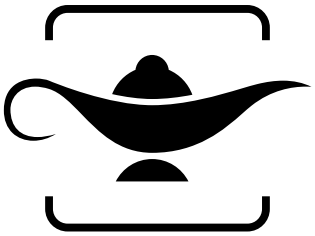
THEORIE-AG 2023

@

RPTU



Foreword

The "Theorietag Automaten und Formale Sprachen" is the annual meeting of the special interest group Automata and Formal Languages of the Gesellschaft für Informatik e.V. (German Informatics society) and is organized since 1991 by members of the special interest group in Germany, Austria, and Czechia. Since 1996, it also includes a set of invited talks. Moreover, the Theorietag includes the annual meeting of the representatives of the special interest group.

In 2023, the Theorietag will be organized by the Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (RPTU) and the Max Planck Institute for Software Systems (MPI-SWS) in Kaiserslautern. It will take place on October 4-6 in the building of the MPI-SWS. The invited talks are given by

- Christoph Haase (University of Oxford),
- Sandra Kiefer (University of Oxford),
- Daniel Neider (Technische Universität Dortmund),
- Joël Ouaknine (MPI-SWS), and
- Anne-Kathrin Schmuck (MPI-SWS).

Moreover, there will be 20 submitted talks. This proceedings booklet contains (extended) abstracts on all 25 talks.

We are grateful to the invited speakers and all other participants for contributing a talk. Furthermore, we thank the Gesellschaft für Informatik, MPI-SWS, and RPTU for generously supporting Theorietag. Moreover, we especially thank Pascal Bergsträßer, Khushraj Madnani, and Chris Köcher for helping with the organization! We wish all participants an inspiring and pleasant stay in Kaiserslautern.

Kaiserslautern, Oct 2nd, 2023

Anthony W. Lin
Georg Zetzsche

Contents

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 3–3.

RPTU

Automata giving small certificates for big solutions.

Christoph Haase

University of Oxford
christoph.haase@cs.ox.ac.uk

This talk will survey recent results and underlying techniques showing that finite-state automata and generalizations thereof allow to decide existential formulas of various extensions of Presburger arithmetic, the first-order theory of the integers with addition and order. Concretely, we will see an NP upper bound for existential Büchi arithmetic based on finite-state automata, and an EXPSPACE upper bound for Semenov arithmetic based on affine vector addition systems with states. I will also briefly touch some very recent work showing that existential Presburger arithmetic augmented with gcd constraints is decidable in NP.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 4–4.

Properties of Polyregular Functions.

Sandra Kiefer

University of Oxford
sandra.kiefer@cs.ox.ac.uk

Regular functions are a well-studied robust class of string-to-string functions, one of whose characterisations is that they are exactly the functions recognisable by two-way deterministic finite automata with output. This implies that their growth rate — i.e. the function describing the output length in terms of the input length — is always linear. To go beyond linear growth, one can equip the two-way automata with multiple reading heads (pebbles), the number of which then still constitutes a bound on the degree of the polynomial describing the growth rate. The functions recognised by these pebble automata are called polyregular.

Over the past years, the properties of polyregular functions have been studied intensively and various equivalent characterisations have been found. In the talk, I will give an introduction to the realm of polyregular functions by discussing some of those characterisations, with a focus on the logical one. I will also present simple constructions which refute that the aforementioned link between the growth exponent and number of heads is symmetric. That is, in general, the degree of the growth rate of a polyregular function is not (equal to or even a bound on) the minimum number of pebbles needed in an automaton to compute the function.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 5–5.

RPTU

Reinforcement Learning with Reward Machines.

Daniel Neider

TU Dortmund
daniel.neider@cs.tu-dortmund.de

Despite its great success, reinforcement learning struggles when the reward signal is sparse and temporally extended (e.g., in cases where the agent has to perform a complex series of tasks over a prolonged period of time). To expedite the learning process in such situations, a particular form of finite-state machines, called reward machines, has recently been shown to help immensely. However, designing a proper reward machine for the task at hand is challenging and remains a tedious and error-prone manual task.

In this presentation, we will survey recent approaches that intertwine reinforcement learning and the inference of reward machines, thereby eliminating the need to craft a reward machine by hand. At their heart, these methods transform the inference task into a series of constraint satisfaction problems that can be solved using off-the-shelf SAT and SMT solvers. We will see how this idea can be used to integrate user-provided advice into the learning process and how it deals with stochastic reward signals. Moreover, we will briefly discuss theoretical properties and hint at empirical evidence demonstrating that reinforcement learning with reward machines outperforms existing methods, such as hierarchical and deep reinforcement learning.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 6–6.

RPTU

What's Decidable about Discrete Linear Dynamical Systems?

Joël Ouaknine[(A)]

[(A)]Max Planck Institute for Software Systems, Saarland Informatics Campus, Germany
joel@mpi-sws.org

Discrete linear dynamical systems (LDS) are a fundamental modelling paradigm in several branches of science, and have been the subject of extensive research for many decades. Within Computer Science, LDS are used, for example, to analyse linear loops and Markov chains, and also arise in areas such as automata theory, differential privacy, control theory, and the study of formal power series, among others.

Perhaps surprisingly, many decision problems concerning LDS (such as reachability of a given hyperplane, known as the *Skolem Problem*) have been open for many decades. In this talk, we explore the landscape of such problems, focussing in particular on model-checking questions.

Also affiliated with Keble College, Oxford as emmy.network Fellow, and supported by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science).

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 7–7.

RPTU

The Power of Feedback in a Cyber-Physical World.

Anne-Kathrin Schmuck

Max Planck Institute for Software Systems
akschmuck AT mpi-sws.org

Feedback allows systems to seamlessly and instantaneously adapt their behavior to their environment and is thereby the fundamental principle of life and technology – it lets animals breathe, it stabilizes the climate, it allows airplanes to fly, and the energy grid to operate. During the last century, control technology excelled at using this power of feedback to engineer extremely stable, robust, and reliable technological systems. With the ubiquity of computing devices in modern technological systems, feedback loops become cyber-physical – the laws of physics governing technological, social or biological processes interact with (cyber) computing systems in a highly nontrivial manner, pushing towards higher and higher levels of autonomy and self-regulation. While stability, reliability and robustness remain to be of uppermost importance in these systems, a control-inspired utilization of cyber-physical feedback loops for this purpose is lacking far behind. In this talk, I will discuss how a control-inspired view on formal methods for reliable software design can enable us to utilize the power of feedback for robust and adaptable cyber-physical system design.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 8–12.

RPTU

On the Satisfiability of Context-free String Constraints with Subword-Ordering

C. Aiswarya[(A)] Soumodev Mal[(B)] Prakash Saivasan[(C)]

[(A)]Chennai Mathematical Institute and CNRS IRL ReLaX, India
aiswarya@cmi.ac.in

[(B)]Chennai Mathematical Institute, India
soumodevmal@cmi.ac.in

[(C)]The institute of Mathematical Sciences, HBNI and CNRS IRL ReLaX, India
prakashs@imsc.res.in

Short abstract We consider a variant of string constraints given by membership constraints in context-free languages and subword relation between variables. The satisfiability problem for this variant turns out to be undecidable. We consider a fragment in which the subword-order constraints do not impose any cyclic dependency between variables. We show that this fragment is nexptime-complete. As an application of our result, we settle the complexity of control state reachability in acyclic lossy channel pushdown systems, which was shown to be decidable in Atig-Bouajjani-Touilli-08. We show that this problem is nexptime-complete.

This work is published in the proceedings of LICS 2022 [6].

Long abstract The study of string constraints has been at the center of research for many decades now, foremost being the seminal work of Makanin [19]. The problem is of particular interest due its close connections to the Hilbert's tenth problem [20]. There have been several studies of string constraints (as word equations). While the decidability status of the general word equation when it includes length constraints is still open [13], the satisfiability of word equations without the length constraints was shown to be decidable by [19]. Subsequently there have been several attempts to simplify the proof and pin down the precise complexity of the problem [22, 21].

In the recent times, the topic has gained much momentum due to its applicability to identifying security vulnerabilities in programs [26, 23, 4, 25, 24]. The fact that almost every program manipulates strings, especially in very diverse ways does not allow for a uniform way to analyse these programs for security vulnerabilities arising out of string processing. There have been many works in this direction, each accounting for different sets of string manipulations and comparisons. In fact there are several string constraint solvers that have been successfully build and employed to this effect [17, 8, 1, 16, 15]. Our work [6] is also an effort in this direction.

One important aspect of checking security vulnerability is to verify if the input to a program is safe. For example the SQL injection attack masquerades program code as an SQL query.

This operation usually involves relating two strings that arise out of programs. To this effect, an interesting class of string constraints is given by 1) membership constraints – which confines the domain of each variable to a class of language and 2) relational constraints – which allows for comparisons between the variables. The problem of interest here is satisfiability which asks whether there is an assignment to the variables that satisfies the constraints.

This formalism has turned out to be quite useful for the modeling capabilities and has been well studied. While most of the work in literature, mostly confines the membership constraints to regular languages, several comparison operations have been considered. Some of them being ReplaceAll [11, 12], relations due to transducers [18, 14], transducer with length constraints [3].

While these kind of models enjoy a very high expressive power, they are often plagued by undecidability. An ongoing research has been to identify subclasses which recover decidability for the satisfiability problem. For example, the straight-line fragment in [11] imposes an acyclicity requirement among the relational constraints between the variables to obtain decidability.

In this work [6] we consider a class of string constraints, given by 1) membership constraints – which confines the domain of each variable to a *context-free language* and 2) relational constraints which relate variables by the subword-order. As far as our knowledge goes, this is the first attempt to include a context-free language in the membership constraint. We believe this is useful and interesting since vulnerable inputs to programs include strings that are generated by programs or are programs themselves [10]. In both of these cases, the generated string can be a context-free language.

While it is possible to recover equality checking through subword relation, we show that the satisfiability problem is undecidable in the presence of cyclic dependencies between variables. We then consider a subclass called the acyclic fragment, inspired by [11, 3]. We show that satisfiability checking under this assumption is nexptime complete. In fact we show that the problem is already nexptime hard when the membership constraints are given as regular languages. Towards the nexptime algorithm, we show that our model enjoys a small model property. That is if the given constraints is satisfiable then it is satisfiable by an assignment of a bounded size. While this technique is not new, the presence of context-free membership constraints makes the problem harder. Towards this, we derive new insights about the combinatorial structure of the parse trees of a context-free grammar embedding a given word as a subword.

As an important application of our result, we derive a complexity upper bound for acyclic lossy channel systems (introduced in [7]). Lossy channel systems are an important model of distributed systems where finite-state processes communicate via point-to-point message transmission over an unbounded channel. When the channels are assumed to be reliable, the control state reachability is undecidable [9]. However, when the channels are assumed to be lossy, control state reachability becomes decidable[2]. Even with lossy channels, if the processes are assumed to be pushdown, the control state reachability problem is undecidable[7, 5]. If the communication topology is assumed to be acyclic, this was shown to be decidable in [7], but no elementary upper bound was known.

We establish strong connection between the acyclic fragment of the subword-ordering string constraints and acyclic lossy channel systems [7]. This also allows us to settle the complexity of control state reachability in acyclic lossy channel systems which has been open for more than a decade. We show that this problem is decidable in elementary time, in fact in nexptime, and supplement the result with a matching lower bound.

References

[1] A. P. ABDULLA, F. M. ATIG, Y.-F. CHEN, D. P. BUI, L. HOLÍK, A. REZINE, P. RUMMER, Trau : SMT solver for string constraints. In: *Proceedings of the 18th Conference on Formal Methods in Computer-Aided Design*. FMCAD Inc., 2019, 165–169.

[2] P. ABDULLA, B. JONSSON, Verifying programs with unreliable channels. In: *Proceedings of 8th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, Los Alamitos, CA, USA, 1993.

[3] P. A. ABDULLA, M. F. ATIG, B. P. DIEP, L. HOLÍK, P. JANKU, Chain-Free String Constraints. In: Y. CHEN, C. CHENG, J. ESPARZA (eds.), *Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings*. Lecture Notes in Computer Science 11781, Springer, 2019.

[4] O. C. ABIKOYE, A. ABUBAKAR, A. H. DOKORO, O. N. AKANDE, A. A. KAYODE, A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *EURASIP Journal on Information Security* **2020** (2020) 1.

[5] C. AISWARYA, On Network Topologies and the Decidability of Reachability Problem. In: C. GEORGIOU, R. MAJUMDAR (eds.), *Networked Systems - 8th International Conference, NETYS 2020, Marrakech, Morocco, June 3-5, 2020, Proceedings*. Lecture Notes in Computer Science 12129, Springer, 2020, 3–10.

[6] C. AISWARYA, S. MAL, P. SAIVASAN, On the Satisfiability of Context-free String Constraints with Subword-Ordering. In: C. BAIER, D. FISMAN (eds.), *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*. ACM, 2022, 6:1–6:13.
`https://doi.org/10.1145/3531130.3533329`

[7] M. F. ATIG, A. BOUAJJANI, T. TOUILI, On the Reachability Analysis of Acyclic Networks of Pushdown Systems. In: F. VAN BREUGEL, M. CHECHIK (eds.), *Concurrency Theory, 19th International Conference, CONCUR 2008, Toronto, Canada, August 19-22, 2008. Proceedings*. Lecture Notes in Computer Science 5201, Springer, 2008.

[8] M. BERZISH, V. GANESH, Y. ZHENG, Z3str3: A String Solver with Theory-Aware Heuristics. In: *Proceedings of the 17th Conference on Formal Methods in Computer-Aided Design*. FMCAD '17, FMCAD Inc, Austin, Texas, 2017.

[9] D. BRAND, P. ZAFIROPULO, On Communicating Finite-State Machines. *J. ACM* **30** (1983) 2.

[10] C. CADAR, V. GANESH, P. M. PAWLOWSKI, D. L. DILL, D. R. ENGLER, EXE: Automatically Generating Inputs of Death. *ACM Trans. Inf. Syst. Secur.* **12** (2008) 2.

[11] T. CHEN, Y. CHEN, M. HAGUE, A. W. LIN, Z. WU, What is decidable about string constraints with the ReplaceAll function. *Proc. ACM Program. Lang.* **2** (2018) POPL, 3:1–3:29.

[12] T. CHEN, M. HAGUE, A. W. LIN, P. RÜMMER, Z. WU, Decision procedures for path feasibility of string-manipulating programs with complex operations. *Proc. ACM Program. Lang.* **3** (2019) POPL.

[13] V. GANESH, M. MINNES, A. SOLAR-LEZAMA, M. C. RINARD, Word Equations with Length Constraints: What's Decidable? In: A. BIERE, A. NAHIR, T. E. J. VOS (eds.), *Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers*. Lecture Notes in Computer Science 7857, Springer, 2012.

[14] L. HOLÍK, P. JANKU, A. W. LIN, P. RÜMMER, T. VOJNAR, String constraints with concatenation and transducers solved efficiently. *Proc. ACM Program. Lang.* **2** (2018) POPL.

[15] S. KAN, A. W. LIN, P. RÜMMER, M. SCHRADER, CertiStr: a certified string solver. In: A. POPESCU, S. ZDANCEWIC (eds.), *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022*. ACM, 2022.

[16] A. KIEZUN, V. GANESH, S. ARTZI, P. J. GUO, P. HOOIMEIJER, M. D. ERNST, HAMPI: A Solver for Word Equations over Strings, Regular Expressions, and Context-Free Grammars. *ACM Trans. Softw. Eng. Methodol.* **21** (2013) 4.

[17] T. LIANG, A. REYNOLDS, N. TSISKARIDZE, C. TINELLI, C. BARRETT, M. DETERS, An Efficient SMT Solver for String Constraints. *Form. Methods Syst. Des.* **48** (2016) 3.

[18] A. W. LIN, P. BARCELÓ, String solving with word equations and transducers: towards a logic for analysing mutation XSS. In: R. BODÍK, R. MAJUMDAR (eds.), *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*. ACM, 2016, 123–136.

[19] G. S. MAKANIN, The problem of solvability of equations in a free smigroup. *Mathematics of the USSR-Sbornik* **32(2)** (1977).

[20] Y. V. MATIYASEVICH, A connection between systems of words-and-lengths equations and Hilbert's tenth problem. *Studies in constructive mathematics and mathematical logic. Part II,Zap. Nauchn. Sem. LOMI* **8** (1968).

[21] W. PLANDOWSKI, Satisfiability of Word Equations with Constants is in PSPACE. In: *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*. IEEE Computer Society, 1999, 495–500.

[22] W. PLANDOWSKI, An efficient algorithm for solving word equations. In: J. M. KLEINBERG (ed.), *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. ACM, 2006.

[23] S. SON, K. S. MCKINLEY, V. SHMATIKOV, Diglossia: Detecting Code Injection Attacks with Precision and Efficiency. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13, Association for Computing Machinery, New York, NY, USA, 2013.

