}^{\mathcal{L}_X}(c, i) = \max\{n \mid n \in \mathcal{O}^{\mathcal{L}_X}(c, i)\}.$$

We mention some further closure operators on languages which are based on (special) subwords and superwords.

$$\begin{aligned}
p(L) &= \{y \mid yx \in L \text{ for some } x \in X^*\},\\
s(L) &= \{y \mid xy \in L \text{ for some } x \in X^*\},\\
f(L) &= \{y \mid x_1 y x_2 \in L \text{ for some } x_1, x_2 \in X^*\},\\
u(L) &= \{uwv \mid w \in L, u, v \in X^*\} = X^* L X^*,\\
ri(L) &= \{wv \mid w \in L, v \in X^*\} = L X^*,\\
li(L) &= \{uw \mid w \in L, u \in X^*\} = X^* L.
\end{aligned}$$

Furthermore, instead of the involution complement, one can consider the involution reversal inductively defined (over an alphabet X) by

$$\lambda^R = \lambda,\ x^R = x \text{ for } x \in X,\ (wx)^R = xw^R \text{ for } w \in X^+, x \in X,\ L^R = \{w^R \mid w \in L\}.$$

3. Results for some "Classical" Closure Operators

We start with the determination the orbit using factors or prefixes or suffixes as closure operator and complement as involution.

Theorem 3.1 *For any alphabet X,*

$$\begin{aligned}\{2,4,6\} &= \mathcal{O}^{\mathcal{L}_X}(f,^-) = \mathcal{O}^{\mathcal{L}_X}(p,^-) = \mathcal{O}^{\mathcal{L}_X}(s,^-)\\ &= \mathcal{O}^{\mathcal{L}_X}(u,^-) = \mathcal{O}^{\mathcal{L}_X}(ri,^-) = \mathcal{O}^{\mathcal{L}_X}(li,^-).\end{aligned}$$

We mention that, for subwords as closure, we have characterizations of the languages with a given cardinality of the orbit. More precisely,

- $\mathcal{N}_{f,^-}^{\mathcal{L}_X}(L) = 2$ if and only if $L = X^*$ or $L = \emptyset$.
- $\mathcal{N}_{f,^-}^{\mathcal{L}_X}(L) = 4$ if and only if one of the following three conditions hold:
  - $f(L) = L$ and $f(\overline{L}) \neq \overline{L}$.
  - $f(L) \neq L$ and $f(\overline{L}) = \overline{L}$.
  - $L \neq f(L)$ and $\overline{L} \neq f(\overline{L})$ and $f(L) = f(\overline{L})$.

Now we give some results concerning the orbit of some known closure operators and reversal. We start with a simple observation on unary alphabets.

Lemma 3.2 *For any unary alphabet X and any closure operator $c \neq$ id on $\mathcal{L}_X$, we have $\mathcal{O}^{\mathcal{L}_X}(c,R) = \{1,2\}$.*

Theorem 3.3 *For any alphabet X with at least two letters, we have*

$$\begin{aligned}&i)\ \ \mathcal{O}^{\mathcal{L}_X}(*,R) = \mathcal{O}^{\mathcal{L}_X}(+,R) = \mathcal{O}^{\mathcal{L}_X}(f,R) = \{1,2,3,4\},\\ &ii)\ \ \mathcal{O}^{\mathcal{L}_X}(p,R) = \mathcal{O}^{\mathcal{L}_X}(s,R) = \{1,2,3,4,5,6,7,8\},\\ &iii)\ \ \mathcal{O}^{\mathcal{L}_X}(u,R) = \mathcal{O}^{\mathcal{L}_X}(ri,R) = \mathcal{O}^{\mathcal{L}_X}(li,R) = \{1,2,3,4\}.\end{aligned}$$

4. Results on Special Closure Operators and Reversal

By Kuratowski's Theorem, we have $\mathcal{K}^X(c,^-) \leq 14$ for all closure operators c, and by the properties of the complement, $\mathcal{O}^X(c,^-)$ contains only even numbers. We now show that the use of reversal and certain closure operators lead to cases where no Kuratowski's number exists (i. e., there are languages from which infinitely many languages can be obtained using repeated applications of the closure operator and reversal), to arbitrary large orbits and to orbits containing even as well as odd numbers.

Theorem 4.1 *Let X be an alphabet with at least two letters. Then there is a closure operation c_1 such that $\mathcal{O}^{\mathcal{L}_X}(c_1,R) = \{1,2,3,\infty\}$.*

Theorem 4.2 *Let X be an alphabet with at least two letters and n a positive integer. Then there is a closure operation c_2 such that*

$$\mathcal{O}^{\mathcal{L}_X}(c_2,R) = \{1,3\} \cup \{2,4,\ldots,2n+2\}.$$

Theorem 4.3 *Let X be an alphabet with at least two letters and n a positive integer. Then there is a closure operation c_3 such that $\mathcal{O}^{\mathcal{L}_X}(c_2, R) = \{1,2,3\} \cup \{5,7,\ldots,2n+3\}$.*

Looking on the previous results one can feel that any possible set of numbers can be an orbit of some closure operator and reversal. However, this is false, because certain small numbers cannot be avoided.

Theorem 4.4 *Let c a closure operator on L_X. If $\mathcal{K}^{\mathcal{L}_X}(c, R) \geq 6$, then we have*

$$\{2k+1 \mid 0 \leq k \leq \frac{\mathcal{K}^{\mathcal{L}_X}(c,R)}{4} - 1\} \subset \mathcal{O}^{\mathcal{L}_X}(c,R)$$

or

$$\{2k \mid 1 \leq k \leq \frac{\mathcal{K}^{\mathcal{L}_X}(c,R)}{4} - 1\} \subset \mathcal{O}^{\mathcal{L}_X}(c,R).$$

References

[1] J. BRZOZOWSKI, E. GRANT, J. SHALLIT, Closures in formal languages and Kuratowski' theorem. In: V. DIEKERT, D. NOWOTKA (eds.), *Development of Languages 2009*. LNCS 5583, Springer-Verlag, Berlin, 2009, 125–144.

[2] D. HAMMER, Kuratowski's closure theorem. *Nieuw Archief v. Wiskunde* **7** (1960), 74–80.

[3] C. KURATOWSKI, Sur l'opération $\overline{A}$ de l'analysis situs. *Fund. Math.* **3** (1922), 182–199.

[4] D. PELEG, A generalized closure and complement phenomenon. *Discrete Math.* **50** (1984), 285–293.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 78–81.