[24] M. TRINH, D. CHU, J. JAFFAR, S3: A Symbolic String Solver for Vulnerability Detection in Web Applications. In: G. AHN, M. YUNG, N. LI (eds.), *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. ACM, 2014.

[25] M. TRINH, D. CHU, J. JAFFAR, Progressive Reasoning over Recursively-Defined Strings. In: S. CHAUDHURI, A. FARZAN (eds.), *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*. Lecture Notes in Computer Science 9779, Springer, 2016.

[26] T.-Y. WU, J.-S. PAN, C.-M. CHEN, C.-W. LIN, Towards SQL Injection Attacks Detection Mechanism Using Parse Tree. In: H. SUN, C.-Y. YANG, C.-W. LIN, J.-S. PAN, V. SNASEL, A. ABRAHAM (eds.), *Genetic and Evolutionary Computing*. Springer International Publishing, 2015.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 13–13.

RPTU

Priority Downward Closures

Ashwani Anand[(A)] Georg Zetzsche[(B)]

[(A)]MPI-SWS, Kaiserslautern
ashwani@mpi-sws.org

[(B)]MPI-SWS, Kaiserslautern
georg@mpi-sws.org

When a system sends messages through a lossy channel, then the language encoding all sequences of messages can be abstracted by its downward closure, i.e. the set of all (not necessarily contiguous) subwords. This is useful because even if the system has infinitely many states, its downward closure is a regular language. However, if the channel has congestion control based on priorities assigned to the messages, then we need a finer abstraction: The downward closure with respect to the priority embedding. As for subword-based downward closures, one can also show that these priority downward closures are always regular.

While computing finite automata for the subword-based downward closure is well understood, nothing is known in the case of priorities. In this talk, we discuss the priority order and provide algorithms to compute priority downward closures for regular languages, one-counter languages, and context-free languages.

This work has been accepted for publication at CONCUR'23.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 14–14.

RPTU

Regular Separability in Büchi VASS

Pascal Baumann[(B)] Roland Meyer[(A)] Georg Zetzsche[(B)]

[(B)]Max Planck Institute for Software Systems (MPI-SWS), Germany
{pbaumann,georg}@mpi-sws.org

[(A)]TU Braunschweig
roland.meyer@tu-bs.de

Abstract

We study the (ω-)regular separability problem for Büchi VASS languages: Given two Büchi VASS with languages L_1 and L_2, check whether there is a regular language that fully contains L_1 while remaining disjoint from L_2. We show that the problem is decidable in general and PSPACE-complete in the 1-dimensional case, assuming succinct counter updates. The results rely on several arguments. We characterize the set of all regular languages disjoint from L_2. Based on this, we derive a (sound and complete) notion of inseparability witnesses, non-regular subsets of L_1. Finally, we show how to symbolically represent inseparability witnesses and how to check their existence.

[(A)]The second author was supported by the DFG project EDS@SYN: Effective Denotational Semantics for Synthesis.

[(B)]Funded by the European Union (ERC, FINABIS, 101077902). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 15–15.

RPTU

Ramsey Quantifiers in Linear Arithmetics

Pascal Bergsträßer[(A)]

[(A)]University of Kaiserslautern-Landau, Kaiserslautern, Germany
bergstraesser@cs.uni-kl.de

We study Satisfiability Modulo Theories (SMT) enriched with the so-called Ramsey quantifiers, which assert the existence of cliques (complete graphs) in the graph induced by some formulas. The extended framework is known to have applications in proving program termination (in particular, whether a transitive binary predicate is well-founded), and monadic decomposability of SMT formulas. Our main result is a new algorithm for eliminating Ramsey quantifiers from three common SMT theories: Linear Integer Arithmetic (LIA), Linear Real Arithmetic (LRA), and Linear Integer Real Arithmetic (LIRA). In particular, if we work only with existentially quantified formulas, then our algorithm runs in polynomial-time and produces a formula of linear size. One immediate consequence is that checking well-foundedness of a given formula in the aforementioned theory defining a transitive predicate can be straightforwardly handled by highly optimized SMT-solvers. We show also how this provides a uniform semi-algorithm for verifying termination and liveness with completeness guarantee (in fact, with an optimal computational complexity) for several well-known classes of infinite-state systems, which include succinct timed systems, one-counter systems, and monotonic counter systems. Another immediate consequence is a solution to an open problem on checking monadic decomposability of a given relation in quantifier-free fragments of LRA and LIRA, which is an important problem in automated reasoning and constraint databases. Our result immediately implies decidability of this problem with an optimal complexity (coNP-complete) and enables exploitation of SMT-solvers. It also provides a termination guarantee for the generic monadic decomposition algorithm of Veanes et al. for LIA, LRA, and LIRA. We report encouraging experimental results on a prototype implementation of our algorithms on micro-benchmarks.

This is joint work with Moses Ganardi, Anthony W. Lin, and Georg Zetzsche.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 16–18.

RPTU

Synchronization and Diversity of Solutions

Emmanuel Arrighi[(A)] Henning Fernau[(B)]
Mateus de Oliveira Oliveira[(C)] Petra Wolf[(D)]

[(A)]Universities of Bergen, Trier, Lyon
emmanuel.arrighi@gmail.com

[(B)]University of Trier
fernau@uni-trier.de

[(C)]Universities of Bergen, Stockholm
Mateus.Oliveira@uib.no

[(D)]Universities of Trier, Bergen, Bordeaux
mail@wolfp.net

This is a very short account on our paper presented this year at AAAI, see [2]. This paper considers the notion of diversity of solutions in the context of synchronizing words.

1. Introduction

A word w is said to be *synchronizing* for a DFA A if there is some state q of A such that any state q' is sent to q by w. The most elementary problem is to determine whether a given DFA has a synchronizing word. This can be decided in polynomial time. Nevertheless, in several applications, one is interested in finding a synchronizing word satisfying certain additional constraints. Here, the complexity landscape changes drastically: even determining the existence of a synchronizing word satisfying additional length or regularity constraints is NP-hard, see [6, 7] for some examples.

As not all important features of a solution may be formalized completely, several applications then request the enumeration of all solutions, possibly additionally satisfying for instance some form of minimality. In the context of strings (as solutions), several possibilities exists to define minimality, based on different partial orderings, like prefix, infix, or subsequence orders. In relation with synchronizing words, these have been discussed in [8]. There, it was shown that, for instance concerning the subsequence ordering $|$, the question if, given a DFA A and a word $u \in \Sigma^*$, there exists a $|$-minimal synchronizing word w with $u|w$ is NP-hard. This rules out at least a simple way to obtain enumeration algorithms that achieve polynomial delay, which implies that a data analyst might have to wait exponentially long even between seeing two different solutions enumerated. This is clearly not acceptable.

Support by the Research Council of Norway, DAAD and DFG is gratefully acknowledged.

Alternative notions have been proposed to overcome this problem, one of them being *diversity*, suggested in [3]. The idea is to find a *small* set of solutions that are sufficiently diverse from one another. How can we adapt the framework of solution diversity to the context of synchronization? One problem is that usual notions of diversity of solutions based on Hamming distance are not appropriate to measure diversity between strings. For instance, distinct solutions may have distinct length. Even strings of the same length that are very similar to each other on an intuitive level may have very large Hamming distance, as $w = abab...ab$ and $w' = baba...ba$ show. Therefore, we base our diversity measure on the notion of *edit distance*. Unfortunately, a set of solutions S in which any two of them are far apart from each other may still not capture solution diversity in our context, as if w is a synchronizing word, then any xwy is synchronizing. We therefore require that each word in S is subsequence-minimal synchronizing.

2. Our Results

The subsequence minimality requirement combined with edit distance not only guarantees that solutions in any given subset are genuinely distinct, but also provides a way of tackling diverse synchronization problems using the machinery of finite automata theory. On the one hand, Higman's lemma [9] implies that the set of subsequence-minimal synchronizing words in the language of an automaton is always finite. On the other hand, the computation of the edit distance between two words is a process that can be simulated using finite automata. More specifically, it is possible to construct finite automata accepting a suitable encoding of pairs of words that are far apart from each other.

Note that subsequence-minimal synchronization problems involving a single DFA A are already very hard. First, subsequence-minimal synchronizing words for a DFA A may have exponential length on the number of states of A. Second, determining if a given word w is subsequence-minimal among all synchronizing words in the language of a DFA A is co-NP-hard. Third, determining if a DFA A has two distinct subsequence-minimal synchronizing words is NP-hard. Finally, the problem of counting the set of subsequence-minimal synchronizing words is #P-hard. We also remind the reader of the already mentioned NP-hardness result concerning the extension problem variant [8].

In order to cope with the inherent intractability of synchronization problems, we leverage on the framework of parameterized complexity theory [5]. In particular, we show that for each fixed value of r, interesting computational problems requiring a diverse set with r subsequence-minimal synchronizing words can be solved in time that is fixed parameter tractable with respect to the size of the synchronizing automaton A. Previously, algorithms with an FPT dependence in $|A|$ were unknown even for $r = 2$. Using our approach, we also show that given a DFA A with state set Q over an alphabet Σ, and a word $w \in \Sigma^*$, one can determine in time $O(f(|\Sigma|, |Q|) \cdot |w|)$, for some function f, if some subsequence-minimal synchronizing word for A is a subsequence of w, and we can construct such a subsequence in case the answer is affirmative. Recall that the unparameterized version of this problems is already co-NP-hard. Our main algorithmic result states that, given numbers $r, k \in \mathbb{N}$, a DFA A, and a possibly nondeterministic finite automaton B over an alphabet Σ, the problem of computing a subset $\{w_1, \ldots, w_r\} \subseteq L(B)$ of subsequence-minimal synchronizing words for A, with pairwise edit distance of at least k, can be solved in

time $O(f_A(r,k) \cdot |B|^r \log(|B|))$ for some suitable function f depending only on A, r and k. Intuitively, the automaton A is a specification of a system which we want to synchronize (or reset), and B is a specification of the set of words that are allowed to be used as synchronizing sequences. As stated above, the unparameterized version of this problem is NP-hard even if we are interested in finding a single solution and the language of the automaton B is as simple as ab^*a. As a consequence of our main result, given a word $w \in \Sigma^*$, the problem of determining whether there exist r subsequence-minimal synchronizing words for A that are subsequences of w and that are at least k apart from each other can be solved in time $O(f_A(r,k) \cdot |w|^r \log(|w|))$.

It turns out that our notion of diversity of solutions can be applied in other contexts where solutions are strings whose sizes may have vary. We adapt our framework to the realm of conformant planning, where the goal is to design plans that achieve goals irrespectively of initial conditions and of nondeterminism that may occur during the execution of these plans [1, 4, 10]. Throughout our paper, classical automata constructions help prove our results.

References

[1] A. S. ANDERS, *Reliably arranging objects: a conformant planning approach to robot manipulation*. Ph.D. thesis, Massachusetts Institute of Technology, USA, 2019.

[2] E. ARRIGHI, H. FERNAU, M. DE OLIVEIRA OLIVEIRA, P. WOLF, Synchronization and Diversity of Solutions. In: *The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)*. AAAI, 2023, 11516–11524.

[3] J. BASTE, M. R. FELLOWS, L. JAFFKE, T. MASARÍK, M. DE OLIVEIRA OLIVEIRA, G. PHILIP, F. A. ROSAMOND, Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence* **303** (2022), 103644.

[4] B. BONET, Conformant plans and beyond: Principles and complexity. *Artificial Intelligence* **174** (2010) 3-4, 245–269.

[5] R. G. DOWNEY, M. R. FELLOWS, *Parameterized Complexity*. Springer, 1999.

[6] D. EPPSTEIN, Reset sequences for monotonic automata. *SIAM Journal on Computing* **19** (1990) 3, 500–510.

[7] H. FERNAU, V. V. GUSEV, S. HOFFMANN, M. HOLZER, M. V. VOLKOV, P. WOLF, Computational Complexity of Synchronization under Regular Constraints. In: P. ROSSMANITH, P. HEGGERNES, J.-P. KATOEN (eds.), *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Leibniz International Proceedings in Informatics (LIPIcs) 138, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, 63:1–63:14.

[8] H. FERNAU, S. HOFFMANN, Extensions to minimal synchronizing words. *Journal of Automata, Languages and Combinatorics* **24** (2019), 287–307.

[9] G. HIGMAN, Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society (3)* **2** (1952) 7, 326–336.

[10] H. PALACIOS, H. GEFFNER, Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *Journal of Artificial Intelligence Research* **35** (2009), 623–675.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 19–24.

RPTU

Remarks on Parikh-recognizable omega-languages (Extended Abstract)

Mario Grobler[(A)] Leif Sabellek[(A)] Sebastian Siebertz[(A)]

[(A)]University of Bremen
{grobler,sabellek,siebertz}@uni-bremen.de

Abstract

Several variants of Parikh automata on infinite words were recently introduced by Guha et al. [FSTTCS, 2022]. We show that one of these variants coincides with blind counter machine as introduced by Fernau and Stiebe [Fundamenta Informaticae, 2008]. Fernau and Stiebe showed that every ω-language recognized by a blind counter machine is of the form $\bigcup_i U_i V_i^\omega$ for Parikh recognizable languages U_i, V_i, but blind counter machines fall short of characterizing this class of ω-languages. They posed as an open problem to find a suitable automata-based characterization. We introduce several additional variants of Parikh automata on infinite words that yield automata characterizations of classes of ω-language of the form $\bigcup_i U_i V_i^\omega$ for all combinations of languages U_i, V_i being regular or Parikh-recognizable. When both U_i and V_i are regular, this coincides with Büchi's classical theorem. We study the effect of ε-transitions in all variants of Parikh automata and show that almost all of them admit ε-elimination. Finally we study the classical decision problems with applications to model checking.

1. Introduction

Finite automata find numerous applications in formal language theory, logic, verification, and many more, in particular due to their good closure properties and algorithmic properties. To enrich this spectrum of applications even more, it has been a fruitful direction to add features to finite automata to capture also situations beyond the regular realm.

One such possible extension of finite automata with counting mechanisms has been introduced by Greibach in her study of blind and partially blind (one-way) multicounter machines [13]. Blind multicounter machines are generalized by weighted automata as introduced in [20]. Parikh automata (PA) were introduced by Klaedtke and Rueß in [19]. A PA is a non-deterministic finite automaton that is additionally equipped with a semi-linear set C, and every transition is equipped with a d-tuple of non-negative integers. Whenever an input word is read, d counters are initialized with the values 0 and every time a transition is used, the counters are incremented by the values in the tuple of the transition accordingly. An input word is accepted if the PA ends in an accepting

The full version of this paper can be found on arXiv [14]

state and additionally, the resulting d-tuple of counter values lies in C. We call such a pair an accepting configuration. Klaedtke and Rueß showed that PA are equivalent to weighted automata over the group $(\mathbb{Z}^k, +, \mathbf{0})$, and hence equivalent to Greibach's blind multicounter machines, as well as to reversal bounded multicounter machines [1, 17]. Recently it was shown that these models can be translated into each other using only logarithmic space [2]. In this work we call the class of languages recognized by any of these models *Parikh recognizable*. Klaedtke and Rueß [19] showed that the class of Parikh recognizable languages is precisely the class of languages definable in weak existential monadic second-order logic of one successor extended with linear cardinality constraints. On finite words, blind counter automata, Parikh automata and related models have been investigated extensively, extending [13, 19] for example by affine PA and PA on letters [4, 5], bounded PA [6], two-way PA [12], PA with a pushdown stack [18] as well as a combination of both [7], history-deterministic PA [8], automata and grammars with valences [9, 16], and several algorithmic applications, e.g. in the context of path logics for querying graphs [11].

Guha et al. [15] introduced *safety, reachability, Büchi- and co-Büchi Parikh automata*. These models provide natural generalization of studied automata models with Parikh conditions on infinite words. One shortcoming of safety, reachability and co-Büchi Parikh automata is that they do not generalize Büchi automata, that is, they cannot recognize all ω-regular languages. The non-emptiness problem, which is highly relevant for model checking applications, is undecidable for safety and co-Büchi Parikh automata. Furthermore, none of these models has ω-closure, meaning that for every model there is a Parikh-recognizable language (on finite words) L such that L^ω is not recognizable by any of these models. They raised the question whether (appropriate variants of) Parikh automata on infinite words have the same expressive power as blind counter automata on infinite words.

Büchi's famous theorem states that ω-regular languages are characterized as languages of the form $\bigcup_i U_i V_i^\omega$, where the U_i and V_i are regular languages [3]. As a consequence of the theorem, many properties of ω-regular languages are inherited from regular languages. For example, the non-emptiness problem for Büchi automata can basically be solved by testing non-emptiness for nondeterministic finite automata. In their systematic study of blind counter automata, Fernau and Stiebe [10] considered the class $\mathcal{K}_*$, the class of ω-languages of the form $\bigcup_i U_i V_i^\omega$ for Parikh-recognizable languages U_i and V_i. They proved that the class of ω-languages recognizable by blind counter machines is a proper subset of the class $\mathcal{K}_*$. They posed as an open problem to provide automata models that capture classes of ω-languages of the form $\bigcup_i U_i V_i^\omega$ where U_i and V_i are described by a certain mechanism.