Networks of Evolutionary Processors with Resources Restricted Filters

Bianca Truthe

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
bianca.truthe@informatik.uni-giessen.de

Abstract

In this paper, we continue the research on networks of evolutionary processors where the filters belong to several special families of regular languages. These subregular families are defined by restricting the resources needed for generating or accepting them (the number of states of the minimal deterministic finite automaton accepting a language of the family as well as the number of non-terminal symbols or the number of production rules of a right-linear grammar generating such a language). We insert the newly defined language families into the hierachy of language families obtained by using as filters languages of other subregular families (such as ordered, non-counting, power-separating, circular, suffix-closed regular, union-free, definite, combinational, finite, monoidal, nilpotent, or commutative languages).

1. Introduction

Networks of language processors have been introduced in [3] by E. CSUHAJ-VARJÚ and A. SALOMAA. Such a network can be considered as a graph where the nodes represent processors which apply production rules to the words they contain. In a derivation step (an evolutionary step), any node derives from its language all possible words as its new language. In a communication step, any node sends those words to other nodes which satisfy an output condition given as a regular language (called the output filter) and any node adopts words sent by the other nodes if the words satisfy an input condition also given by a regular language (called the input filter). The language generated by a network of language processors consists of all (terminal) words which occur in the languages associated with a given node.

Inspired by biological processes, in [1] a special type of networks of language processors was introduced. The nodes of such networks are called evolutionary processors because the allowed productions model the point mutation known from biology. The productions of a node allow that one letter is substituted by another letter, letters are inserted, or letters are deleted; the nodes are then called substitution nodes, insertion nodes, or deletion nodes, respectively. Results on networks of evolutionary processors can be found, e. g., in [1], [2], [6]. For instance, in [2], it was shown that networks of evolutionary processors are complete in that sense that they can generate any recursively enumerable language.

This is a short version of a paper presented at NCMA 2018 in Košice (Slovakia) [7].

In [4] and [5], the generative capacity of networks of evolutionary processors was investigated for cases that all filters belong to a certain subfamily of the set of all regular languages.

In [7], the research on networks of evolutionary processors is continued where the filters are restricted by further bounded resources, namely the number of non-terminal symbols or the number of production rules which are necessary for generating the languages. Additionally, filters are considered which are accepted by deterministic finite automata over an arbitrary alphabet with a bounded number of states.

2. Preliminaries

Here, we explain networks of evolutionary processors (NEPs) and present the language families which are considered for the filters. Let V be an alphabet. By V^* we denote the set of all words over the alphabet V (including the empty word λ).

Intuitively, a network over an alphabet V with n evolutionary processors is a graph consisting of n nodes (also called processors) and a set of directed edges between nodes. Any processor N_i ($1 \leq i \leq n$) consists of a set M_i of evolutionary rules, a set A_i of words, an input filter I_i, and an output filter O_i. We say that N_i is

- a substitution node if $M_i \subseteq \{a \to b \mid a,b \in V\}$ (by any rule, a letter is substituted by another one),
- a deletion node if $M_i \subseteq \{a \to \lambda \mid a \in V\}$ (by any rule, a letter is deleted), or
- an insertion node if $M_i \subseteq \{\lambda \to b \mid b \in V\}$ (by any rule, a letter is inserted).

Every node has rules from one type only. The input filter I_i and the output filter O_i control the words which are allowed to enter and to leave the node, respectively. With any node N_i and any time moment $t \geq 0$, we associate a set $C_t(i)$ of words (the words contained in the node at time t). Initially, N_i contains the words of A_i. In an evolutionary step, we derive from $C_t(i)$ all words by applying rules from the set M_i. In a communication step, any processor N_i sends out all words from the set $C_t(i) \cap O_i$ (which pass the output filter) to all processors to which a directed edge exists (only the words from $C_t(i) \setminus O_i$ remain in the set associated with N_i) and, moreover, it receives from any processor N_k such that there is an edge from N_k to N_i all words sent by N_k and passing the input filter I_i of N_i, i. e., the processor N_i gets in addition all words of $C_t(k) \cap O_k \cap I_i$. We start with an evolutionary step and then communication steps and evolutionary steps are alternately performed. The language consists of all words which are in some node N_j (also called the output node, j is chosen in advance) at some moment t, $t \geq 0$.

For a family X, we denote the family of languages generated by networks of evolutionary processors where all filters are of type X by $\mathcal{E}(X)$.

In this paper, we consider filters which are defined by bounding the resources which are necessary for accepting or generating these languages. By RL_n^V, RL_n^P, and REG_n^Z, we denote the family of all languages which are generated by a right-linear grammar with at most n non-terminal symbols or production rules or accepted by a deterministic finite automaton with at most n states, respectively.

In [4], the following restrictions for regular languages are considered. In order to relate our results of this paper to the results there, we explain here those special regular languages. Let L be a language and V the minimal alphabet of L. We say that the language L, with respect to the

alphabet V, is

- *monoidal* if $L = V^*$,
- *combinational* if it has the form $L = V^*A$ for some subset $A \subseteq V$,
- *definite* if it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *nilpotent* if L is finite or $V^* \setminus L$ is finite,
- *commutative* if $L = \{\, a_{i_1} \dots a_{i_n} \mid a_1 \dots a_n \in L,\ n \geq 1,\ \{i_1, i_2, \dots, i_n\} = \{1, 2, \dots, n\} \,\}$,
- *circular* if $L = \{\, vu \mid uv \in L,\ u, v \in V^* \,\}$,
- *suffix-closed* if the relation $xy \in L$ for some words $x, y \in V^*$ implies that also the suffix y belongs to L or equivalently, $L = \{\, v \mid uv \in L,\ u, v \in V^* \,\}$,
- *non-counting* (or star-free) if there is an integer $k \geq 1$ such that, for any $x, y, z \in V^*$, the relation $xy^k z \in L$ holds if and only if also $xy^{k+1}z \in L$ holds,
- *power-separating* if for any word $x \in V^*$ there is a natural number $m \geq 1$ such that either the equality $J_x^m \cap L = \emptyset$ or the inclusion $J_x^m \subseteq L$ holds where $J_x^m = \{x^n \mid n \geq m\}$,
- *ordered* if L is accepted by some finite automaton $\mathcal{A} = (Z, V, \delta, z_0, F)$ where $(Z, \preceq)$ is a totally ordered set and, for any $a \in V$, $z \preceq z'$ implies $\delta(z, a) \preceq \delta(z', a)$,
- *union-free* if L can be described by a regular expression which is only built by product and star.

Among the commutative, circular, suffix-closed, non-counting, and power-separating languages, we consider only those which are also regular.

By *FIN*, *MON*, *COMB*, *DEF*, *NIL*, *COMM*, *CIRC*, *SUF*, *NC*, *PS*, *ORD*, and *UF*, we denote the families of all finite, monoidal, combinational, definite, nilpotent, regular commutative, regular circular, regular suffix-closed, regular non-counting, regular power-separating, ordered, and union-free languages, respectively. Furthermore, *REG*, *CF*, and *RE* denote the families of all regular, all context-free, and all recursively enumerable languages, respectively.

The following theorem is known (see, e. g., [2]).