2. Results

In this work, we propose *reachability-regular Parikh automata*, *limit Parikh automata*, and *reset Parikh automata* as new automata models.

We pick up the question of Fernau and Stiebe [10] to consider classes of ω-languages of the form $\bigcup_i U_i V_i^\omega$ where U_i and V_i are described by a certain mechanism. We define the four classes $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{Reg}}$, $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{Reg}}$, $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{PA}}$ and $\mathcal{L}^\omega_{\mathsf{PA},\mathsf{PA}}$ of ω-languages of the form $\bigcup_i U_i V_i^\omega$, where the U_i, V_i are regular or Parikh-recognizable languages of finite words, respectively. By Büchi's theorem the class $\mathcal{L}^\omega_{\mathsf{Reg},\mathsf{Reg}}$ is the class of ω-regular languages.

Guha et al. [15] showed that the class of Büchi PA-recognizable ω-languages is a strict subclass of $\mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{PA}}$. First we show the following characterization.

Theorem 2.1 *The following are equivalent for all ω-languages $L \subseteq \Sigma^{\omega}$:*

1. *L is Büchi PA-recognizable.*
2. *L is of the form $\bigcup_i U_i V_i^{\omega}$, where $U_i \in \Sigma^*$ is Parikh-recognizable and $V_i \in \Sigma^*$ is recognized by a PA where the initial state is the only accepting state and C is a linear set without base vector.*

We next show that the newly introduced reachability-regular Parikh automata, which are a small modification of reachability Parikh automata (as introduced by Guha et al. [15]) capture exactly the class $\mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{Reg}}$. Such an automaton accepts an infinite word if it has a prefix that leads to an accepting configuration, and an accepting state is seen infinitely often. This model turns out to be equivalent to limit Parikh automata. Such an automaton utilizes semi-linear sets over $\mathbb{N}^d \cup \{\infty\}$ and computes the Parikh image over the whole infinite word component-wise. This model was hinted at in the concluding remarks of [19].

Theorem 2.2 *The following are equivalent for all ω-languages $L \subseteq \Sigma^{\omega}$.*

1. *L is of the form $\bigcup_i U_i V_i^{\omega}$, where $U_i \in \Sigma^*$ is Parikh-recognizable, and $V_i \subseteq \Sigma^*$ is regular.*
2. *L is limit PA-recognizable.*
3. *L is reachability-regular.*

Fully resolving the classification of the above mentioned classes we introduce reset Parikh automata. Such an automaton resets the counters every time an accepting state is seen and the current counter values lie in the semi-linear set, and accepts an infinite word if it resets infinitely often. In contrast to all other Parikh models, these are closed under the ω-operation, while maintaining all algorithmic properties of PA (in particular, non-emptiness is NP-complete and hence decidable). We show that the class of Reset-recognizable ω-languages is a strict superclass of $\mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{PA}}$. We show that appropriate graph-theoretic restrictions of reset Parikh automata exactly capture the classes $\mathcal{L}^{\omega}_{\mathsf{PA},\mathsf{PA}}$ and $\mathcal{L}^{\omega}_{\mathsf{Reg},\mathsf{PA}}$, yielding the first automata characterizations for these.

Theorem 2.3 *The following are equivalent for all ω-languages $L \subseteq \Sigma^{\omega}$.*

1. *L is of the form $\bigcup_i U_i V_i^{\omega}$, where $U_i, V_i \subseteq \Sigma^*$ are Parikh-recognizable.*
2. *L is recognized by a strong reset PA $\mathcal{A}$ with the property that accepting states appear only in the leaves of the condensation of $\mathcal{A}$, and there is at most one accepting state per leaf.*

Theorem 2.4 *The following are equivalent for all ω-languages $L \subseteq \Sigma^{\omega}$.*

1. *L is of the form $\bigcup_i U_i V_i^{\omega}$, where $U_i \subseteq \Sigma^*$ is regular and $V_i \subseteq \Sigma^*$ is Parikh-recognizable.*
2. *L is recognized by a strong reset PA $\mathcal{A}$ with the following properties.*

(a) At most one state q per leaf of the condensation of $\mathcal{A}$ may have incoming transitions from outside the leaf, this state q is the only accepting state in the leaf, and there are no accepting states in non-leaves.

(b) only transitions connecting states in a leaf may be labeled with a non-zero vector.

The automata models introduced by Guha et al. [15] do not have ε-transitions, while blind counter machines have such transitions. Towards answering the question of Guha et al. we study the effect of ε-transitions in all Parikh automata models. We show that all models except safety and co-Büchi Parikh automata admit ε-elimination.

Theorem 2.5 *ε-reachability, ε-reachability-regular, ε-limit PA, Büchi PA and reset PA admit ε-elimination.*

This in particular answers the question of Guha et al. [15] whether blind counter machines and Büchi Parikh automata have the same expressive power over infinite words affirmative, as we can easily show that blind counter machines and ε-Büchi PA are equivalent.

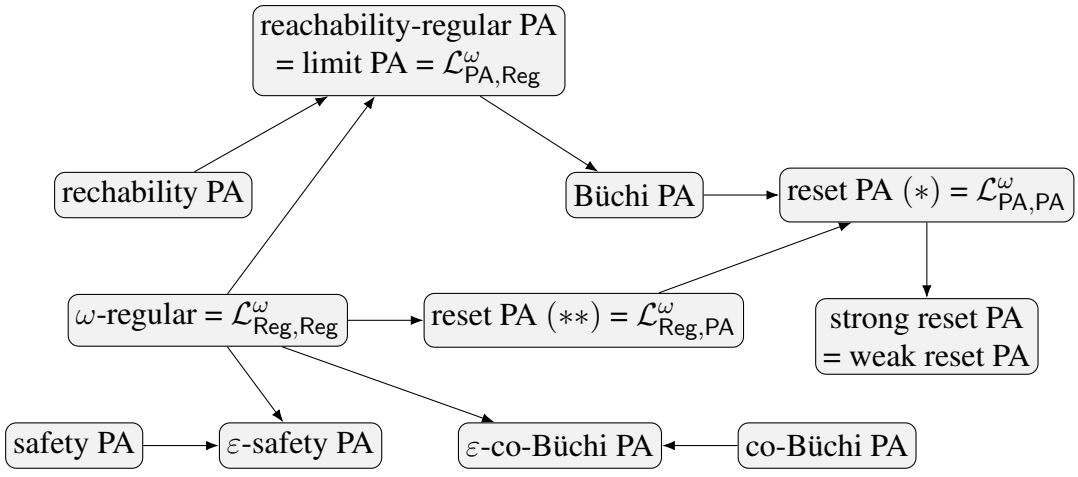
Lemma 2.6 *Blind counter machines and Büchi PA are equivalent.*

We show that safety and co-Büchi automata with ε-transitions are strictly more powerful than their variants without ε-transitions, and in particular, they give the models enough power to recognize all ω-regular languages.

Lemma 2.7 *Every ω-regular language is ε-safety PA and ε-co-Büchi PA recognizable.*

Corollary 2.8 *ε-safety PA and ε-co-Büchi PA do not admit ε-elimination.*

Find an overview of these results in Figure 1.



$(*)$ At most one state q per leaf of $C(\mathcal{A})$ may have incoming transitions from outside the leaf, this state q is the only accepting state in the leaf, and there are no accepting states in non-leaves;
$(**)$ and only transitions connecting states in leaves may be labeled with non-zero vectors.

Figure 1: Overview of our results. Arrows mean strict inclusions.

References

[1] B. S. BAKER, R. V. BOOK, Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences* **8** (1974) 3, 315–332.

[2] P. BAUMANN, F. D'ALESSANDRO, M. GANARDI, O. IBARRA, I. MCQUILLAN, L. SCHÜTZE, G. ZETZSCHE, Unboundedness Problems for Machines with Reversal-Bounded Counters. In: *Foundations of Software Science and Computation Structures*. Springer Nature Switzerland, Cham, 2023, 240–264.

[3] J. R. BÜCHI, Weak Second-Order Arithmetic and Finite Automata. *Mathematical Logic Quarterly* **6** (1960) 1-6, 66–92.

[4] M. CADILHAC, A. FINKEL, P. MCKENZIE, On the Expressiveness of Parikh Automata and Related Models. In: *Third Workshop on Non-Classical Models for Automata and Applications - NCMA 2011*. books@ocg.at 282, Austrian Computer Society, 2011, 103–119.

[5] M. CADILHAC, A. FINKEL, P. MCKENZIE, Affine Parikh automata. *RAIRO Theor. Informatics Appl.* **46** (2012) 4, 511–545.

[6] M. CADILHAC, A. FINKEL, P. MCKENZIE, Bounded Parikh Automata. *Int. J. Found. Comput. Sci.* **23** (2012) 8, 1691–1710.

[7] L. DARTOIS, E. FILIOT, J. TALBOT, Two-Way Parikh Automata with a Visibly Pushdown Stack. In: *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019*. Lecture Notes in Computer Science 11425, Springer, 2019, 189–206.

[8] E. ERLICH, S. GUHA, I. JECKER, K. LEHTINEN, M. ZIMMERMANN, History-deterministic Parikh Automata. *arXiv preprint arXiv:2209.07745* (2022).

[9] H. FERNAU, R. STIEBE, Sequential grammars and automata with valences. *Theoretical Computer Science* **276** (2002) 1, 377–405.

[10] H. FERNAU, R. STIEBE, Blind Counter Automata on omega-Words. *Fundam. Inform.* **83** (2008), 51–64.

[11] D. FIGUEIRA, L. LIBKIN, Path Logics for Querying Graphs: Combining Expressiveness and Efficiency. In: *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. LICS '15, IEEE, 2015, 329–340.

[12] E. FILIOT, S. GUHA, N. MAZZOCCHI, Two-Way Parikh Automata. In: *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019*. LIPIcs 150, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 40:1–40:14.

[13] S. A. GREIBACH, Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science* **7** (1978) 3, 311–324.

[14] M. GROBLER, L. SABELLEK, S. SIEBERTZ, Remarks on Parikh-recognizable omega-languages. *arXiv preprint arXiv:2307.07238* (2023).

[15] S. GUHA, I. JECKER, K. LEHTINEN, M. ZIMMERMANN, Parikh Automata over Infinite Words. In: *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*. Leibniz International Proceedings in Informatics (LIPIcs) 250, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 40:1–40:20.

[16] H. J. HOOGEBOOM, Context-Free Valence Grammars - Revisited. In: *Developments in Language Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, 293–303.

[17] O. H. IBARRA, Reversal-Bounded Multicounter Machines and Their Decision Problems. *J. ACM* **25** (1978) 1, 116—133.

[18] W. KARIANTO, Parikh automata with pushdown stack. *Diplomarbeit, RWTH Aachen* (2004).

[19] F. KLAEDTKE, H. RUESS, Monadic Second-Order Logics with Cardinalities. In: *Automata, Languages and Programming*. Springer, Berlin, Heidelberg, 2003, 681–696.

[20] V. MITRANA, R. STIEBE, Extended finite automata over groups. *Discrete Applied Mathematics* **108** (2001) 3, 287–300.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 25–28.

RPTU

Infinite Nyldon words

Pamela Fleischmann[(A)] Annika Huch[(A)] Dirk Nowotka[(A)]

[(A)]Department of Computer Science, Kiel University, Germany
{fpa,dn}@informatik.uni-kiel.de,stu216885@mail.uni-kiel.de

Abstract

In this work we investigate infinite Nyldon words. Those are defined by reversing the lexicographical order in the infinite Lyndon factorisation by Siromoney et al. They showed that each infinite word can be uniquely, lexicographically non-increasingly factorised into either a finite sequence of (finite) Lyndon words followed by one infinite Lyndon word or an infinite sequence of (finite) Lyndon words. Here, we can observe several similarities to the behaviour of finite Nyldon words (in detail examined by Charlier et al.). We show that each infinite word, has a unique infinite Nyldon factorisation. Further, we state structural results on infinite Nyldon words as a characterisation of their suffixes and a standard factorisation.

1. Introduction

A *Lyndon word* is defined as a non-repetitive word which is the smallest amongst its cyclic rotations. This class of words is strongly studied in, e.g., [2, 3, 7, 5]. One major result by [2], today known as the Chen-Fox-Lyndon theorem shows that Lyndon words factorise the free monoid, i.e., for each word in the free monoid there exists a unique, lexicographically non-increasing factorisation into Lyndon words. In a Mathoverflow post from November 2014 Grinberg raised the question how the factorisation of the free monoid changes when investigating the above factorisation from Chen, Fox and Lyndon w.r.t. a reversed lexicographical order, i.e., a lexicographically non-decreasing factorisation where each factor is smaller or equal than its successor. Hence, the notion of *Nyldon words* developed as the set of those words which are either letters or cannot be factorised into at least two, lexicographically non decreasing Nyldon factors. For example, the word 10 is Nyldon since 1 and 0 are Nyldon words and thus, the only factorisation of 10 is $1 \cdot 0$ which is lexicographically decreasing. Further, 110 is not Nyldon since $1 \cdot 10$ is its non-decreasing factorisation into Nyldon words. In [1] they investigated Nyldon words and showed that they also form a unique factorisation of the free monoid. Further, they give a standard factorisation of Nyldon words (similar to the well-known standard factorisation for Lyndon words). It turns out that Nyldon words are harder to grasp because they miss the property of being the smallest word under its cyclic rotations (neither they are the largest). Thus, the properties of Lyndon words cannot be immediately transferred to Nyldon words which strengthens the relevance of their investigation. An example are the *infinite Nyldon words*, i.e., infinite words that cannot be lexicographically non-decreasingly factorised into (1) a finite sequence

of (finite) Nyldon words followed by an infinite Nyldon word, or (2) an infinite sequence of (finite) Nylon words (cf. [6] for results on infinite Lyndon words). We present several properties to deeper understand and characterise these infinite words. First, we show that this infinite Nyldon factorisation is unique for each infinite Nyldon word. Further, each infinite Nyldon word is lexicographically larger than its infinite Nydon suffixes. Using these results we present a standard factorisation for infinite Nyldon words.

2. Preliminaries

Let $\mathbb{N} = \{1,2,\ldots\}$, define $[m] = \{1,\ldots,m\}$. For the standard definitions of combinatorics on words, especially for the whole background of Lyndon words, we refer to [4, Chapter 5]. Denote by $\Sigma^\omega := \{\mathtt{a}_1\mathtt{a}_2\cdots \mid \mathtt{a}_i \in \Sigma, 1 \leq i\}$ the set of all right infinite concatenations over Σ. Further, a tuple $f = (w_1,\ldots,w_k) \in (\Sigma^*)^k$ is called a *factorisation* of the (finite) word $w \in \Sigma^*$ if $w = w_1\cdots w_k$. For a *factorisation* of an infinite word $w \in \Sigma^\omega$ there are two options: (1) a factorisation $g = (w_1,w_2,\ldots)$ with $w_i \in \Sigma^*$ for $i \in \mathbb{N}$ such that $w = w_1w_2\cdots$, or (2) a factorisation $h = (w_1,\ldots,w_{n-1},w_n)$ with $w_i \in \Sigma^*$ for $i \in [n-1]$ and $w_n \in \Sigma^\omega$ such that $w = w_1\cdots w_{n-1}w_n$. Let $\lhd$ be a total order on Σ. We extend this order to Σ^* by $u \lhd v$ for $u,v \in \Sigma^*$ iff u is a prefix of v or $u = x\mathtt{a}u'$ and $v = x\mathtt{b}v'$ with $\mathtt{a} \lhd \mathtt{b}$ for $\mathtt{a},\mathtt{b} \in \Sigma$ and some $u',v',x \in \Sigma^*$. This extended order is called *lexicographical order* on words over Σ and forms a total order on Σ^*. This lexicographical order can be further extended to Σ^ω. Note that the first condition, u is a prefix of v, is only applicable if $u \in \Sigma^*$ is finite. For the second condition both, u and v may belong to Σ^* or Σ^ω. A word $w \in \Sigma^*$ is a Lyndon word iff it is primitive and the lexicographically smallest in its conjugacy class. We denote the set of Lyndon words by $\mathcal{L}$. It is well known that Lyndon words factorise the free monoid: each $w \in \Sigma^*$ has a unique *Lyndon factorisation* $(\ell_1,\ldots,\ell_k)$ with $\ell_j \in \mathcal{L}$ for $j \in [k]$ and $\ell_i \trianglerighteq \ell_{i+1}$ for $i \in [k-1]$ (Chen-Fox-Lyndon Theorem). Thus, Lyndon words can be defined as those words that do not have any non-decreasing factorisation into at least two Lyndon words, i.e., they cannot be further factorised into a Lyndon factorisation with at least two Lyndon factors. In [1], the authors introduce Nyldon words. A word $w \in \Sigma^+$ is called *Nyldon* ($w \in \mathcal{N}$) if $w \in \Sigma$ or there does not exist any factorisation $(n_1,\ldots n_k)$ of w with $k \geq 2$, $n_j \in \mathcal{N}$ for $j \in [k]$ and $n_i \trianglelefteq n_{i+1}$ for $i \in [k-1]$. We now continue by defining infinite Lyndon words [6]. An infinite word $w \in \Sigma^\omega$ is *ω-Lyndon* if it has an infinite number of Lyndon prefixes. Denote the set of all infinite Lyndon words by $\mathcal{L}_\omega$. Similar to finite Lyndon words, each infinite word has a unique infinite Lyndon factorisation [6], i.e., any word $w \in \Sigma^\omega$ has a unique factorisation of the form: (1) $(\ell_1\ell_2,\ldots,\ell_k)$ where $\ell_i \in \mathcal{L}$ for $i \in [k-1]$ and $\ell_k \in \mathcal{L}_\omega$ with $\ell_i \trianglerighteq \ell_{i+1}$, or (2) $(\ell_1,\ell_2,\ldots)$ where $\ell_i \in \mathcal{L}$ and $\ell_i \trianglerighteq \ell_{i+1}$ for $i \in \mathbb{N}$. We define infinite Nyldon words similar to the finite case.

Definition 2.1 *An infinite word $w \in \Sigma^\omega$ is* ω-Nyldon *if it cannot be factorised into one of the following factorisations:*
1. $(n_1,n_2,\ldots,n_k)$ where $n_i \in \mathcal{N}$ for $i \in [k-1]$, $n_k \in \mathcal{N}_\omega$ with $n_i \trianglelefteq n_{i+1}$ and $k \geq 2$, or
2. $(n_1,n_2,\ldots)$ where $n_i \in \mathcal{N}$ and $n_i \trianglelefteq n_{i+1}$ for $i \in \mathbb{N}$.

The set of ω-Nyldon words will be denoted by $\mathcal{N}_\omega$. As an extension to the factorisations of finite words, any factorisation $(n_1,\ldots,n_k)$ or $(n'_1,n'_2,\ldots)$ of $w \in \mathcal{N}_\omega$ into Nyldon words such that $n_1 \trianglelefteq \cdots \trianglelefteq n_k$, or $n'_1 \trianglelefteq n'_2 \trianglelefteq \ldots$, respectively, is called a Nyldon factorisation *of w.*

An infinite word w is ω-Nyldon iff its infinite Nyldon factorisation is of length 1. Further, an infinite word is not ω-Nyldon, iff its infinite Nyldon factorisation is of length at least 2.

Example 2.2 *To get an intuition, we give examples for Nyldon and non-Nyldon words together with a justification regarding the Nyldon factorisation. First, consider* $w = \mathtt{10}^\omega \in \mathcal{N}_\omega$. *It has no factorisation of the first form of Definition 2.1 since* $\mathtt{0}^\omega \notin \mathcal{N}_\omega$ *(*$\mathtt{0}$ *is the only finite Nyldon word starting with zero, i.e., the only possible infinite Nyldon factorisation of* $\mathtt{0}^\omega$ *is* $(\mathtt{0},\mathtt{0},\ldots)$*). Further there exists no factorisation of the second form because* $\mathtt{1} \not\trianglerighteq \mathtt{0}$ *and* $\mathtt{10}^n \not\trianglerighteq \mathtt{0}$. *Thus, each possible factorisation is not increasing. As another example, one can verify that* $\mathtt{101}^\omega \in \mathcal{N}_\omega$.

Moreover, $\mathtt{010}^\omega \notin \mathcal{N}_\omega$ *has a Nyldon factorisation of the first form of Definition 2.1 since* $\mathtt{0} \trianglelefteq \mathtt{10}^\omega$. *Further, consider* $(\mathtt{10})^\omega \notin \mathcal{N}_\omega$. *We know that* $\mathtt{10} \in \mathcal{N}$ *so* $\mathtt{10}\cdots\mathtt{10}$ *is an infinite Nyldon factorisation of the second form of Definition 2.1.*

3. Infinite Nyldon words

In 1994, Siromoney et al. [6] have shown that several known results for Lyndon words also apply to infinite Lyndon words, e.g., the standard factorisation, and a unique infinite Lyndon factorisation. The aim of this chapter is to investigate infinite Nyldon words.

First, we want to show the uniqueness of the Nyldon factorisation for infinite words. With a similar proof this result was shown for infinite Lyndon words in [6].

Theorem 3.1 *Any infinite word* $w \in \Sigma^\omega$ *has a unique factorisation of the form*
1. $(n_1, n_2, \ldots, n_k)$ *where* $n_i \in \mathcal{N}$ *for* $i \in [k-1]$, $n_k \in \mathcal{N}_\omega$ *and* $n_i \trianglelefteq n_{i+1}$, *or*
2. $(n_1, n_2, \ldots)$ *where* $n_i \in \mathcal{N}$ *and* $n_i \trianglelefteq n_{i+1}$, $i \in \mathbb{N}$.

In [1] is is shown that the last factor of a words Nyldon factorisation is always the longest proper Nyldon suffix of this word. Since the notion of longest Nyldon suffixes is not applicable in the case of infinite words we adapt this result. The last factor of the Nyldon factorisation might be an infinite word. So we will show that there exists no shorter prefix such that this last factor is an infinite Nyldon word. The proof works similar to the finite case.

Proposition 3.2 *Let* $w \in \Sigma^\omega$ *such that its Nyldon factorisation* $(n_1, \ldots, n_k)$ *is finite with* $n_i \in \mathcal{N}$ *for* $i \in [k-1]$, $n_k \in \mathcal{N}_\omega$ *and* $n_i \trianglelefteq n_{i+1}$ *(Case 1 of Definition 2.1). Then* $n_1 \cdots n_{k-1}$ *is the shortest prefix of* w *such that* $n_k \in \mathcal{N}_\omega$.

Remark 3.3 *Note that the first factor of the infinite Nyldon factorisation is in both cases of Theorem 3.1 not necessarily the longest Nyldon prefix. For example, let* $w = \mathtt{10100}^\omega \in \Sigma^\omega$. *Its infinite Nyldon factorisation is* $(\mathtt{10}, \mathtt{100}^\omega)$ *but* $\mathtt{101}$ *is its longest Nyldon prefix. Second, consider* $w' = (\mathtt{10})^\omega$ *that decomposes into* $(\mathtt{10}, \mathtt{10}, \ldots)$ *although* $\mathtt{101}$ *is its longest Nyldon prefix.*

One useful property of infinite Nyldon words would be if, similar to finite Nyldon words, all their infinite Nyldon suffixes are smaller than the words themselves. To prove this, we need the following result. Note that the this result is not applicable to finite Nyldon words.

Lemma 3.4 *Let* $w = ps \in \mathcal{N}_\omega$. *If* $s \in \mathcal{N}_\omega$ *then* $p \in \mathcal{N}$.

Example 3.5 *Consider* $w = \mathtt{101100}^\omega \in \mathcal{N}_\omega$. *Now,* $\mathtt{100}^\omega \in \mathcal{N}_\omega$ *and* $\mathtt{101} \in \mathcal{N}$ *(Proposition 3.4).*

Theorem 3.6 *Let $w \in \mathcal{N}_\omega$. Then for all infinite Nyldon suffixes $s \in \mathcal{N}_\omega$ of w, $s \lhd w$ holds.*

Remark 3.7 *Note, that a standard factorisation for infinite Nyldon words cannot work similar to the standard factorisation of finite Nyldon words. The problem is that the standard factorisation of finite Nyldon words relies on Nyldon suffixes. For example, the infinite Nyldon words $\mathtt{10}^\omega$ and $\mathtt{101}^\omega$ both do not have any infinite Nyldon suffix (neither $\mathtt{0}^\omega$, $\mathtt{1}^\omega$, nor $\mathtt{01}^\omega$ are Nyldon).*

Lemma 3.8 *Let $w \in \Sigma^*$ with an infinite Nyldon factorisation $(n_1, n_2, \ldots)$. Then there exists no proper infinite Nyldon suffix $s <_s w$.*

This allows us to introduce an adapted version of the standard factorisation to the finite case.

Theorem 3.9 *Let $p \in \mathcal{N}$ and $s \in \mathcal{N}_\omega$. We have $w \in \mathcal{N}_\omega$ iff one of the following holds:*
1. There exists no proper infinite Nyldon suffix s of w, or
2. if p is the shortest proper Nyldon prefix of ps such that $s \in \mathcal{N}_\omega$ then $p \rhd s$.

4. Conclusion

In this work we introduced infinite Nyldon words and started their investigation. We can observe that infinite Nyldon words resemble the finite Nyldon words in many aspects, i.e., the uniqueness of the infinite Nyldon factorisation and the fact that each Nyldon word is smaller than all its Nyldon suffixes. When investigating a standard factorisation, we need to add the adaption that an infinite Nyldon word might not have any infinite Nyldon suffix. For further research it would be interesting to determine the infinite Nyldon factorisations of several famous infinite words like the Thue-Morse word, the Fibonacci word and Sturmian words in general.

References

[1] É. CHARLIER, M. PHILIBERT, M. STIPULANTI, Nyldon words. *J. Comb. Theory, Ser. A* **167** (2019), 60–90.

[2] K. T. CHEN, R. H. FOX, R. C. LYNDON, Free differential calculus, IV. The quotient groups of the lower central series. *Annals of Mathematics* (1958), 81–95.

[3] J. DUVAL, Mots de Lyndon et Périodicité. *RAIRO Theor. Informatics Appl.* **14** (1980) 2, 181–191.

[4] M. LOTHAIRE, *Combinatorics on Words*. Cambridge Mathematical Library, Cambridge University Press, 1997.

[5] R. SIROMONEY, L. MATHEW, A Public Key Cryptosystem Based on Lyndon Words. *Inf. Process. Lett.* **35** (1990) 1, 33–36.

[6] R. SIROMONEY, L. MATHEW, V. R. DARE, K. G. SUBRAMANIAN, Infinite Lyndon Words. *Inf. Process. Lett.* **50** (1994) 2, 101–104.

[7] K. G. SUBRAMANIAN, R. SIROMONEY, L. MATHEW, Lyndon Trees. *Theor. Comput. Sci.* **106** (1992) 2, 373–383.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 29–32.

RPTU

Separability and Non-Determinizability of WSTS

Eren Keskin[(A)] Roland Meyer[(B)]

[(A)]TU Braunschweig
e.keskin@tu-bs.de
[(B)]TU Braunschweig
r.meyer@tu-bs.de

Well-structured transition systems (WSTS) are among the most liberal transition systems that still admit decidability results. These are (typically infinite state) labeled transition systems (LTS), whose states are endowed with a well quasi order (WQO). The transitions of the WSTS must be compatible with this order, and if a state is final, then so must all the states that dominate it. Many popular models of computation, vector addition systems, lossy channel systems, and concurrent programs operating under weak memory models fall under the WSTS umbrella [4].

A recent separability result [3] for the languages of WSTS makes a surprisingly general statement: For two disjoint languages, respectively accepted by a deterministic and an unrestricted WSTS, there is a regular language that includes one language and completely excludes the other. The principal proof technique developed in [3] works for any order. However, the argument for ensuring a separator with finitely many states uses ideal decompositions of WQO's. Because the WSTS property is lost upon naive determinization, the determinicity assumption is hard to decouple from the argument. In this light, the separability result can be generalized to languages of all WSTS in one of two ways, none of which has lead to conclusions so far: (i) show that all WSTS languages can be accepted by deterministic WSTS, (ii) develop a new technique that is not based on ideal decompositions. Our first contribution is to develop a technique in line with (ii). Here, we employ a more subtle concept of limits, instead of ideal decompositions. Our second contribution is to show that (i) is not possible by giving a witness WSTS language that cannot be accepted by a deterministic WSTS.

1. Regular Separability

To show regular separability for all WSTS, we employ the proof principle developed in [3], which was also used to show the main result in [3]. Note that the proof principle refers to ULTS instead of WSTS. ULTS are LTS endowed with any (not necessarily WQO) order with which they are compatible. They form a superset of WSTS, because WSTS also require the endowed order to be a WQO.

Theorem 1.1 (Proof Principle for Regular Separability, [3]) *Given any two ULTS U and V, one deterministic with $L(U) \cap L(V) = \emptyset$, if there is a finitely represented inductive invariant S in $U \times V$, then the languages $L(U)$ and $L(V)$ are regularly separable.*

Inductive invariants are key to Theorem 1.1. An inductive invariant is a set of states that is (i) disjoint from the final states, (ii) contains the initial states, (iii) cannot be escaped by taking transitions. It is guaranteed to exist as soon as the language of the LTS is empty, which is the case for $L(U \times V) = L(U) \cap L(V)$. The challenging step in applying Theorem 1.1, is finding a *finitely represented* inductive invariant. Finite representation of S refers to the existence of a finite set $X \subseteq Q_\times$ with $S = {\downarrow}X \coloneqq \{q \in Q_\times \mid q \leq p \in X\}$, where $(Q_\times, \leq)$ refers to the (ordered) states of the product ULTS. For Theorem 1.1 to apply, this challenge must be overcome in the setting of deterministic systems. Any ULTS can be determinized by moving on to the downward closed subsets of the original state space, ordered by inclusion. However, this determinization is not guaranteed to preserve the WQO property. Even though this is the case, a seldom used, weaker property must still hold. First observed by Rado [5], this property states that for any sequence of downward closed sets of states $[X_i]_{i\in\mathbb{N}}$, there is a convergent subsequence $[X_{\varphi(i)}]_{i\in\mathbb{N}}$ in the following sense. Any element p that appears in any set $X_{\varphi(i)}$, also appears in all but finitely many of the other sets $X_{\varphi(j)}$. A lattice-theoretic description of this property allows us to abstract away from the membership relation.

Definition 1.2 *A* converging lattice $(Q, \leq)$ *is a completely distributive lattice, where every sequence* $[p_i]_{i\in\mathbb{N}}$ *has a converging subsequence* $[p_{\varphi(i)}]_{i\in\mathbb{N}}$*. A* converging sequence $[q_i]_{i\in\mathbb{N}}$ *is an infinite sequence with*

$$\bigsqcup_{i\in\mathbb{N}} \bigsqcap_{j\geq i} q_j = \bigsqcup_{i\in\mathbb{N}} q_i \ .$$

Our approach is to initially determinize both WSTS and to find a finitely represented inductive invariant in the product by relying on convergence. We show that converging sequences $[q_i]_{i\in\mathbb{N}}$ and their limits are stable under transitions. Furthermore, we argue that if the limit is in the final states, then so must be an element from the sequence. Then, including the limits of all the converging sequences in a given inductive invariant S also results in an inductive invariant, $cl(S)$. This is the precise process that gives us the finite representation. We show that $cl(S)$ is chain complete (under the assumption of a countable state space), because all increasing sequences of subsets are convergent wrt. Definition 1.2. In this case, we can apply Zorn's Lemma to get maximal elements which represent the inductive invariant. Finally, we observe that there can only be finitely many maximal elements. Supposing there were infinitely many maximal elements, we see that a converging sequence could be extracted from these elements. This leads to comparability among maximal elements, which is a contradiction.

2. Non-Determinizability of WSTS

One way of getting rid of the determinicity assumption in [3] would be to show that all WSTS can be determinized. We show that this is not possible by constructing a WSTS language T that no deterministic WSTS accepts. To prove this, we employ a novel characterization of deterministic WSTS languages. This relies on a classical concept in formal languages, the Nerode quasi order. For a language $L \subseteq \Sigma^*$, the Nerode quasi order $w \leq_L v$ holds for $w, v \in \Sigma^*$, if $w.u \in L$ implies $v.u \in L$ for all $u \in \Sigma^*$. By a similar approach to the Myhill-Nerode Theorem, the characterization says that deterministic WSTS languages are precisely the languages whose

Nerode quasi order is a WQO. This is in contrast to the folklore result [1, Proposition 5.1] that says a language is regular if and only if the syntactic quasi order is a WQO.