Theorem 2.1 $\mathcal{E}(REG) = RE$.

3. Summary of the Results

Networks of evolutionary processors are investigated where the filters belong to subregular language families which are defined by restricting the resources needed for generating or accepting them (the number of states of the minimal deterministic finite automaton accepting a language of the family, the number of non-terminal symbols, or the number of production rules of a right-linear grammar generating such a language). These language families are inserted into the hierachy of language families obtained by using languages of other subregular families as filters (such as ordered, non-counting, power-separating, circular, suffix-closed regular, union-free, definite, combinational, finite, monoidal, nilpotent, or commutative languages) which was published in [4]. The hierarchy with the new results is shown in Figure 1.

Theorem 3.1 *The relations shown in Figure 1 hold.*

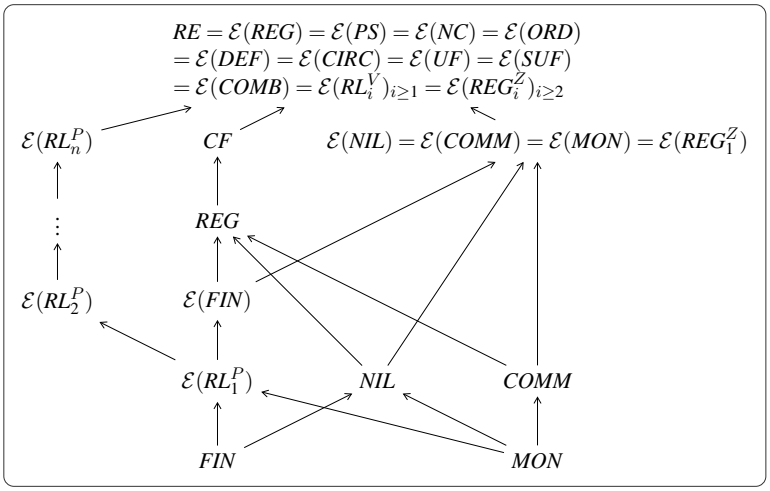


Figure 1: Hierarchy of language families by NEPs with filters from subregular families. An arrow from a language family X to a language family Y stands for the proper inclusion $X \subset Y$. If two families X and Y are not connected by a directed path, then the families are incomparable.

References

[1] J. CASTELLANOS, C. MARTÍN-VIDE, V. MITRANA, J. M. SEMPERE, Solving NP-Complete Problems with Networks of Evolutionary Processors. In: *IWANN '01: Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks*. LNCS 2084, Springer-Verlag Berlin, 2001, 621–628.

[2] J. CASTELLANOS, C. MARTÍN-VIDE, V. MITRANA, J. M. SEMPERE, Networks of Evolutionary Processors. *Acta Informatica* **39** (2003) 6–7, 517–529.

[3] E. CSUHAJ-VARJÚ, A. SALOMAA, Networks of Parallel Language Processors. In: *New Trends in Formal Languages – Control, Cooperation, and Combinatorics*. LNCS 1218, Springer-Verlag Berlin, 1997, 299–318.

[4] J. DASSOW, F. MANEA, B. TRUTHE, Networks of Evolutionary Processors: The Power of Subregular Filters. *Acta Informatica* **50** (2013) 1, 41–75.

[5] J. DASSOW, B. TRUTHE, On Networks of Evolutionary Processors with Filters Accepted by Two-State-Automata. *Fundamenta Informaticae* **112** (2011) 2–3, 157–170.

[6] C. MARTÍN-VIDE, V. MITRANA, Networks of Evolutionary Processors: Results and Perspectives. In: M. GHEORGHE (ed.), *Molecular Computational Models: Unconventional Approaches*. Idea Group Publishing, 2005, 78–114.

[7] B. TRUTHE, Networks of Evolutionary Processors with Resources Restricted Filters. In: R. FREUND, M. HOSPODÁR, G. JIRÁSKOVÁ, G. PIGHIZZINI (eds.), *10th Workshop on Non-Classical Models of Automata and Applications (NCMA), Košice, Slovakia, August 21–22, 2018, Proceedings*. books@ocg.at 332, Österreichische Computer Gesellschaft, Austria, 2018, 165–180.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 82–86.

One-Dimensional Tiling Systems and String Rewriting

Alfons Geser[(A)] Dieter Hofbauer[(B)] Johannes Waldmann[(A)]

[(A)]HTWK Leipzig, Germany

[(B)]ASW – Berufsakademie Saarland, Germany

Abstract

We use one-dimensional tiling systems (strictly locally testable languages) to over-approximate reachability sets in string rewriting, and apply this to prove termination automatically. This refines the root labeling method by restricting to right-hand sides of forward closures.

1. Motivation

The k-tiles of a string are its factors (contiguous sub-words) of length k. The tiled version $\mathsf{tiled}_k(R)$ of a rewrite system R over Σ describes the action of R on tiled words. Since $\mathsf{tiled}_k(R)$ has a larger alphabet (namely, Σ^k), it may be easier to analyze:

Example 1.1 *For the rewriting system $R = \{aa \to aba\}$, we have $\mathsf{tiled}_2(R) = \{[aa] \to [ab, ba]\}$. It is easy to see that $\mathsf{tiled}_2(R)$ terminates, since each rule application reduces the number of occurrences of tile aa. The original system R does not admit such a proof of termination, since R does not remove any letters.*

2. Tiling Systems

A tiling system specifies a language by considering prefixes, factors, and suffixes of bounded length. We give an equivalent definition that allows a uniform description, using end markers $\lhd, \rhd \notin \Sigma$. A similar method is used for two-dimensional tiling [3].

Definition 2.1 *For $w \in \Sigma^*$, the k-bordered version is $\mathsf{bord}_k(w) = \lhd^{k-1} w \rhd^{k-1}$ over $\Sigma \cup \{\lhd, \rhd\}$. The k-tiled version $\mathsf{tiled}_k(w)$ is the string over Σ^k of all factors of length k, or ϵ in case $|w| < k$. Let $\mathsf{tiles}_k(w)$ denote $\mathsf{alphabet}(\mathsf{tiled}_k(w))$, the set of letters in $\mathsf{tiled}_k(w)$.*

Example 2.2 *$\mathsf{tiled}_2(\mathsf{bord}_2(abbb)) = \mathsf{tiled}_2(\lhd abbb \rhd) = [\lhd a, ab, bb, bb, b \rhd]$, $\mathsf{tiles}_2(\mathsf{bord}_2(abbb)) = \{\lhd a, ab, bb, b \rhd\}$, $\mathsf{tiles}_2(\mathsf{bord}_2(a)) = \{\lhd a, a \rhd\}$, $\mathsf{tiles}_2(\mathsf{bord}_2(\epsilon)) = \{\lhd \rhd\}$, and $\mathsf{tiled}_3(\mathsf{bord}_3(a)) = [\lhd \lhd a, \lhd a \rhd, a \rhd \rhd]$.*

The language defined by a set of tiles T of length k is $\{w \in \Sigma^* \mid \mathsf{tiles}_k(\mathsf{bord}_k(w)) \subseteq T\}$. This is an equivalent definition of the class of strictly locally k-testable languages [6, 8], a subclass of regular languages. We will use one-dimensional tiling systems to over-approximate reachability sets in string rewriting.

3. Rewriting and Reachability

A *string rewriting system* over alphabet Σ consists of rewrite rules. We use standard concepts and notation, with this extension: a *constrained rule* is a pair of strings l, r with a constraint $c \in \{\mathsf{factor}, \mathsf{suffix}\}$, indicating where the rule is to be applied. The rewrite relations are:

$$\to_{l,r,\mathsf{factor}} = \{(xly, xry) \mid x, y \in \Sigma^*\}, \qquad \to_{l,r,\mathsf{suffix}} = \{(xl, xr) \mid x \in \Sigma^*\},$$

A constrained rule (l, r, c) is denoted by $l \to_c r$. Standard rewriting corresponds to the factor constraint, therefore $\to$ abbreviates $\to_{\mathsf{factor}}$. For a rewrite system R, we define $\to_R$ as the union of the rewrite relations of its rules. For a relation ρ on Σ^* and a set $L \subseteq \Sigma^*$, let $\rho(L) = \{y \mid \exists x \in L, (x, y) \in \rho\}$. Hence the set of R-*reachable* strings from L is $\to_R^*(L)$, or $R^*(L)$ for short.

A language $L \subseteq \Sigma^*$ is *closed w.r.t.* R if $\to_R(L) \subseteq L$.

Example 3.1 *For* $R = \{cc \to_{\mathsf{factor}} bc, ba \to_{\mathsf{factor}} ac, c \to_{\mathsf{suffix}} bc, b \to_{\mathsf{suffix}} ac\}$ *, we have* $bbb \to_{\mathsf{suffix}} bbac \to_{\mathsf{factor}} bacc$*. The reachability set* $R^*(\{bc, ac\})$ *is* $(a+b)b^*c$*. This set is closed w.r.t.* R*.*

R over Σ is called *terminating on* $L \subseteq \Sigma^*$ if for each $w \in L$, each R-derivation starting at w is finite, and R is *terminating* if it is terminating on Σ^*.

4. Closures

Given a rewrite system R over alphabet Σ, a *closure* $C = (l, r)$ of R is a pair of strings with $l \to_R^+ r$ such that each position of r took part in some step of the derivation. In particular, we use *forward closures* [5]. Their right-hand sides can be computed by (factor and) suffix rewriting.

Proposition 4.1 *[4]* $\mathsf{RFC}(R) = (R \cup \mathsf{forw}(R))^*(\mathsf{rhs}(R))$*, where*

$$\mathsf{forw}(R) = \{l_1 \to_{\mathsf{suffix}} r \mid (l_1 l_2 \to r) \in R, l_1 \neq \epsilon \neq l_2\}.$$

They are related to termination by

Theorem 4.2 *[1]* R *is terminating (on* Σ^**) if and only if* R *is terminating on* $\mathsf{RFC}(R)$*.*

Example 4.3 *For* $R = \{cc \to bc, ba \to ac\}$ *we have* $\mathsf{forw}(R) = \{c \to_{\mathsf{suffix}} bc, b \to_{\mathsf{suffix}} ac\}$ *and* $\mathsf{RFC}(R) = (a+b)b^*c$*, cf. Example 3.1. As* $\mathsf{RFC}(R)$ *contains no* R*-redex,* R *is trivially terminating on* $\mathsf{RFC}(R)$*, therefore by Theorem 4.2,* R *is terminating.*

In the following, we use tiled rewriting to approximate $\mathsf{RFC}(R)$. This allows to obtain the termination proof of Example 4.3 automatically.

5. Tiled Rewrite Systems

We enlarge the alphabet of a rewrite system by tiling.

Definition 5.1 *For a rule $l \to_{\mathsf{factor}} r$ over Σ we define*

$$\mathsf{tiled}_k(l \to_{\mathsf{factor}} r) = \{\mathsf{tiled}_k(xly) \to_{\mathsf{factor}} \mathsf{tiled}_k(xry) \mid x \in \mathsf{tiles}_{k-1}(\lhd^*\Sigma^*), y \in \mathsf{tiles}_{k-1}(\Sigma^*\rhd^*)\}$$

and for a given set of tiles $S \subseteq \mathsf{tiles}_k(\Sigma^*)$

$$\mathsf{tiled}_S(l \to_{\mathsf{factor}} r) = \mathsf{tiled}_k(l \to_{\mathsf{factor}} r) \cap S^* \times S^* \times \{\mathsf{factor}\}.$$

Both tiled_S and tiled_k are extended to sets of rules.

Example 5.2 *$\mathsf{tiled}_2(ba \to_{\mathsf{factor}} ac)$ contains 16 rules, among them $[\lhd b, ba, a\rhd] \to [\lhd a, ac, c\rhd]$, $[\lhd b, ba, aa] \to [\lhd a, ac, ca], \ldots, [ab, ba, a\rhd] \to [aa, ac, c\rhd], \ldots, [cb, ba, ac] \to [ca, ac, cc]$. For $S = \{ac, ba, bb, cc\}$ we get $\mathsf{tiled}_S(ba \to_{\mathsf{factor}} ac) = \{[bb, ba, ac] \to [ba, ac, cc]\}$ and for any strict subset T of S, $\mathsf{tiled}_T(ba \to_{\mathsf{factor}} ac) = \emptyset$.*

Derivations w.r.t. R and $\mathsf{tiled}_k(R)$ are bi-similar, and we obtain

Theorem 5.3 *For $S \subseteq \mathsf{tiles}_k(\Sigma^*)$, if $\mathsf{Lang}(S)$ is closed w.r.t. R, then R is terminating on $\mathsf{Lang}(S)$ if and only if $\mathsf{tiled}_S(R)$ is terminating.*

Example 5.4 *(cont.) $R = \{cc \to bc, ba \to ac\}$. $\mathsf{RFC}(R) = \mathsf{Lang}(S)$ for the set of tiles $S = \{\lhd a, \lhd b, ab, ac, bb, bc, c\rhd\}$. The set $\mathsf{RFC}(R)$ is closed w.r.t. R by definition and $\mathsf{tiled}_S(R)$ is empty, therefore terminating. By Theorem 5.3, R is terminating on $\mathsf{RFC}(R)$ and by Theorem 4.2, R is terminating.*

We obtain a set of tiles for using Theorem 5.3 by the following algorithm.