Lemma 2.1 (Characterization of L(detWSTS)) *$L \in L(detWSTS)$ iff $\leq_L$ is a WQO.*

The state space of the WSTS that accepts our witness language T is the Rado structure $(R, \leq_R)$. This is a structure that is particularly suited for this task. Any WQO that loses the WQO property upon powerset construction embeds this WQO [2]. Using the Rado structure as our state space, we construct a (non-deterministic) WSTS that accepts the language $T \subseteq \{a, \bar{a}, \mathit{zero}\}^*$ with the property

$$T \cap a^*.\bar{a}^*.\mathit{zero}^* = \{a^n.\bar{a}^n.\mathit{zero}^i \mid i \in \mathbb{N}\} \cup \{a^n.\bar{a}^k.\mathit{zero}^i \mid i \in \mathbb{N}, n-k > i\}.$$

We can already deduce that $\leq_T$ is not a WQO from this description. The order contains the infinite antichain $[a^i]_{i\in\mathbb{N}}$. Assume $n > k$. For $a^n \not\leq_T a^k$, we have $a^n.\bar{a}^n \in T$ while $a^k.\bar{a}^n \notin T$. Conversely, for $a^k \not\leq_T a^n$, we have $a^n.\bar{a}^k.\mathit{zero}^{n-k} \notin T$ while $a^k.\bar{a}^k.\mathit{zero}^{n-k} \in T$.

3. Further results

If the states of an ULTS are ordered by a reversed WQO, the ULTS is called a downward-WSTS (DWSTS). By slightly modifying our proofs from the previous sections, we also deduce results for the class of languages accepted by DWSTS. First we note that the languages of DWSTS are those of WSTS with the words reversed. Combining this with the closure of regular languages under reversal yields the regular separability of disjoint DWSTS languages. We also observe that reversing the transitions in the WSTS that accepts T results in a deterministic DWSTS. This insight gives us the remaining relations between the language classes, summarized in Figure 1.

We shortly clarify the relations depicted in Figure 1. Reversing the languages of deterministic WSTS might not result in deterministic DWSTS languages, and vice-versa. For both classes WSTS and DWSTS, non-determinism results in a strictly more expressive class of langauges. Finally, the languages of deterministic WSTS are exactly the complements of the deterministic DWSTS languages, and the languages of WSTS are exactly the revesals of the languages of DWSTS.

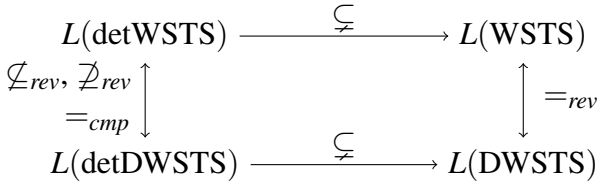


Figure 1: Relations between language classes.

References

[1] G.Rozenberg A.Salomaa, editor. *Handbook of Formal Languages*. Springer, 1997.

[2] P. Jančar. A note on well quasi-orderings for powersets. *IPL*, 72(5):155–160, 1999.

[3] W. Czerwiński, S. Lasota, R. Meyer, S. Muskalla, K. Narayan Kumar, and P. Saivasan. Regular separability of well-structured transition systems. In *CONCUR*, volume 118 of *LIPIcs*, pages 35:1–35:18. Dagstuhl, 2018.

[4] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *TCS*, 256(1-2):63–92, 2001.

[5] R. Rado. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95, 1954.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 33–36.

RPTU

Regular Separators for VASS Coverability Languages

Chris Köcher[(A)] Georg Zetzsche[(A)]

[(A)]Max Planck Insitute for Software Systems, Kaiserslautern
{ckoecher,georg}@mpi-sws.org

1. Introduction

Safety verification of concurrent systems typically consists of deciding whether two languages $K, L \subseteq \Sigma^*$ are disjoint: If each of the languages describes the set of event sequences that (i) are consistent with the behavior of a some system component and (ii) reach an undesirable state, then their intersection is exactly the set of event sequences that are consistent with both components and reach the undesirable state.

If we wish to not only decide, but *certify* disjointness of languages $K, L \subseteq \Sigma^*$, then a natural kind of certificate is a *regular separator*: a regular language $R \subseteq \Sigma^*$ such that $K \subseteq R$ and $L \cap R = \emptyset$. Regular separators can indeed act as disjointness certificates: Deciding whether a given language intersects (resp. is included in) a regular language is usually simple.

The *regular separability* problem asks whether for two given languages there exists a regular separator. This decision problem has recently attracted a significant amount of interest. After the problem was shown to be undecidable for context-free languages in the 1970s [8, 6], recent work had a strong focus on *vector addition systems* (VASS), which are automata with counters that can be incremented, decremented, but not tested for zero. Typically, VASS are considered with two possible semantics: With the *reachability semantics*, where a target configuration has to be reached exactly, and the *coverability semantics*, where the target only has to be covered. Decidability of regular separability remains an open problem for reachability semantics. However, decidability has been established for coverability languages of VASS [4] and several other subclasses, such as one-dimensional VASS [3], integer VASS [1] (where counters can become negative), and commutative VASS languages [2]. Moreover, for each of these subclasses, decidability is retained if one of the input languages is an arbitrary VASS reachability language [5].

The decidability result about VASS coverability languages is a consequence of a remarkable and surprising result by Czerwiński, Lasota, Meyer, Muskalla, Kumar, and Saivasan [4]: Two languages of finitely-branching well-structured transition systems (WSTS) are separable by a regular language if and only if they are disjoint. (In fact, very recently, Keskin and Meyer [7] have even shown that the finite branching assumption can be lifted.) Moreover, VASS (with coverability semantics) are a standard example of (finitely branching) WSTS.

Funded by the European Union (ERC, FINABIS, 101077902).

Despite this range of work on decidability, very little is known about a fundamental aspect of the separators: *What is the size of the separator, if they exist?* Here, by size, we mean the number of states in an NFA or DFA. In fact, the only result we are aware of is a partial answer for VASS coverability languages: In [4] a triply exponential upper bound and a doubly exponential lower bound is shown for NFA separating VASS coverability languages, leaving open whether there always exists a doubly-exponential separator.

Contribution. We study the size of regular separators in VASS coverability languages. Our first main result is that if two VASS coverability languages are disjoint, then there exists a doubly exponential-sized separating NFA. We then provide a comprehensive account of separator sizes for VASS languages: We study separator sizes in (i) fixed/arbitrary dimension, (ii) with unary/binary counter updates and (iii) deterministic/non-deterministic separators. In each case, we provide a tight polynomial or singly, doubly, or triply exponential bound.

2. Vector Addition Systems

Let $d \in \mathbb{N}^+$. A *(d-dimensional) vector addition system with states* or *(d-)VASS* is a tuple $\mathfrak{V} = (Q, \Sigma, \Delta, s, t)$ where Q is a finite set of *states*, Σ is an alphabet, $\Delta \subseteq Q \times \Sigma_\varepsilon \times \mathbb{Z}^d \times Q$ is a finite set of transitions, and $s, t \in Q$ are its *source* resp. *target states*. Here, Σ_ε denotes the set $\Sigma \cup \{\varepsilon\}$.

A *configuration* is a tuple from $Q \times \mathbb{N}^d$. For two configurations $(p, \vec{u}), (q, \vec{v}) \in Q \times \mathbb{N}^d$ and $w \in \Sigma^*$ we write $(p, \vec{u}) \xrightarrow{w}_{\mathfrak{V}} (q, \vec{v})$ if there is $\ell \in \mathbb{N}$, configurations $(q_i, \vec{v_i}) \in Q \times \mathbb{N}^d$ for each $0 \leq i \leq \ell$ and transitions $(q_{i-1}, a_i, \vec{x_i}, q_i) \in \Delta$ with $\vec{v_i} = \vec{v_{i-1}} + \vec{x_i}$ for each $1 \leq i \leq \ell$ such that $w = a_1 a_2 \dots a_\ell$ holds. Here, $+$ is the component-wise addition of integers in d-dimensional vectors.

The *(coverability) language* of $\mathfrak{V}$ is $\mathrm{L}(\mathfrak{V}) = \{w \in \Sigma^* \mid \exists \vec{v} \in \mathbb{N}^d \colon (s, \vec{0}) \xrightarrow{w}_{\mathfrak{V}} (t, \vec{v})\}$. Note that $\vec{v} \geq \vec{0}$ holds for any $\vec{v} \in \mathbb{N}^d$; we say that $(t, \vec{v})$ *covers* the target configuration $(t, \vec{0})$. We call $L \subseteq \Sigma^*$ a *(coverability) d-VASS-language* if there is a d-VASS $\mathfrak{V}$ with $L = \mathrm{L}(\mathfrak{V})$.

The following equivalence is known about regular separability of coverability VASS-languages:

Theorem 2.1 ([4]) *Let $\mathfrak{V}$ and $\mathfrak{W}$ be two VASS. The languages $\mathrm{L}(\mathfrak{V})$ and $\mathrm{L}(\mathfrak{W})$ are regular separable if, and only if, $\mathrm{L}(\mathfrak{V}) \cap \mathrm{L}(\mathfrak{W}) = \emptyset$ holds.*

3. Main Results

In this section, we present the main results of this work. An overview can be found in Table 1. Here, by i-exp, we mean that there is an i-fold exponential upper bound. All our bounds are tight in the sense that for each i-exp upper bound, there is also an i-fold exponential lower bound.

First upper bound. Our first upper bound result is the following.

		NFAs		DFAs	
		unary	binary	unary	binary
d as input		2-exp.	2-exp.	3-exp.	3-exp.
d fixed	$d \geq 2$	poly.	exp.	exp.	2-exp.
	$d = 1$	poly.	exp.	exp.	exp.

Table 1: An overview over the upper and lower bounds for finite automata separating two disjoint d-VASS. We distinguish between (i) whether the dimension $d \in \mathbb{N}^+$ is part of the input, (ii) whether the separating automaton should be an NFA or a DFA, and (iii) whether counter updates are encoded in unary or binary. The colors denote the employed lower bound technique.

Theorem 3.1 *Let $\mathfrak{V}_1$ and $\mathfrak{V}_2$ be d-VASS with at most $n \geq 1$ states and updates of norm at most $m \geq 1$. If $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$, then $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$ are separated by an NFA with at most $(n+m)^{2^{\mathrm{poly}(d)}}$ states.*

This provides almost all upper bounds in Table 1. In particular, it closes the gap left by [4] by providing a doubly exponential upper bound for NFA separators in the general case.

Let us explain how we avoid one exponential blow-up compared to [4]. In [4], the authors first construct VASS $\mathfrak{V}_1'$ and $\mathfrak{V}_2'$ such that (i) $\mathfrak{V}_2'$ is deterministic, (ii) $\mathrm{L}(\mathfrak{V}_1') \cap \mathrm{L}(\mathfrak{V}_2') = \emptyset$ and (iii) any separator for $\mathrm{L}(\mathfrak{V}_1')$ and $\mathrm{L}(\mathfrak{V}_2')$ can be transformed into a separator for $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$. Then, relying on Rackoff-style bounds for covering runs in VASS, they construct a doubly exponential NFA separator for $\mathrm{L}(\mathfrak{V}_1')$ and $\mathrm{L}(\mathfrak{V}_2')$. The latter step yields an inherently non-deterministic separator. However, the transformation mentioned in (iii) requires a complementation, which results in a triply exponential bound overall.

Instead, roughly speaking, we first apply an observation from [5] to reduce to an even more specific case: Namely, we construct $\mathfrak{V}$ such that for the language C_d of all counter instruction sequences that keep d counter above zero, we have (a) $\mathrm{L}(\mathfrak{V}) \cap C_d = \emptyset$ and (b) any separator of $\mathrm{L}(\mathfrak{V})$ and C_d can be transformed into a separator for $\mathrm{L}(\mathfrak{V}_1)$ and $\mathrm{L}(\mathfrak{V}_2)$. Then, we rely on the fact that a particular family $(B_k)_{k \in \mathbb{N}}$ of regular languages is a family of *basic separators* (a concept introduced by Czerwiński and the second author in [5]): Every language regularly separable from C_d is included in a finite union of sets B_k. Here, B_k contains all sequences of counter instructions such that at least one counter at some point falls below zero, but before that, it never exceeds the value k. We prove a version of this with complexity bounds: We show that $\mathrm{L}(\mathfrak{V}) \cap C_d = \emptyset$ implies that $\mathrm{L}(\mathfrak{V})$ is included in B_k for some doubly exponential bound k. Here, the key advantage is that we understand the structure of the B_k so well that we can just observe that the separator B_k is already deterministic. Thus, the complementation step will not result in another exponential blow-up.

Second upper bound. Theorem 3.1 provides all upper bounds for NFA separators in Table 1. It also provides all upper bounds for DFAs where the DFA bound is exponential in the corresponding NFA bound (via the powerset construction). The only exception to this is the dark gray entry: Here, the tight DFA bound is actually the same as for NFA.

Theorem 3.2 *Let $\mathfrak{V}_1$ and $\mathfrak{V}_2$ be 1-VASS with binary updates. If $\mathrm{L}(\mathfrak{V}_1) \cap \mathrm{L}(\mathfrak{V}_2) = \emptyset$, then there exists a separating DFA with at most exponentially many states.*

For this, we observe that the states of NFA resulting from Theorem 3.1 for $d = 1$ can be equipped with a partial ordering $\leq$ such that (i) if $p \leq q$, then all words accepted from p are also accepted from q and (ii) every antichain in this ordering has at most polynomial size. This permits determinization without a blow-up.

Lower bounds. The lower bounds for the first row in our table have already been shown in [4]. For the others, we use two types of pairs. The first is similar to the language pairs in [4]:

$$K_{f,n} = \{w \in \{a,b\} \mid \text{the } f(n)\text{-th last letter of } w \text{ is an } a \text{ and } |w| \geq f(n)\}$$
$$L_{f,n} = \{w \in \{a,b\} \mid \text{the } f(n)\text{-th last letter of } w \text{ is a } b \text{ or } |w| < f(n)\}$$

where $f\colon \mathbb{N} \to \mathbb{N}$ is one of the functions $n \mapsto n$ (a separating DFA needs 2^n states; the purple entries) or $n \mapsto 2^n$ (a separating DFA needs 2^{2^n} states, the yellow entry). In [4], these are used for $n \mapsto 2^{2^n}$. The second language pair consists of $L_n = \{a^m \mid m \geq 2^n\}$, and $K_n = \{a^m \mid m < 2^n\}$ (an NFA needs 2^n states, the light and dark gray entries).

References

[1] L. CLEMENTE, W. CZERWINSKI, S. LASOTA, C. PAPERMAN, Regular Separability of Parikh Automata. In: I. CHATZIGIANNAKIS, P. INDYK, F. KUHN, A. MUSCHOLL (eds.), *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. LIPIcs 80, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 117:1–117:13.

[2] L. CLEMENTE, W. CZERWINSKI, S. LASOTA, C. PAPERMAN, Separability of Reachability Sets of Vector Addition Systems. In: H. VOLLMER, B. VALLÉE (eds.), *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*. LIPIcs 66, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 24:1–24:14.

[3] W. CZERWINSKI, S. LASOTA, Regular Separability of One Counter Automata. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. IEEE Computer Society, 2017, 1–12.

[4] W. CZERWINSKI, S. LASOTA, R. MEYER, S. MUSKALLA, K. N. KUMAR, P. SAIVASAN, Regular Separability of Well-Structured Transition Systems. In: S. SCHEWE, L. ZHANG (eds.), *29th International Conference on Concurrency Theory (CONCUR 2018)*. Leibniz International Proceedings in Informatics (LIPIcs) 118, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, 35:1–35:18.

[5] W. CZERWIŃSKI, G. ZETZSCHE, An Approach to Regular Separability in Vector Addition Systems. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. Association for Computing Machinery, New York, NY, USA, 2020, 341–354.

[6] H. B. HUNT III, On the Decidability of Grammar Problems. *Journal of the ACM* **29** (1982) 2, 429–447.

[7] E. KESKIN, R. MEYER, Separability and Non-Determinizability of WSTS. *CoRR* **abs/2305.02736** (2023).

[8] T. G. SZYMANSKI, J. H. WILLIAMS, Noncanonical Extensions of Bottom-up Parsing Techniques. *SIAM Journal on Computing* **5** (1976) 2.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 37–37.

RPTU

k-Universality of Regular Languages

Duncan Adamson[(A)] Pamela Fleischmann[(B)] Annika Huch[(B)]
Tore Koß[(C)] Florin Manea[(C)] Dirk Nowotka[(B)]

[(A)]Leverhulme Centre for Functional Material Design, University of Liverpool, UK
d.a.adamson@liverpool.ac.uk

[(B)]Department of Computer Science, Kiel University, Germany
{fpa,dn@informatik,stu216885@mail}.uni-kiel.de

[(C)]Department of Computer Science, University of Göttingen, Göttingen, Germany
{tore.koss,florin.manea}@cs.uni-goettingen.de

Abstract

A subsequence of a word w is a word u such that $u = w[i_1]w[i_2]\dots w[i_k]$, for some set of indices $1 \leq i_1 < i_2 < \cdots < i_k \leq |w|$. A word w is k-subsequence universal over an alphabet Σ if every word in Σ^k appears in w as a subsequence. In this talk, we focus on the intersection between the set of k-subsequence universal words over some alphabet Σ and regular languages over Σ. We call a regular language L k-$\exists$-*subsequence universal* if there exists a k-subsequence universal word in L, and k-$\forall$-*subsequence universal* if every word of L is k-subsequence universal. We present algorithms solving the problems of deciding if a given regular language, represented by a finite automaton accepting it, is k-$\exists$-*subsequence universal* and, respectively, if it is k-$\forall$-*subsequence universal*, for a given number k. The algorithms are FPT w.r.t. the size of the input alphabet, and their run-time does not depend on k; they run in polynomial time in the number n of states of the input automaton when the size of the input alphabet is $O(\log n)$. Moreover, we show that the problem of deciding if a given regular language is k-$\exists$-*subsequence universal* is NP-complete, when the language is over a large alphabet. Further, we provide algorithms for counting the number of k-subsequence universal words (paths) accepted by a given deterministic (respectively, nondeterministic) finite automaton, and ranking an input word (path) within the set of k-subsequence universal words accepted by a given finite automaton.