Algorithm 5.5

- *Input: A rewrite system R over Σ, a set of tiles $T \subseteq \mathsf{tiles}_k(\Sigma^*)$.*
- *Output: A set of tiles $S \subseteq \mathsf{tiles}_k(\Sigma^*)$ such that $T \subseteq S$ and $\mathsf{Lang}(S)$ is closed w.r.t. R.*
- *Implementation: $S = \bigcup_i S_i$ for the sequence given by*

$$S_0 = T,\ S_{i+1} = S_i \cup \mathsf{alphabet}(\mathsf{rhs}(\mathsf{tiled}_k(R) \cap \mathsf{lhs}^{-1}(S_i^*))).$$

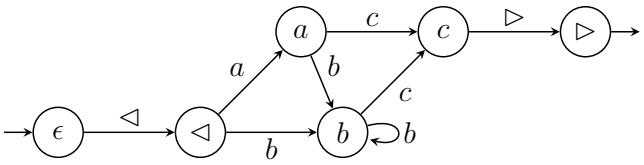
In each step, each rule is extended by contexts of length $k-1$ on both sides such that the extended left-hand side can be covered. Then the tiles of the extended right-hand side are added. The algorithm terminates since (S_i) is increasing w.r.t. $\subseteq$ and bounded by $\mathsf{tiles}_k(\Sigma^*)$.

6. Representing Tiling Systems by Automata

For an efficient implementation of the closure algorithm 5.5, we represent a set of tiles of length k by a deterministic (not necessarily complete or minimal) automaton over $\Sigma \cup \{\lhd, \rhd\}$ with states from $\lhd^{<k} \cup \mathsf{tiles}_{k-1}(\Sigma^*) \cup \{\rhd^{k-1}\}$, initial state ϵ and final state $\rhd^{k-1}$. For each transition $p \xrightarrow{c} q$, state q is the suffix of length $k-1$ of $p \cdot c$. Such an automaton A represents the set of tiles

$$\mathsf{tiles}(A) = \{p \cdot c \mid p \xrightarrow{c}_A q, |p| = k-1\}.$$

Example 6.1 *(Example 5.4 cont'd) This automaton represents* $\{\lhd a, \lhd b, ab, ac, bb, bc, c\rhd\}$*:*



Adding tiles in Algorithm 5.5 then corresponds to adding states and edges. With the automata representation, we can quickly check whether a left-hand side of a rule is covered by the current set of tiles. Our implementation can handle automata with 10^4 transitions (tiles) in a few seconds.

7. Discussion

We have presented a method to compute a regular over-approximation of reachability sets, using tiling systems, represented as automata, and we applied this to termination analysis. The root labeling method [7] corresponds to tiling on the full set Σ^*, for width 2. Our method allows any width, and restricts the set of tiles. Restriction to right-hand sides of forward closures (RFC) had already been applied to enhance the power of the matchbound termination proof method [2]. Our method decouples the RFC method from the matchbound method.

References

[1] NACHUM DERSHOWITZ, Termination of Linear Rewriting Systems. In: SHIMON EVEN, ODED KARIV (eds.), *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*. Lecture Notes in Computer Science 115, Springer, 1981, 448–458.

[2] ALFONS GESER, DIETER HOFBAUER, JOHANNES WALDMANN, Match-Bounded String Rewriting Systems. *Appl. Algebra Eng. Commun. Comput.* **15** (2004) 3-4, 149–171.

[3] DORA GIAMMARESI, ANTONIO RESTIVO, Two-Dimensional Languages. In: ARTO SALOMAA, GRZEGORZ ROZENBERG (eds.), *Handbook of Formal Languages*. 3, Springer, 1997, 215–267.

[4] MIKI HERMANN, *Divergence des systèmes de réécriture et schématisation des ensembles infinis de termes*. Habilitation, Université de Nancy, France, 1994.

[5] DALLAS S. LANKFORD, D. R. MUSSER, *A finite termination criterion*. Technical report, Information Sciences Institute, Univ. of Southern California, Marina-del-Rey, CA, 1978.

[6] ROBERT MCNAUGHTON, SEYMOUR PAPERT, *Counter-Free Automata*. MIT Press, 1971.

[7] CHRISTIAN STERNAGEL, AART MIDDELDORP, Root-Labeling. In: ANDREI VORONKOV (ed.), *Rewriting Techniques and Applications, 19th International Conference, RTA 2008,*

Hagenberg, Austria, July 15-17, 2008, Proceedings. Lecture Notes in Computer Science 5117, Springer, 2008, 336–350.

[8] YECHEZKEL ZALCSTEIN, Locally testable languages. *Journal of Computer and System Sciences* **6** (1972) 2, 151 – 167.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 87–90.

Unfair P Systems

Artiom Alhazov[(B)] Rudolf Freund[(A)] Sergiu Ivanov[(C)]

[(A)] Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Vienna, Austria
rudi@emcc.at

[(B)] Institute of Mathematics and Computer Science
Academiei 5, Chişinău, MD-2028, Moldova
artiom@math.md

[(C)] IBISC, Université Évry, Université Paris-Saclay
23 Boulevard de France, 91025, Évry, France
sergiu.ivanov@univ-evry.fr

Abstract

We consider variants P systems in which the application of rules in each step is controlled by a function on the applicable multisets of rules. Some examples are given to exhibit the power of this general concept. Moreover, for several well-known models of P systems we show how they can be simulated by P systems with a suitable fairness function.

1. (Hierarchical) P System with Fairness Function

A comprehensive overview of different variants of membrane systems and their expressive power is given in the handbook, see [4]. For a state of the art view of the domain, we refer the reader to the P systems website [6] as well as to the bulletin series of the International Membrane Computing Society [5].

In this paper we consider a new model of P systems – first introduced in [2] and then also published in [1] – in which the application of rules in each step is controlled by a function – the *fairness function* – on the multisets of rules applicable to the underlying configuration C.

A (generating) *hierarchical P system* (*P system* for short) is a construct

$$\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_o),$$

where O is the alphabet of objects, $T \subseteq O$ is the alphabet of terminal objects, μ is the membrane structure injectively labeled by the numbers from $\{1, \ldots, n\}$ and usually given by a sequence of correctly nested brackets, w_i are the multisets giving the initial contents of each membrane i ($1 \leq i \leq n$), R_i is the finite set of rules associated with membrane i ($1 \leq i \leq n$), and h_o is the label of the output membrane ($1 \leq h_o \leq n$).

A (generating) *hierarchical P system with fairness function* (*unfair P system* for short) is a construct $\Pi' = (\Pi, f)$ where Π is a P system and f is the fairness function defined for any configuration C of Π, the corresponding set $Appl_\delta(\Pi, C)$ of multisets of rules from Π applicable

to C in the given derivation mode δ, and any multiset of rules $R \in Appl_\delta(\Pi, C)$; we then use the values $f(C, Appl_\delta(\Pi, C), R)$ for all $R \in Appl_\delta(\Pi, C)$ to choose a multiset $R' \in Appl_\delta(\Pi, C)$ of rules to be applied to the underlying configuration C.