The paper on which this talk is based is accepted at ISAAC 2023 [1].

References

[1] D. ADAMSON, P. FLEISCHMANN, A. HUCH, T. KOSS, F. MANEA, D. NOWOTKA, k-Universality of Regular Languages. In: *ISAAC 2023, Proceedings*. LIPIcs (and full version on Arxiv), to appear, 2023.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 38–40.

RPTU

Rational trace relations

Dietrich Kuske

Technische Universität Ilmenau
dietrich.kuske@tu-ilmenau.de

Rational relations form a classical and well-studied concept (cf. [15, 14, 4, 17]) that embraces homomorphisms, inverse homomorphisms as well as substitutions. Rational relations appear in the study of automatic structures [1, 18, 27], rational Kripke frames [3], graph databases [2], the representation of infinite graphs and automata [19, 25, 28, 26, 8, 7], and natural language processing [20]. One particular application of rational relations can be found in the theory of pushdown systems: the reachability relation is prefix recognizable [10, 16] and therefore a rational relation which implies that forwards and backwards reachability preserve the regularity of a set of configurations ([6] provides an alternative proof for the backwards reachability).

Also the second theme of this paper has a long and diverse research history starting with Cartier and Foata's work in combinatorics [9] and Mazurkiewicz's ideas about the semantics of concurrent systems [24] that he modelled as equivalence classes of words, called traces today. Much of the work in computer science has concentrated on recognizable sets of traces, on model checking and synthesis problems, and on combinatorics, see [11] for a comprehensive presentation of the theory of traces; many of these results have been extended to more general concurrent systems like concurrent automata (cf., e.g. [13]), message passing automata [23], and other abstract models of distributed automata (e.g. [12, 5]).

Recently, Köcher and the current author considered a generalization of pushdown systems where the stack's contents is not a word, but a trace [22]; these systems were called *cooperating pushdown systems* or cPDS. Our main results state that the forwards reachability relation preserves the rationality and the backwards reachability the recognizability of sets of configurations (but not vice versa). While the reachability relation of a classical pushdown system is prefix recognizable, we also observed that this is not the case for cPDS. In addition, Köcher [21] infered from the main result that the reachability relation is a rational trace relation, but it was not clear whether this rationality could be used to prove the preservation results as in the word case.

The first insights of this work show that rational trace relations differ significantly from rational word relations since they do not preserve rationality nor recognizability nor do they compose. To overcome these deficits, we study the restricted class $lc\mathcal{R}$ of *left-closed rational trace relations* and demonstrate that these relations enjoy many of the important properties of rational word relations: they preserve rationality (but not recognizability), their inverses preserve recognizability (but not rationality), they compose, and any rational relation is the composition of the inverse of a relation from $lc\mathcal{R}$ and a relation from $lc\mathcal{R}$.

From lemmas in [22], it follows that the reachability relation of a cPDS is a finite union of compositions of certain trace relations that resemble prefix-recognizable word relations.

We show that these "building blocks" are left-closed rational. It follows that the reachability relation of a cPDS is left-closed rational. Hence forwards reachability preserves rationality and backwards reachability preserves recognizability or sets of configurations (the main results from [22]).

Thus, the talk introduces a new class of relations and uses them to prove (parts of) the results from [22] in a more uniform and (as the author hopes) transparent way.

References

[1] V. BÁRÁNY, E. GRÄDEL, S. RUBIN, Automata-based presentations of infinite structures. In: *Finite and Algorithmic Model Theory*. Cambridge University Press, 2011, 1–76.

[2] P. BARCELÓ, D. FIGUEIRA, L. LIBKIN, Graph Logics with Rational Relations and the Generalized Intersection Problem. In: *LICS'12*. IEEE Computer Society, 2012, 115–124.

[3] W. BEKKER, V. GORANKO, Symbolic Model Checking of Tense Logics on Rational Kripke Models. In: *ILC'07*. Lecture Notes in Comp. Sciecne, Springer, 2009, 2–20.

[4] J. BERSTEL, *Transductions and context-free languages*. Teubner Studienbücher, Stuttgart, 1979.

[5] B. BOLLIG, *Formal Models of Communicating Systems - Languages, Automata, and Monadic Second-Order Logic*. Springer, 2006.

[6] A. BOUAJJANI, J. ESPARZA, O. MALER, Reachability Analysis of Pushdown Automata: Application to Model-Checking. In: *CONCUR'97*. Lecture Notes in Mathematics, vol. 1243, Springer, 1997.

[7] A. CARAYOL, A. MEYER, Context-Sensitive Languages, Rational Graphs and Determinism. *Log. Methods Comput. Sci.* **2** (2006) 2.

[8] A. CARAYOL, C. MORVAN, On Rational Trees. In: *CSL'06*. Lecture Notes in Computer Science vol. 4207, Springer, 2006, 225–239.

[9] P. CARTIER, D. FOATA, *Problèmes combinatoires de commutation et réarrangements*. Lecture Notes in Mathematics vol. 85, Springer, Berlin - Heidelberg - New York, 1969.

[10] D. CAUCAL, On the Regular Structure of Prefix Rewriting. *Theoretical Computer Science* **106** (1992), 61–86.

[11] V. DIEKERT, G. ROZENBERG, *The Book of Traces*. World Scientific Publ. Co., 1995.

[12] M. DROSTE, P. GASTIN, D. KUSKE, Asynchronous cellular automata for pomsets. *Theoretical Computer Science* **247** (2000), 1–38. (Fundamental study).

[13] M. DROSTE, D. KUSKE, Recognizable and logically definable languages of infinite computations in concurrent automata. *International Journal of Foundations of Computer Science* **9** (1998), 295–314.

[14] S. EILENBERG, *Automata, Languages and Machines vol. A*. Academic Press, New York, 1974.

[15] C. ELGOT, G. MEZEI, On relations defined by generalized finite automata. *IBM J. Res. Develop.* **9** (1965), 47–65.

[16] A. FINKEL, B. WILLEMS, P. WOLPER, A Direct Symbolic Approach to Model Checking Pushdown Systems. *Electronic Notes in Theoretical Computer Science* **9** (1997), 27–37.

[17] C. FROUGNY, J. SAKAROVITCH, Synchronized rational relations of finite and infinite words. *Theoretical Computer Science* **108** (1993), 45–82.

[18] E. GRÄDEL, Automatic Structures: Twenty Years Later. In: *LICS '20*. ACM, 2020, 21–34.

[19] J. H. JOHNSON, Rational Equivalence Relations. *Theor. Comput. Sci.* **47** (1986) 3, 39–60.

[20] J. H. JOHNSON, Uniformizing Rational Relations for Natural Language Applications Using Weighted Determinization. In: *CIAA'10*. Lecture Notes in Computer Science vol. 6482, Springer, 2010, 173–180.

[21] C. KÖCHER, *Verification of Automata with Storage Mechanisms*. TU Ilmenau, 2022. Doctoral dissertation.

[22] C. KÖCHER, D. KUSKE, Forwards- and Backwards-Reachability for Cooperating Multi-Pushdown Systems. In: *FCT'23*. 2023. Accepted.

[23] D. KUSKE, A. MUSCHOLL, Communicating Automata. In: J.-E. PIN (ed.), *Handbook of Automata Theory*. 2, EMS Press, 2021, 1147–1188.

[24] A. MAZURKIEWICZ, *Concurrent program schemes and their interpretation*. Technical report, DAIMI Report PB-78, Aarhus University, 1977.

[25] C. MORVAN, On rational graphs. In: *FOSSACS'00*. Lecture Notes in Comp. Science vol. 1784, Springer, 2000, 252–266.

[26] C. MORVAN, C. STIRLING, Rational graphs trace context-sensitive languages. In: *MFCS'01*. Lecture Notes in Comp. Science vol. 2136, Springer, 2001, 548–559.

[27] S. RUBIN, Automatic structures. In: J.-E. PIN (ed.), *Handbook of Automata Theory*. EMS Press, 2021, 1031–1070.

[28] W. THOMAS, A short introduction to infinite automata. In: *DLT'01*. Lecture Notes in Comp. Science vol. 2295, Springer, 2002, 130–144.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 41–45.

RPTU

Error-Correcting Parsing – This Time We Want All!

Florian Bruse Stefan Kablowski Martin Lange

Theoretische Informatik / Formale Methoden, Universität Kassel
{florian.bruse, martin.lange}@uni-kassel.de

The Problem and its Motivation. A well-known problem in the theory of formal languages is that of error-correcting parsing: given a, say, context-free language L over some alphabet Σ, and a word $w \in \Sigma^*$, compute a word $v \in L(G)$ s.t. $\Delta(w, v)$ is minimal, where $\Delta : \Sigma^* \times \Sigma^* \to \mathbb{R}^{\geq 0}$ is some fixed distance metric. Here we are are solely concerned with the Levenshtein metric [9] which measures distance between two words as the minimal number of insertion, deletion or replacement operations on letters that turn one of them into the other.

It has been shown that error-correcting parsing (for context-free languages) is conceptually not much more difficult than ordinary parsing; there are solutions based on Earley's parser or the CYK algorithm which, given some w, compute a parse tree for some $v \in L(G)$ and a minimal sequence of edit operations that turn w into v, cf. [1, 10].

The requirement of minimality in the formulation of error-correcting parsing is important. Suppose a file contains a syntactically misshaped Java program. There are of course many ways to edit it to become a correct one, for instance by deleting it entirely and inserting the infamous hello-world program. Such a correction is not minimal in general, though. Minimality allows us not to introduce a notion of semantical distance between correctly formed and ill-formed programs; instead we simply assume that the faulty program originated from a correct one, and the process that introduced syntactic errors is governed by statistical laws so that the program that is obtained by applying a minimal correction is most likely the original one.

There are, however, applications of parsing in which the reason for some w not belonging to L cannot be found in some "original" $v \in L$ which has been modified to w under laws of statistics. Consider the following scenario. The curricula of natural sciences secondary-education classes typically contain experimental lessons whose purpose it is to teach pupils the principles of scientific discovery and reasoning. They are given a research question and are asked to formulate a matching hypothesis and then to (in-)validate it using some experimental setup. The Theoretical Computer Science / Formal Methods group at the Univ. of Kassel is involved in the development of a digital learning tool that initiates adaptive learning by giving feedback on each step of the process, from formulating a hypothesis to checking its (semantic) correctness [7]. The set of syntactically correct hypotheses can easily be formalised by a context-free grammar, and it is not hard to imagine that hypotheses formulated by some 8th-grade pupil are not always grammatically correct. However, here it is neither right nor helpful to automatically apply a minimal correction in order to continue with the semantical checks, for the following two reasons.

- The cause of syntactical incorrectness may be more than merely a typo; it could be that the pupil has not fully understood the grammatical structure of hypotheses yet.
- For learning purposes, it is better to present the pupil with some feedback on why his/her formulation is misshaped and let them correct it.

This leads to the following problem UECP of *universal* error-correcting parsing.

given: a context-free language L over some alphabet Σ, and a word $w \in \Sigma^*$
compute: the set of *all* minimal corrections ρ s.t. $\rho(w) \in L$

As it turns out, in order to suit the application sketched above, we also need a more relaxed notion of minimality. Consider, for example, the following attempt at formulating a hypothesis w.r.t. the research question "*Does temperature influence yeast growth?*"

yeast grows it is warm

An obvious way to correct this would be to insert the word *when* in position 2, forming "*yeast grows when it is warm*." So the edit distance to a correctly formed hypothesis is 1, and there are also other ways to execute a single edit to form a correct sentence with potentially different meaning, for instance inserting *because*, *if*, *and*, etc. However, maybe the author of this pre-hypothesis has a different idea of causality and actually tried to state

if yeast grows then it is warm

or they actually rightly predicted another aspects of the influence between temperature and yeast growth but failed to formulate

yeast grows unless it is hot

at edit distance 2. So while these do not reside at an edit distance of *minimal length*, the former should definitely be considered to be a minimal correction in the sense that it results from a minimal set of edit operations (here: insertions only) that create a valid sentence. The notion of minimality that is formally defined below, does not capture the latter, though. Note that the two edit operations – inserting "*unless*" at position 2 and replacing "*warm*" with "*hot*" at position 5 – are independent; they can be carried out in any order (with appropriate adjustments to the index positions). There is one particular order, namely the one stated here, applying the insertion before the replacement, which leads to an intermediate word in the language, namely "*yeast grows unless it is warm*." This is why we do not consider this correction to be minimal – there is an order of its edit operations which produces a word in the language before all edits are being carried out.

A Theory of Minimality in Corrections. A *deletion*, resp. *insertion operation* is written $a{\downarrow_i}$, resp. $a{\uparrow_i}$ for $a \in \Sigma$, $i \in \mathbb{N}$. A *replacement operation* is written $a/_i b$ for $a, b \in \Sigma$, $i \in \mathbb{N}$. An *(edit) operation* is either of these three.

Each operation α induces a partial map of type $\Sigma^* \to \Sigma^*$, straight-forwardly realising the effect of deleting, inserting or replacing a symbol at a particular position in a word. A *correction* is a (possibly empty) sequence $\rho = (\alpha_1, \ldots, \alpha_m)$ of operations. The effect of the application of

a correction to a word, or simply *correcting* the word, is explained by a homomorphic extension of the effect that singular edit operations have: $\rho(w) := \alpha_m(\ldots\alpha_1(w)\ldots)$.

Two corrections ρ, ρ' are *equivalent* if $\rho(w) = \rho'(w)$ for all $w \in \Sigma^*$. A correction ρ is *normalised* if

$$\rho = (a_1/_{i_1}b_1, \ldots a_n/_{i_n}b_n, c_1{\downarrow_{j_1}}, \ldots, c_m{\downarrow_{j_m}}, d_1{\uparrow_{h_1}}, \ldots, d_k{\uparrow_{h_k}}) \tag{1}$$

for some n, m, k s.t. $i_1 > \ldots > i_n$, $j_1 > \ldots > j_m$ and $h_1 \geq \ldots \geq h_k$.

It is possible to define rules of a rewrite system $\rightarrow$ operating on pairs of edit operations that are sound w.r.t. equivalence. For instance, we would have

$$\rho, \mathit{unless}{\uparrow_2}, \mathit{warm}/_5\mathit{hot}, \rho' \;\; \rightarrow \;\; \rho, \mathit{warm}/_4\mathit{hot}, \mathit{unless}{\uparrow_2}, \rho'$$

for any ρ, ρ'. Likewise, some combinations cancel each other out like a deletion of a letter following its insertion, and other pairs can be shortened; e.g. a deletion followed by an insertion of a different letter at the same position can be rewritten into a replacement. We leave it as an exercise to formulate up to $3 \cdot 3 \cdot 2 = 18$ rules covering the cases in which an operation of one of the three types is followed by another at either the same or the succeeding position in a word.

Proposition 1 *Every correction ρ is equivalent to a normalised ρ' s.t. $\rho \rightarrow^* \rho'$.*

Hence, it suffices to only consider normalised corrections henceforth. We write $\rho' \preceq \rho$ if ρ' is a subsequence of ρ. We say that ρ is a *$\preceq$-minimal correction* (for some CFG L and a word w), if $\rho(w) \in L$ and $\rho'(w) \notin L$ for every $\rho' \prec \rho$. We write $\mathcal{C}_L(w)$ for the set of normalised ρ s.t. $\rho(w) \in L$, and $\mathcal{C}_L^{\mathsf{min}}(w)$ for the set of $\rho \in \mathcal{C}_L(w)$ that are $\preceq$-minimal. The following result is important in order to make UECP well-defined.

Proposition 2 *Let L be a CFL, $w \in \Sigma^*$. Then (I) $\mathcal{C}_L(w)$ is a context-free language over a finite alphabet of edit operations, and (II) $\mathcal{C}_L^{\mathsf{min}}(w)$ is finite.*

Computing Minimal Corrections: Theory and Practice. Prop. 1 can be used as a basis for a simple but highly inefficient enumeration procedure for solving UECP [5]: given L and w, enumerate all corrections ρ in normal form and check for each of them whether

- $\rho(w) \in L$ by computing $\rho(w)$ straight-forwardly and then using a standard parsing algorithm for CFLs, and
- then compute the necessarily finitely many $\rho' \prec \rho$ and equally check $\rho'(w) \notin L$ for all of them.

Part (II) of Prop. 2 ensures that this procedure can be terminated at some point.

There is, however, a better way to solve UECP by internalising the construction of (minimal) corrections into the parser's work. Just as an ordinary (non-error-correcting) context-free parsing, the problem opens itself up to a solution using dynamic programming as corrections for a language L and word w can be built from corrections for the subwords of w and potentially different languages. It is not clear, though, whether minimality can be maintained in such a modular way, too.

There is a conceptually simple way to extend the CYK algorithm [11, 6, 12, 3] to UECP. Given a CFG G and a word $w = a_0 \ldots a_{n-1}$, we maintain, likewise, a table T of entries for each

subword represented by a pair (i,j) with $i \leq j$. However, unlike the original CYK algorithm which only stores a set of nonterminals A in entry (i,j) s.t. $A \Rightarrow^* a_i \dots a_j$, we store a set of pairs of nonterminals and minimal corrections (A,ρ) s.t. $A \Rightarrow^* \rho(a_i \dots a_j)$. We then just need the following amendments, resp. adjustments.

- A table entry $T(i,i)$ is filled with pairs (A,ε) whenever $A \to a_i$ as in CYK, and additionally
  - with pairs $(A, b/_i a_i)$ whenever $A \to b$,
  - with pairs $(A, a_i{\downarrow_i})$ whenever $A \to \varepsilon$.
- We get $(A,\rho) \in T(i,j)$ for $j \geq i$, whenever $A \to BC$ and there are h with $i \leq h < j$, ρ', ρ'' s.t. $(B,\rho') \in T(i,h)$, $(C,\rho'') \in T(h+1,j)$ and ρ is the normalisation of $\rho'\rho'''$ where ρ''' results from ρ'' by shifting all indices by the number of insertion operations minus the number of deletion operations in ρ'.

  Note that this can potentially add multiple entries with the same nonterminal in a table entry. Whenever $(A,\rho),(A,\sigma) \in T(i,j)$ and $\rho \preceq \sigma$ then (A,σ) is removed from $T(i,j)$.
- At last, note that so far, no insertion operations are generated. We first observe that a sequence of insertions operating consecutively on a word can be ordered and grouped into parts that consecutively insert letters at the same position. In the special case of the word to apply them to being ε we easily see that sequences of insertions of the form $a{\uparrow_0}$ for some $a \in \Sigma$ suffice to turn ε into any target word. It then only remains to see that it suffices to pre-compute, for any nonterminal A, a set I of pairs of nonterminals A and minimal pure insertion corrections $\sigma = b_0{\uparrow_0} \dots b_{\ell-1}{\uparrow_0}$ s.t. $A \Rightarrow b_{\ell-1} \dots b_0$. These can be pre-computed once and then used in the following way to additionally fill table entries (i,j) with $i \leq j$.
  - Whenever $A \to BC$, $(B,\rho) \in T(i,j)$ and $(C,\sigma) \in I$, then add (A,ρ') to $T(i,j)$ where ρ' is the normalisation of $\rho \cdot \sigma'$ and σ' is obtained from σ by setting all position indices to $j+1$ plus the difference of insertions and deletions in ρ as above.
  - Whenever $A \in BC$, $(C,\rho) \in T(i,j)$ and $(B,\sigma) \in I$, then add (A,ρ') to $T(i,j)$ where ρ' is the normalisation of $\sigma \cdot \rho''$ and ρ'' is obtained from ρ by shifting all indices by $|\sigma|$.

Tests run with an OCaml implementation of this algorithm are promising in that it is possible to compute sets of minimal corrections for grammars with dozens of rules. In order to avoid costly normalisation in the grammar we build on a CYK variant that does not require Chomsky normal form [8]. The benchmarks also show, however, that sets of minimal corrections need not be small, and that in the light of the targeted application described above, it may be useful to further relax the notion of $\prec$-minimality s.t. that nature of computing more than just one correction is sufficiently retained.

We also aim to investigate the possibility to build a solution for UECP based on the Earley parser [4, 2] to see whether this would lead to a more efficient solution than the CYK-based one.

References

[1] A. V. AHO, T. G. PETERSON, A minimum distance error correcting parser for context-free languages. *SIAM Journal on Computing* **1** (1972) 4, 305–312.

[2] J. AYCOCK, R. N. HORSPOOL, Practical Earley Parsing. *The Computer Journal* **45** (2002) 6, 620–630.

[3] J. COCKE, J. T. SCHWARTZ, *Programming Languages and Their Compilers*. Courant Institute of Mathematical Sciences, New York, 1970.

[4] J. EARLEY, An Efficient Context-Free Parsing Algorithm. *Communications of the ACM* **13** (1970), 94–102.

[5] S. KABLOWSKI, *Computing All Minimal Corrections for a Word to Match a Context-Free Description*. B.sc. thesis, Univ. of Kassel, Germany, Faculty of Electr. Eng. and Comp. Sci., 2022. `https://www.uni-kassel.de/eecs/tifm/abschlussarbeiten/abg`

[6] T. KASAMI, *An efficient recognition and syntax analysis algorithm for context-free languages*. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts, 1965.

[7] M. KASTAUN, M. MEIER, N. HUNDESHAGEN, M. LANGE, ProfiLL: Professionalisierung durch intelligente Lehr-Lernsysteme. In: *Bildung, Schule, Digitalisierung*. Waxmann-Verlag, 2020, 357–363.

[8] M. LANGE, H. LEISS, To CNF or not to CNF? An Efficient Yet Presentable Version of the CYK Algorithm. *Informatica Didactica* **8** (2009).

[9] V. I. LEVENSHTEIN, Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* **10** (1966) 8, 707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[10] S. RAJASEKARAN, M. NICOLAE, An Error Correcting Parser for Context Free Grammars that Takes Less Than Cubic Time. In: *Proc. 10th Int. Conf. on Language and Automata, Theory and Applications, LATA'16*. LNCS 9618, Springer, 2016, 533–546.

[11] I. SAKAI, Syntax in universal translation. In: *Proc. Int. Conf. on Machine Translation of Languages and Applied Language Analysis*. 1961.

[12] D. H. YOUNGER, Recognition and parsing of context-free languages in time n^3. *Information and Control* **10** (1967) 2, 372–375.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 46–50.

RPTU

Lyndon Partial Arrays

Meenakshi Paramasivan[(A)]

[(A)]Faculty of Business Administration, Economics, Social Sciences, Mathematics and Computer Sciences, Universität Trier, 54286, Trier, Germany

Abstract

Lyndon words have been extensively studied in different contexts of free Lie algebra and combinatorics. Lyndon partial words, arrays and trees have been recently introduced by us and we study on free monoid morphisms that preserve finite Lyndon partial words and check whether a morphism preserves or does not preserve the lexicographic order. We proposed an algorithm to determine Lyndon partial words of given length over the binary alphabet. Image analysis in several way of scanning via automata and grammars has a significance in two-dimensional models, we connect 2D Lyndon partial words with few automata and grammar models.

1. Introduction

Partial words are nothing but words with holes over the alphabet. The study of partial words was initiated by Berstel and Boasson [1] and later the work was extended by Blanchet Sadri [3, 4]. Lyndon words serve to be a useful tool for a variety of problems in combinatorics. There are many applications of Lyndon words in semigroups, pattern matching, representation theory of certain algebras and combinatorics such as they are used to describe the generators of the free Lie algebras. All of these applications make use of the combinatorial properties of Lyndon words, in particular the factorisation theorem. Their role in factorising a string over an ordered alphabet was initially illustrated by Chen et.al [5]. Duval [7] presented a algorithm to derive a factorisation of strings over an ordered alphabet known as Lyndon factorisation. Lyndon trees [6] are associated with Lyndon words under the name of standard lexicographic sequences. The Lyndon arrays[9, 2] of Lyndon words has recently become of interest since it could be used to efficiently compute all the maximal periodicities in a word. Both Lyndon and partial words have wide application in pattern matching. In [8], the authors have derived an automaton model namely Boustrophedon finite automata (BFA) for picture processing, which is equivalent to Regular matrix grammars (RMGs). The paper has the following organisation. In Section 2 we introduce Lyndon partial words and Lyndon partial arrays. A relation between the Lyndon partial words and trees is established. In Section 3 we characterise $\ell_\diamond$-morphism and show that they are order-preserving morphism. In Section 4 we investigate few connections to 2D Lyndon words through 2D Lyndon partial words.

2. Lyndon Partial Words

Here we introduce and study the generalisation of finite Lyndon partial words by using trees. In [11], the authors have defined that a primitive partial word is a partial Lyndon word if and only if it is minimal in its conjugate class with respect to alphabetical order by assuming the order of $\Diamond$ as $\{a \prec b \prec \ldots \prec \Diamond\}$. The order of $\Diamond$ does not play a special role in the definition by studying properties of partial Lyndon words since the $\Diamond$ is considered as a letter with highest order which makes the definition similar to that of Lyndon words. In our definition of Lyndon partial word, the order of $\Diamond$ plays a special role in studying certain properties.

Definition 2.1 *A* Lyndon partial word $l_\Diamond = l_\Diamond[1 \ldots n]$ *over the ordered alphabet* $\Sigma_{k,\Diamond} = \Sigma_k \bigcup \{\Diamond\} = \{a_1 \prec a_2 \prec \ldots \prec a_k\} \bigcup \{\Diamond\}, k > 1$ *is less than all its conjugates (rotations) with respect to the alphabetical order. Here the order of* $\Diamond$ *is considered as* $a_1 \preceq \Diamond$, $\Diamond \preceq a_k$ *and* $\Diamond$ *is compatible with all other elements of* Σ_k. *A* Lyndon partial language *over* Σ *is a subset of* $\Sigma^*_\Diamond$, *the set of all Lyndon partial words over* $\Sigma_\Diamond$.

For readability we use $L_\Diamond$ notation for partial languages which shall not be confused with the $\ell_\Diamond$ notation for Lyndon partial languages. Table 1 shows the set of all Lyndon partial words with length at most five over the ordered alphabet $\Sigma_\Diamond = \{a \prec b\} \bigcup \{\Diamond\}$.

Remark 2.2 *It is easy to observe that Lyndon partial words on binary alphabet takes the same integer sequence starting from 2, 3, 6, 9 by excluding the first three numbers namely 1, 2, 1 of that of Lyndon words as compared and evidenced in Table 1.*

Definition 2.3 *A Lyndon partial factor* $l_\Diamond[i \ldots j]$ *of a Lyndon partial word* $l_\Diamond[i \ldots n]$ *for any* $j \leq n$ *is* a maximal Lyndon partial factor *if it is Lyndon.*

Definition 2.4 *A* Lyndon partial array *(denoted as* $l^A_\Diamond$*) of* $l_\Diamond[1 \ldots n]$ *is an array of integers in the range* $[1 \ldots n]$ *such that, at each position* $i = 1 \ldots n$ *stores the length of the longest Lyndon partial factor of* $l_\Diamond[1 \ldots n]$ *starting at* i.

Example 2.5 *Consider a Lyndon partial word* $l_\Diamond[1 \ldots 7] = aabab\Diamond b$. *The maximal Lyndon partial factor starting at position 1 is* $aabab$, *so* $l^A_\Diamond[1] = 5$. *The maximal Lyndon partial factor at position 2 is* ab, *so* $l^A_\Diamond[2] = 3$. *The maximal Lyndon partial factor starting at position 3 is* b, *so* $l^A_\Diamond[3] = 3$. *The maximal Lyndon partial factor starting at position 4 is* ab, *so* $l^A_\Diamond[4] = 5$. *The maximal Lyndon partial factor starting at position 5 is* b, *so* $l^A_\Diamond[1] = 5$. *The maximal Lyndon partial factor starting at position 6 is* $\Diamond b$, *so* $l^A_\Diamond[6] = 7$. *The maximal Lyndon partial factor starting at position 7 is* b, *so* $l^A_\Diamond[7] = 7$. *Therefore,* $l^A_\Diamond = [5\ \ 3\ \ 3\ \ 5\ \ 5\ \ 7\ \ 7]$.

Definition 2.6 *A* tree ζ associated with a Lyndon partial word *is described with its minimal among all of its rotations.* $\Im$ *denotes set of such trees. A sub-tree of* ζ *is a tree with set of nodes as a subset of* ζ.

Theorem 2.7 *No proper sub-tree exists as both initial and terminal of the tree* ζ.

Theorem 2.8 ζ *is a tree of a Lyndon partial word if and only if* $\zeta = P + vQ, v \in \delta(P)$ *where* $\zeta, P, Q \in \Im$ *and* $P \prec Q$.

Table 1: Lyndon words along with Lyndon partial words

Length	Lyndon words	Lyndon partial words
0	λ	-
1	a, b	-
2	ab	$a\Diamond, \Diamond b$
3	aab, abb	$aa\Diamond, a\Diamond b, \Diamond bb$
4	$aaab$, $aabb$, $abbb$	$aaa\Diamond, aa\Diamond b, a\Diamond ab, a\Diamond bb, ab\Diamond b, \Diamond bbb$
5	$aaaab, aaabb, aabab,$ $aabbb, ababb, abbbb$	$aaaa\Diamond, aaa\Diamond b, aa\Diamond ab, aa\Diamond bb, aab\Diamond b,$ $a\Diamond abb, a\Diamond bbb, ab\Diamond bb, \Diamond bbbb$
$\vdots$	$\ldots$	$\ddots$

Theorem 2.9 *Any tree ζ over the alphabet $\Sigma_\Diamond^+$ can be uniquely written as $\zeta = P_0 + v_1P_1 + v_2P_2 + ...v_kP_k, v_m \in \delta(v_nP_n)$ for some $n \succeq m$ such that $P_0 \succeq P_1 \succeq P_2.... \succeq P_k$.*

3. $\ell_\Diamond$ - Morphism

In this section we characterise $\ell_\Diamond$-morphism and show that they are order-preserving morphism. A non-empty morphism g over an ordered alphabet $\Sigma_{k,\Diamond}$ containing atleast two letters is an order-preserving morphism if for all partial words $r_\Diamond, s_\Diamond$ over $\Sigma_\Diamond$, $r_\Diamond \prec s_\Diamond \Rightarrow g(r_\Diamond) \prec g(s_\Diamond)$.

Definition 3.1 *Consider two ordered alphabets $U_\Diamond$ and $V_\Diamond$ each containing atleast two letters such that a morphism g from $U_\Diamond^*$ to $V_\Diamond^*$ is called a $\ell_\Diamond$-* morphism *if for any Lyndon partial words $l_\Diamond$ over $U_\Diamond$, $g(l_\Diamond)$ is a Lyndon partial word over $V_\Diamond$. In short a morphism that preserves the property of Lyndon partial words is defined as $\ell_\Diamond$ - morphism.*

Theorem 3.2 *A non-empty morphism g on $\Sigma_\Diamond^+$ containing atleast two letters is a $\ell_\Diamond$- morphism if and only if g is an order preserving morphism such that for each $u_\Diamond \in \Sigma_\Diamond$, $g(u_\Diamond)$ is a Lyndon partial word.*

Corollary 3.3 *g is a $\ell_\Diamond$- morphism on $\Sigma_\Diamond = \{a, b\} \cup \{\Diamond\}$ if and only if $g(a)$ and $g(b)$ are Lyndon partial words with $g(a) \prec g(b)$.*

4. Two-dimensional Lyndon partial words

The concept of Lyndon words are extended as two-dimensional Lyndon words in [10]. Those are useful to capture 2D horizontal periodicity of a matrix in which each row is highly periodic. It is also utilised to solve 2D horizontal suffix–prefix matching among a set of rectangular patterns efficiently. We introduce the following.

Definition 4.1 *A* two-dimensional row Lyndon partial word *is a horizontally primitive matrix which is least among its horizontal conjugates.*

Definition 4.2 *A* regular two-dimensional Lyndon partial word *is a horizontally primitive matrix which is least of its horizontal conjugates by maintaining a regular order.*

In one-dimensional case Lyndon partial words of length 4 over binary alphabet are $aaa\Diamond$, $aa\Diamond b$, $a\Diamond ab$, $a\Diamond bb$, $ab\Diamond b$, $\Diamond bbb$. Now we can derive two-dimensional partial words as follows where there will be many 2D partial words, few sample of those 2D partial words are given in Example below which maintains a specific/regular ordering of Lyndon partial arrays.