We here consider the rules associated with membranes to be multiset rewriting rules, also allowing symbols to be sent to neighboring membranes. In the maximally parallel derivation mode, multisets rules are applied in a non-extendable way. If only one copy of each rule can be used, we speak of the set maximally parallel mode. In the sequential mode, only one rule is applied. A computation of a P system is traditionally considered to be a sequence of configurations it successively can pass through, stopping at the halting configuration, i.e., no rule can be applied any more, in any membrane. The result of a computation of a P system Π is the contents of the output membrane h_o in a halting configuration, projected over the terminal alphabet T.

Example 1.1 *Consider the P system $\Pi_1 = (\{a, b\}, \{b\}, [\,_1\,]_1, a, R_1, 1)$ with the rule set $R_1 = \{1 : a \to aa, 2 : a \to b\}$. Both in the sequential as well as the maximally parallel derivation mode Π generates the whole of $\mathbb{N}_+$.*

Now consider the unfair P system $\Pi_2 = (\Pi_1, f_2)$ obtained by extending Π_1 with the fairness function f_2 defined as follows: if a rule is applied n times then it contributes to the function value of the fairness function f_2 for the multiset of rules with 4^n. In this unfair P system with one membrane working in the maximally parallel way, starting with the axiom a, we use the rule $1 : a \to aa$ in the maximal way k times thus obtaining 2^k symbols a. Then in the last step, for all a we use the rule $2 : a \to b$ thus obtaining 2^k symbols b. We cannot mix the two rules in one of the derivation steps as only the clean use of exactly one of them yields the maximal value for the fairness function.

We observe that the effect is similar to that of controlling the application of rules by the well-known control mechanism called label selection, e.g., see [3], where either the rule with label 1 or the rule with label 2 has to be chosen. We will return to this model in Section 2.2. □

The following weird example shows that the fairness function should be chosen from a suitable class of (at least recursive) functions, as otherwise the whole computing power comes from the fairness function:

Example 1.2 *Take the unfair P system Π_3 with one membrane working in the maximally parallel way, starting with the axiom a and using the three rules $1 : a \to aa$, $2 : a \to a$, and $3 : a \to b$. Moreover, let $M \subset \mathbb{N}_+$, i.e., an arbitrary set of positive natural numbers. The fairness function f_M on multisets of rules over these three rules and a configuration containing m symbols a is defined as follows: For any multiset of rules R containing copies of the rules $1 : a \to aa$, $2 : a \to a$, and $3 : a \to b$,*

- *$f_M(R) = 1$ if R only contains m copies of rule 3 and $m \in M$,*
- *$f_M(R) = 1$ if R only contains exactly one copy of rule 1 and the rest are copies of rule 2,*
- *$f_M(R) = 0$ for any other applicable multiset of rules.*

Again the choice is made by applying only multisets of rules which yield the maximal value $f_M(R) = 1$. If we use rule $1 : a \to aa$ once and rule $2 : a \to a$ for the rest, this increases the number of symbols a in the skin membrane by one. Thus, in $m - 1$ steps we get m symbols a. If m is in M, we now may use rule $3 : a \to b$ for all symbols a, thus obtaining m symbols b, and the system halts. In that way, the system generates exactly $\{b^m \mid m \in M\}$. To make this example a

little bit less weird, we may only allow computable sets M. Still, the whole computing power is in the fairness function f_M alone, with f_M only depending on the multiset of rules. □

2. Simulation Results

In this section, we show two general results. The first one describes how priorities can be simulated by a suitable fairness function in P systems of any kind working in the sequential mode. The second one exhibits how P systems with rule label control, see [3], can be simulated by suitable unfair P systems for any arbitrary derivation mode.

2.1. Simulating Priorities in the Sequential Derivation Mode

In the sequential derivation mode, exactly one rule is applied in every derivation step of the P system Π. Given a configuration C and the set of applicable rules $Appl(\Pi, C)$ not taking into account a given priority relation $<$ on the rules, we define the fairness function to yield 1 for each rule in $Appl(\Pi, C)$ for which no rule in $Appl(\Pi, C)$ with higher priority exists, and 0 otherwise. Thus, only a rule with highest priority can be applied.

Theorem 2.1 *Let $(\Pi, <)$ be a P system of any kind with the priority relation $<$ on its rules and working in the sequential derivation mode. Then there exists an unfair P system (Π, f) with the fairness function f simulating the computations in $(\Pi, <)$ selecting the multisets of rules with maximal values.*

Proof. First we observe that the main ingredient Π is exactly the same in both $(\Pi, <)$ and (Π, f), i.e., we only replace the priority relation $<$ by the fairness function f, which here not only depends on $\{r\}$, but also on $Appl(\Pi, C)$):

- $f(Appl(\Pi, C), \{r\}) = 1$ if and only if there exists no rule $r' \in Appl(\Pi, C)$ such that $r < r'$, and
- $f(Appl(\Pi, C), \{r\}) = 0$ if and only if there exists a rule $r' \in Appl(\Pi, C)$ such that $r < r'$.

We now define the task of f as choosing only those rules with maximal value, i.e., a rule r can be applied to configuration C if and only if $f(Appl(\Pi, C), \{r\}) = 1$. □

2.2. Simulating Label Selection

In P systems with label selection only rules belonging to one of the predefined subsets of rules can be applied to a given configuration, see [3], i.e., we label all the rules in the sets $R_1, \ldots, R_m$ in a one-to-one manner by labels from a set H and then, in any derivation step, we take a set W containing subsets of H. A *P system with label selection* is a construct

$$\Pi^{ls} = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_i, h_o, H, W),$$

where $\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_o)$ is a P system as in Section 1, H is a set of labels for the rules in the sets $R_1, \ldots, R_m$, and $W \subseteq 2^H$. In any transition step in Π^{ls} we first select a set of labels $U \in W$ and then apply a non-empty multiset R of rules applicable in the given derivation mode restricted to rules with labels in U. The following proof exhibits how the fairness function can also be used to capture the underlying derivation mode.

Theorem 2.2 *Let (Π, H, W) be a P system with label selection using any kind of rules in any kind of derivation mode. Then there exists an unfair P system (Π', f) with fairness function f simulating the computations in (Π, H, W) with f selecting the multisets of rules with maximal values.*

Proof. By definition, in the P system (Π, H, W) with label selection a multiset of rules can be applied to a given configuration only if all the rules have labels in a selected set of labels $U \in W$. We now consider the set of all multisets of rules applicable to a configuration C, denoted by $Appl_{asyn}(\Pi, C)$, as it corresponds to the asynchronous derivation mode (abbreviated $asyn$); from those we select all R which obey to the label selection criterion, i.e., there exists a $U \in W$ such that the labels of all rules in R belong to U, and then only take those which also fulfill the criteria of the given derivation mode restricted to rules with labels from U.

Hence we define (Π', f) by taking $\Pi' = \Pi$ and, for any derivation mode δ, f_δ for any multiset of rules $R \in Appl_{asyn}(\Pi, C)$ as follows:

- $f_\delta(C, Appl_{asyn}(\Pi, C), R) = 1$ if there exists a $U \in W$ such that the labels of all rules in R belong to U, and, moreover, $R \in Appl_\delta(\Pi_U, C)$, where Π_U is the restricted version of Π only containing rules with labels in U, as well as
- $f_\delta(C, Appl_{asyn}(\Pi, C), R) = 0$ otherwise.

According to our standard selection criterion, we choose only those multisets of rules where the fairness function yields the maximal value 1, i.e., those R such that there exists a $U \in W$ such that the labels of all rules in R belong to U and R is applicable according to the underlying derivation mode with rules restricted to those having a label in U, which exactly mimicks the way of choosing R in (Π, H, W). Therefore, in any derivation mode δ, (Π', f_δ) step by step simulates the derivations in (Π, H, W) and thus yields the same computation results. □

References

[1] A. ALHAZOV, R. FREUND, S. IVANOV, P systems and the concept of fairness. In: S. COJOCARU, C. GAINDRIC, D.I.. DRUGUS (eds.), *Proceedings of the Conference on Mathematical Foundations of Informatics MFOI2017, November 9-11, 2017, Chişinău, Republic of Moldova*. 2017, 7–26.

[2] A. ALHAZOV, R. FREUND, S. IVANOV, Unfair P systems. In: C. GRACIANI, GH. PĂUN, A. RISCOS-NÚÑEZ, L. VALENCIA-CABRERA (eds.), *Proceedings BWMC 2017*. Fénix Editora, 2017, 1–12.

[3] R. FREUND, M. OSWALD, GH. PĂUN, Catalytic and purely catalytic P systems and P automata: control mechanisms for obtaining computational completeness. *Fundamenta Informaticae* **136** (2015) 1-2, 59–84.
https://doi.org/10.3233/FI-2015-1144

[4] GH. PĂUN, G. ROZENBERG, A. SALOMAA, *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.

[5] Bulletin of the International Membrane Computing Society (IMCS). http://membranecomputing.net/IMCSBulletin/index.php.

[6] The P Systems Website. http://ppage.psystems.eu/.

THEORIE-TAGE 2018
K. Reinhardt (Herausgeber): Theorietage 2018, Wittenberg, 24. – 27.9.2018
Universität Halle-Wittenberg, Technischer Bericht 18-1, S. 91–94.

Half-Terminal Grammars (HTG): A Formal Two-stage Structured String Derivation and Interpretation System

Dominikus Heckmann[A]

[A]Ostbayerische Technische Hochschule Amberg-Weiden,
Kaiser-Wilhelm-Ring 22, 92224 Amberg
d.heckmann@oth-aw.de

Abstract

Early string rewriting systems, or *Semi-Thue-Systems*, come along with one alphabet. Quoting [1]: *The simple artifice of partitioning the alphabet in terminals and non-terminals is a powerful one; it allows the definition of the Chomsky hierarchy based on what combination of terminals and non-terminals rules contain. This was a crucial development in the theory of formal languages.*

In this paper we look at the idea of partitioning the alphabet (of regular and context-free grammars) into three parts instead of two, namely into *terminals*, *non-terminals* and *half-terminals*. We call this concept *half-terminal*, see [4], since it acts in its intended usage of a two-stage process half of the time (during the derivation stage) in a terminal functionality and half of the time (during the reading stage) in a structure-preserving functionality:

$$S \in V_N \overset{derive}{\Longrightarrow}_G^* w_1 \in (V_H \cup V_T)^* \overset{read}{\longmapsto}_R^? w_2 \in {V_T}^*$$

The main intension behind this approach is to preserve the underlying structure, and to find out how much *mildly context-sensitivity* can be obtained by this approach with non-context-sensitive grammars. *Half-Terminal Grammars* are related to *Linear Context-Free Rewriting Systems* [6], and furthermore they form a revision of *Linear Structure Grammars*, as introduced in [4]. In this paper we present a formal definition of HTGs and some examples of mildly context-sensitive languages.

1. Introduction

We consider the described formalism mainly as a fine tool with which we try to bring a variety of existing grammar formalisms closer together on their structural layer and not only on the layer of their generated formal languages. We have in mind grammars like Tree-Adjoining Gramas (Joshi), Recursive Matrix Systems (Becker, Heckmann), Coupled Context-Free Grammars (Hotz) and [5], and further mildly context-sensitive grammars with their underlying, sometimes hidden, structures like trees, matrixes, nested lists, etc. The proposed framework seems not to be a cooperating distributed (CD) grammar system, see [3], since only the first stage forms a grammar while the second stage forms a (possible recursively defined) function.

[A]Ganz herzlichen Dank an die *Ostbayerische Technische Hochschule* für den gewährten Freiraum

Half-Terminal Grammars (HTG) form a direct, straight-forward, structure preserving extension of basic chomsky grammars. One feature is that some of the grammar functionality that is normally executed in the production rules attached to the non-terminals can now be shifted towards the structure interpretation part that can be attached to the terminals. We add a third alphabet V_H to the two existing disjoint alphabets V_N and V_T. Furthermore we add a second stage to erase these half-terminals again.

An example for a structured string word is $\langle[a.b.c][a.b.c][a.b.c][\epsilon.\langle[d.e.f][d.e.f][d.e.f]\rangle.\epsilon]\rangle$ It consists of the known terminal symbols $a, b, c, d, e, \ldots$ and some other symbols "$\langle$", "[", ".", "]", "$\rangle$" that are neither terminals nor nonterminals. They represent structural information of this object that could also be visualized graphically, for example as shown in figure 1:

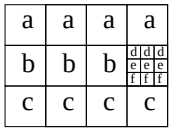


Figure 1: one possible realization and visualization of the underlying structure, source from [2]

As we mentioned before, the half-terminal grammars consist of two generative parts: in the *derivation part* structured words as elements of $(V_T \cup V_H)^*$ are generated. In the *interpretation part* the *structured words* are transformed into pure *terminal strings* as elements of $(V_T)^*$. We think of a restricted transformation without copying and without deleting any terminal sign. It is more like a sorting or reading in a pre-defined order. As a convention the non-terminals are represented by upper-case letters like S, A, B etc., the terminals are represented by lower-case letters like a, b, c, and the half-terminals are represented by punctuation marks like parentheses, dots, commas like $\langle -[.]\rangle$ or numbers.

2. Definitions

Definition 2.1 *Let V be an alphabet, divided into three disjoint partitions*

- *V_N, the set of non-terminals, also denoted as N,*
- *V_H, the set of half-terminals,*
- *V_T, the set of terminals, also denoted as T.*

A half-terminal grammar is the six-tuple construct $HTG = (V_N, V_H, V_T, P, S, R)$, where

- *G = $(V_N, V_H \cup V_T, P, S)$ is a grammar, either regular, linear or context-free, where S is the start symbol, and P is the set of production rules, restricted according to the chosen grammar with being context-free if $P \subset V_N \times (V_N \cup V_H \cup V_T)^*$*
- *R is a restricted partial read-function $(V_H \cup V_T)^* \longmapsto (V_T)^*$, that guides the interpretation step, which is realized as a kind of 'reading-out' the final terminals. The concrete realization of the definition of R is highly dependent from the underlying structure of the concrete application.*

If we further abbreviate $T^{\#}$ = $(V_H \cup V_T)$, we could also denote the grammar G by the four tuple $G = (N, T', P, S)$ to underline the backward compatibility. Thus we are able to denote an Half-Terminal Grammar by a tuple HTG = (G, R).

The following definition looks at the derivation relation

Definition 2.2 *Let HTG =* (G, R) *be a half-terminal grammar*

- *The direct derivation relation induced by G, denoted* $\Longrightarrow^1_G$*, is a binary relation between sentential forms. It is defined as:*

$$\alpha u\beta \Longrightarrow^1_G \alpha v\beta \quad \textit{iff} \quad (u,v) \in P, \textit{ and } \alpha u\beta,\ \alpha v\beta \in V^*.$$

- *The derivation relation induced by G, denoted* $\Longrightarrow^*_G$*, is the reflexive and transitive closure of the relation* $\Longrightarrow^1_G$*.*

The following definition looks at the two different languages that can be defined, after the two different stages of the generation process.

Definition 2.3 *Let HTG be a half-terminal grammar*

- *The language generated by G=* $(V_N, V_H \cup V_T, P, S)$ *is called structured string language:*

$$L_G = \{w_1 \mid w_1 \in (V_H \cup V_T)^*, S \Longrightarrow^*_G w_1\}$$

- *The language generated by* HTG *is called terminal string language:*

$$L_{HTG} = \{w_2 \mid w_2 \in {V_T}^*, w_1 \in (V_T \cup V_H)^*, S \Longrightarrow^*_G w_1, v \longmapsto_R w_2\}$$

We introduce the following abbreviation: $T^{\#}$:= $(V_H \cup V_T)^*$, which denotes the set of structured string words that contains terminals as well as half-terminals.

To summarize the idea of this paper in a simplified formal notation: $S \overset{G}{\Longrightarrow} \mathbf{T}^{\#} \overset{R}{\longmapsto} T^*$

3. Examples

Example 3.1 *COUNT-3*

Let HTG_1 be defined by

- $V_N = \{S, A\}$, $V_T = \{a, b, c\}$, $V_H = \{\rangle, \langle,], [\}$, $S = S$,
  $P_1 = \{S \to \langle A\rangle,\ A \to [abc],\ A \to [abc]A\}$,
  $R:\ \langle[\alpha_1\beta_1\gamma_1][\alpha_2\beta_2\gamma_2]...\rangle \longmapsto \alpha_1\alpha_2...\beta_1\beta_2...\gamma_1\gamma_2...$
- A possible derivation with interpretation:
  derive (Stage 1): $S \Rightarrow \langle A\rangle \Rightarrow \langle[abc]A\rangle \Rightarrow \langle[abc][abc]A\rangle \Rightarrow \langle[abc][abc][abc]\rangle$
  read (Stage 2): $\langle[abc][abc][abc]\rangle \longmapsto_R aaabbbccc$
- G_1 generates the structure of a *list of triples* in $T^{\#}$.
- R_1 reorders the terminals in such a way that the first elements of the triples form a string, also the second and third elements, and these three strings are catenated.
- The generated terminal language is $L_{HTG_1} = \{a^n b^n c^n \mid n \geq 1\}$.

Example 3.2 *COPY*

Let HTG_2 be defined by
- $V_N = \{S, A\}$, $V_T = \{a, b, c\}$, $V_H = \{\rangle, \langle,], [, .\}$, $S = S$,
  $P_2 = \{S \to \langle A\rangle,\ A \to [a.a]A,\ A \to [b.b]A,\ A \to [c.c]A,\ A \to [\epsilon.\epsilon]\}$,
  $R_2 : \langle [\alpha_1.\beta_1][\alpha_2.\beta_2]...\rangle) \longmapsto \alpha_1\alpha_2...\beta_1\beta_2...$
- A possible derivation with realization-interpretation:
  $S \Rightarrow \langle A\rangle \Rightarrow \langle [bb]A\rangle \Rightarrow \langle [bb][cc]A\rangle \Rightarrow \langle [bb][cc][cc]\rangle \longmapsto_R bccbcc$
- G_2 generates the structure of a *list of tuples* in $T^{\#}$.
- R_2 reorders the terminals in the way that the first and the second elements of the tuples form a string each, that are con catenated. The intermediate half-terminals are deleted.
- The generated terminal language is $L_{HTG_2} = \{ww \mid w \in T^*\}$.

4. Conclusion and Outlook

This paper presents the idea of half-terminals, it presents the idea of enriching the terminal part within the rewriting process of formal grammars. Furthermore, HTG as a revised grammar formalism, has been defined while leaving the read-realization-function slightly underspecified. We expect and hope to come up with partial R-functions that suit well the corresponding parsing algorithms for many recursively defined $T^{\#}$ structures. We are happy with the compactness of the grammar formalism: if you look at example 3.1 for a moment, you can easily image how the grammar should be extended to implement the count-5-language for example, while other existing mildly-context-sensitive grammar formalisms needed more complex production rules to generate this language. However, one of the biggest advantages that we envision with *half-terminal grammars* is the fact, that off-the-shelf grammars can be used at stage one. Thus all existing definitions, research results, parsers, implementations etc. can be reused at once.

In the next step, we will look for efficient integrated structure parsing algorithms for $T^{\#}$.

References

[1] Wikipedia: Semi-Thue system. `https://en.wikipedia.org/wiki/Semi-Thue_system`. Accessed: 2018-09-09.

[2] T. BECKER, D. HECKMANN, Recursive matrix systems (RMS) and TAG. In: *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*. Philadelphia, PA, USA, 1998, 9–12.

[3] J. DASSOW, GH. PĂUN, G. ROZENBERG, Grammar Systems. In: G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Langauges*. 2, Springer-Verlag, Berlin Heidelberg, 1997, 155–213.

[4] D. HECKMANN, *Recursive Matrix Systems*. Diplomarbeit, Saarland University, 1999.

[5] SEKI, MATSUMURA, FUJII, KASAMI, On multiple context-free grammars. *TCS: Theoretical Computer Science* **88** (1991) 1.

[6] D. WEIR, *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Dissertation, University of Pennsylvania, 1988.