Example 4.3 $\begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ \Diamond\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ \Diamond\ a\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond\ b \\ a\ a\ \Diamond\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond\ b \\ a\ \Diamond\ a\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond\ b \\ a\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ \Diamond\ a\ b \\ a\ \Diamond\ a\ b \end{matrix},$

$\begin{matrix} a\ \Diamond\ a\ b \\ a\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ \Diamond\ a\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ \Diamond\ b\ b \\ a\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \end{matrix}, \cdots, \begin{matrix} \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \end{matrix}, \cdots, \begin{matrix} \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond \end{matrix}, \cdots, \begin{matrix} \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ \Diamond\ b \\ a\ \Diamond\ a\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \cdots, \begin{matrix} \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}.$

One can observe that these 6 elements that follows are similar to many other elements which are NOT present in above collection: $\begin{matrix} \Diamond\ a\ a\ a \\ b\ a\ a\ \Diamond \\ b\ a\ \Diamond\ a \\ b\ a\ \Diamond\ b \\ b\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ \Diamond\ a\ a \\ \Diamond\ b\ a\ a \\ a\ b\ a\ \Diamond \\ b\ b\ a\ \Diamond \\ b\ b\ \Diamond\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond\ b \\ a\ \Diamond\ a\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \\ a\ a\ a\ \Diamond \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ \Diamond\ a\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \\ a\ \Diamond\ b\ a \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ \Diamond\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \\ a\ \Diamond\ a\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ \Diamond\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \\ a\ \Diamond\ b\ b \end{matrix}$ as these do not satisfy the property of being a regular 2D Lyndon partial word. Now it is of interest to identify specific/regular 2D Lyndon partial words among those elements. Due to Remark 2.2, we see the following pattern collected as a 2D partial language which shall be named as L_{DD} for further references in our work based on [12].

$$L_{DD} = \left\{ \begin{matrix} a\ \Diamond \\ \Diamond\ b \end{matrix}, \begin{matrix} a\ a\ \Diamond \\ a\ \Diamond\ b \\ \Diamond\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ \Diamond \\ a\ a\ \Diamond\ b \\ a\ b\ \Diamond\ b \\ a\ \Diamond\ a\ b \\ a\ \Diamond\ b\ b \\ \Diamond\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ a\ \Diamond \\ a\ a\ a\ \Diamond\ b \\ a\ a\ b\ \Diamond\ b \\ a\ a\ \Diamond\ a\ b \\ a\ a\ \Diamond\ b\ b \\ a\ b\ \Diamond\ b\ b \\ a\ \Diamond\ a\ b\ b \\ a\ \Diamond\ b\ b\ b \\ \Diamond\ b\ b\ b\ b \end{matrix}, \begin{matrix} a\ a\ a\ a\ a\ \Diamond \\ a\ a\ a\ a\ \Diamond\ b \\ a\ a\ a\ b\ \Diamond\ b \\ a\ a\ b\ b\ \Diamond\ b \\ a\ a\ a\ \Diamond\ a\ b \\ a\ a\ a\ \Diamond\ b\ b \\ a\ a\ b\ \Diamond\ a\ b \\ a\ a\ b\ \Diamond\ b\ b \\ a\ a\ \Diamond\ a\ a\ b \\ a\ a\ \Diamond\ a\ b\ b \\ a\ a\ \Diamond\ b\ a\ b \\ a\ a\ \Diamond\ b\ b\ b \\ a\ \Diamond\ a\ a\ a\ b \\ a\ \Diamond\ a\ b\ b\ b \\ a\ \Diamond\ b\ b\ b\ b \\ \Diamond\ b\ b\ b\ b\ b \end{matrix}, \cdots, \right\}.$$

References

[1] J. BERSTEL, L. BOASSON, Partial Words and a Theorem of Fine and Wilf. *Theor. Comput. Sci.* **218** (1999) 1, 135–141.

[2] P. BILLE, J. ELLERT, J. FISCHER, I. L. GØRTZ, F. KURPICZ, J. I. MUNRO, E. ROTENBERG, Space Efficient Construction of Lyndon Arrays in Linear Time. In: A. CZUMAJ, A. DAWAR, E. MERELLI (eds.), *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*. LIPIcs 168, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 14:1–14:18.

[3] F. BLANCHET-SADRI, Primitive partial words. *Discret. Appl. Math.* **148** (2005) 3, 195–213.

[4] F. BLANCHET-SADRI, K. GOLDNER, A. SHACKLETON, Minimal partial languages and automata. *RAIRO Theor. Informatics Appl.* **51** (2017) 2, 99–119.

[5] K. T. CHEN, R. H. FOX, R. C. LYNDON, Free differential calculus, IV. The quotient groups of the lower central series. *Annals of Mathematics* (1958), 81–95.

[6] M. CROCHEMORE, L. M. RUSSO, Cartesian and Lyndon trees. *Theoretical Computer Science* **806** (2020), 1–9.

[7] J. DUVAL, Factorizing Words over an Ordered Alphabet. *J. Algorithms* **4** (1983) 4, 363–381.

[8] H. FERNAU, M. PARAMASIVAN, M. L. SCHMID, D. G. THOMAS, Simple picture processing based on finite automata and regular grammars. *J. Comput. Syst. Sci.* **95** (2018), 232–258.

[9] F. FRANEK, A. S. M. S. ISLAM, M. S. RAHMAN, W. F. SMYTH, Algorithms to Compute the Lyndon Array. In: J. HOLUB, J. ZDÁREK (eds.), *Proceedings of the Prague Stringology Conference 2016, Prague, Czech Republic, August 29-31, 2016*. Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, 2016, 172–184.

[10] S. MARCUS, D. SOKOL, 2D Lyndon Words and Applications. *Algorithmica* **77** (2017) 1, 116–133.

[11] A. C. NAYAK, K. KAPOOR, On the Language of Primitive Partial Words. In: A. DEDIU, E. FORMENTI, C. MARTÍN-VIDE, B. TRUTHE (eds.), *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*. Lecture Notes in Computer Science 8977, Springer, 2015, 436–445.

[12] M. PARAMASIVAN, R. K. KUMARI, R. ARULPRAKASAM, V. R. DARE, Lyndon Partial Words and Arrays with Applications. In: R. P. BARNEVA, V. E. BRIMKOV, G. NORDO (eds.), *Combinatorial Image Analysis - 21st International Workshop, IWCIA 2022, Messina, Italy, July 13-15, 2022, Proceedings*. Lecture Notes in Computer Science 13348, Springer, 2022, 226–244.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 51–53.

RPTU

The Pumping Lemma for Regular Languages is Hard

Hermann Gruber[(A)] Markus Holzer[(B)] Christian Rauch[(B)]

[(A)]Planerio GmbH, Gewürzmühlstr. 11, 80538 München
h.gruber@planerio.de

[(B)]Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen
{holzer,christian.rauch}@informatik.uni-giessen.de

The automata theory and formal languages curriculum introduces pumping lemmata for regular and context-free languages to demonstrate non-regularity or non-context-freeness in specific cases. Variations of these lemmata are taught based on instructor preferences and chosen materials. For example, refer to the pumping lemma in [6, page 70, Theorem 11.1], which outlines a key criterion for language regularity.

Lemma 1 *Let L be a regular language over Σ. Then, there is a constant p (depending on L) such that the following holds: If $w \in L$ and $|w| \geq p$, then there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$ and $xy^tz \in L$ for $t \geq 0$—it is then said that y can be* pumped *in w.*

A lesser-known pumping lemma, attributed to Jaffe [5], characterizes the regular languages, by describing a necessary and sufficient condition for languages to be regular. For other pumping lemmata see, e.g., the annotated bibliography on pumping [7]:

Lemma 2 *A language L is regular if and only if there is a constant p (depending on L) such that the following holds: If $w \in \Sigma^*$ and $|w| = p$, then there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$ and*[1]

$$wv = xyzv \in L \iff xy^tzv \in L$$

for all $t \geq 0$ and each $v \in \Sigma^*$.

For a regular language L the value of p in Lemma 1 can always be chosen to be the number of states of a finite automaton, regardless whether it is deterministic (DFA) or nondeterministic (NFA), accepting L. Sometimes an even smaller number suffices. For instance, the language

$$L = a^* + a^*bb^* + a^*bb^*aa^* + a^*bb^*aa^*bb^*,$$

is accepted by a (minimal) deterministic finite automaton with five states, the sink state included, but for $p = 1$ the statement of Lemma 1 is satisfied since regardless whether the considered word

This is a summary of a paper presented at the 27th International Conference on Implementation and Application of Automata (CIAA) held in Famagusta, Cyprus, September 19–22, 2023.

[1]Observe that the words $w = xyz$ and xy^tz, for all $t \geq 0$, belong to the same Myhill-Nerode equivalence class of the language L. Thus, one can say that the pumping of the word y in w *respects equivalence classes*.

starts with a or b, this letter can be readily pumped. For Lemma 2 the situation is even more involved and we refer to [2] and [3] for a detailed discussion on that subject. This gives rise to the definition of the LANGUAGE-PUMPING-PROBLEM or for short PUMPING-PROBLEM:

INPUT: a finite automaton A and a natural number p, i.e., an encoding $\langle A, 1^p \rangle$.

OUTPUT: Yes, if and only if the statement from Lemma 1 holds for the language $L(A)$ w.r.t. the value p.

A similar definition applies when considering the condition of Lemma 2 instead.

These problems turn out to be surprisingly difficult, even in the case of deterministic finite automata as inputs. The following table summarizes our findings for finite automata in general. The **coNP**-hardness result for NFAs gives us a nice non-approximability by-product under the

	PUMPING-PROBLEM w.r.t. ...	
	Lemma 1	Lemma 2
DFA	**coNP**-complete	
NFA	**coNP**-hard contained in $\Pi_2^{\mathbf{P}}$	**PSPACE**-complete

Table 1: Complexity of the PUMPING-PROBLEM for variants of finite state devices in general.

assumption of the so-called *Exponential-Time Hypothesis (ETH)* [1, 4]: there is no deterministic algorithm that solves 3SAT in time $2^{o(n+m)}$, where n and m are the number of variables and clauses, respectively. More precisely we find the following non-approximability statement:

Theorem 1 *Let A be an NFA with s states, and let δ be a constant such that $0 < \delta \leq 1/2$. Then no deterministic $2^{o(s^\delta)}$-time algorithm can approximate the minimal pumping constant w.r.t. Lemma 1 (Lemma 2, respectively) within a factor of $o(s^{1-\delta})$, unless ETH fails.*

When considering restricted automata such as, e.g., unary automata, the situation changes dramatically. For NFAs the **coNP**-hardness result and thus its intractability remains for both considered pumping lemmata, while for DFAs the problem becomes efficiently solvable. More precisely, for both pumping lemmata the PUMPING-PROBLEM can be shown to be complete for deterministic logspace **L** under weak reductions.

References

[1] M. CYGAN, F. FOMIN, Ł. KOWALIK, D. LOKSHTANOV, D. MARX, M. PILIPCZUK, M. PILIPCZUK, S. SAURABH, *Parameterized Algorithms*, chapter Lower Bounds Based on the Exponential-Time Hypothesis. Springer, 2015, 467–521.

[2] H. GRUBER, M. HOLZER, C. RAUCH, The Pumping Lemma for Regular Languages is Hard. In: B. NAGY (ed.), *Proceedings of the 27th International Conference on Implementation and Application of Automata*. Number 14151 in LNCS, Springer, Famagusta, Cyprus, 2023, 128–140.

[3] M. HOLZER, C. RAUCH, On Jaffe's Pumping Lemma, Revisited. In: H. BORDIHN, N. TRAN, G. VASZIL (eds.), *Proceedings of the* 25*th International Conference on Descriptional Complexity of Formal Systems*. Number 13918 in LNCS, Springer, Potsdam, Germany, 2023, 65–78.

[4] R. IMPAGLIAZZO, R. PATURI, F. ZANE, Which Problems Have Strongly Exponential Complexity? *J. Comput. System Sci.* **63** (2001) 4, 512–530.

[5] J. JAFFE, A necessary and sufficient pumping lemma for regular languages. *SIGACT News* **10** (1978) 2, 48–49.

[6] D. C. KOZEN, *Automata and Computability*. Undergraduate Texts in Computer Science, Springer, 1997.

[7] A. NIJHOLT, YABBER—Yet Another Bibliography: Pumping Lemma's. An Annotated Bibliography of Pumping. *Bull. EATCS* **17** (1982), 34–53.

THEORIE-TAG 2023
A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 1–4.

RPTU

Checking Directedness of Regular and Context-free Languages

Moses Ganardi[(A)] Irmak Sağlam*[(A)] Georg Zetzsche[(A)]

[(A)]Max Planck Institute for Software Systems (MPI-SWS), Germany
{mganardi,isaglam,georg}@mpi-sws.org

1. Abstract

In this work we focus on the following problem: Given a downward closed language L, what is the complexity of deciding L's upward directedness? We study the problem on regular L and context-free L given via their grammars. We show that if L is a regular language on a fixed alphabet (alphabet is a part of the input), the problem is in NL, whereas if L is on an arbitrary alphabet it is in AC^1. On the other hand, if L is given as a CFG, we show the problem to be PSPACE-complete.

2. Preliminaries

Subword ordering. Let w_1 and w_2 be two finite words on the finite alphabet Σ. Then w_1 is said to be a *subword* of w_2, if we can get w_1 by deleting some letters of w_2.
Downward closedness and upward directedness A language L over the finite alphabet Σ is called *downward closed*, if for every word $w \in L$, all subwords of w are also in L. Similarly, L is called *upward directed* (or for short, *directed*) if for any two words $w_1, w_2 \in L$ there exists a word $w_3 \in L$ such that both w_1 and w_2 are subwords of w_3.
Ideals. Ideals I over Σ are downward closed and directed subsets of Σ^*, and the languages they accept can be represented as a concatenation of languages accepted by *atoms*, as follows:

$$I = L(A_1 A_2 \dots A_n)$$

$A_1, \dots, A_n$ are called *atoms* over Σ and are languages either of shape $\{a, \epsilon\}$ for some $a \in \Sigma$ or of shape Δ^* for some $\Delta \subseteq \Sigma^*$. $A_1 A_2 \dots A_n$ is called a representation of ideal I.

Clearly, one ideal can have different representations, e.g. $I = \{a, \epsilon\}\{a\}^* = \{a\}^*$.

It is known that all downward closed languages over Σ can be written as a finite union of their ideals. That is, for a downward language L, there exists a finite set of ideals $\mathcal{I}$ such that

$$L = \bigcup_{I \in \mathcal{I}} I \tag{1}$$

$\mathcal{I}$ is called a *ideal decomposition* of L.

Reduced ideals. We call an ideal representation $A_1A_2\dots A_n$ *reduced* if for all $i \in [1, n-1]$, neither the language of A_i contains the language of A_{i+1}, or vice versa.

We show that all ideals have a reduced representation. Therefore, a downward closed language can also be written as a finite union of their reduced ideals, as in (1).

Weight function. Next, we define a function μ_k that assigns a weight to each ideal representation, and call μ_k the *weight function*.

Formally, $\mu_k(A_1 \dots A_n) = \sum_{i=1}^{n} \mu_k(A_i)$ where

$$\mu_k(A_i) = \begin{cases} 1, & \text{if } A_i = \{a, \epsilon\} \text{ for some } a \in \Sigma, \\ (k+1)^{|\Delta|}, & \text{if } A_i = \Delta^* \text{ for some } \Delta \subseteq \Sigma \end{cases}$$

We show that for two reduced ideal representations $A_1 \dots A_n$ and $B_1 \dots B_m$ representing ideals I and J, respectively,

$$\text{if } I \subseteq J, \text{ then } \mu_k(A_1 \dots A_n) \leq \mu_k(B_1 \dots B_m) \text{ for any } k \geq \max(n, m) \tag{2}$$

Moreover, if $I \subseteq J$, then the inequality is strict.

3. Main approach

Our main approach to tackling the problem, both for regular and context-free L is to efficiently manipulate the finite abstraction the language is given by, to obtain a similar model that accepts a reduced ideal decomposition of L. That is, in the case of regular L we will get a DFA that accepts a finite language of reduced ideals of L; and in the case of context-free L we will get a CFG that accepts the same. In both of these models, each accepted word of the model gives one reduced ideal in the decomposition of L. The union of all these ideals gives L. Then, we efficiently check the weights of all accepted words in the respective model, and obtain an ideal with the maximum weight $I_{\max}$ (according to μ_k where k is the length of the longest accepted word).

Since L is downward closed, checking its directedness correspond to checking whether L is an ideal itself. Due to the inclusion result we have on the weight function (2), if L is an ideal itself, then it should be contained by $I_{\max}$ (Note that $I_{\max} \subseteq L$ trivially holds). Then the result of this inclusion check, gives us the answer to the directedness of L.

To efficiently transform the abstractions that accepts L into models that accept the reduced ideal decompositions of L, we use transducers. We show that this translation can be achieved in NL for regular languages and P for context-free languages.

Then we show that the maximum weighted-path in the model can be obtained in NL for regular languages with fixed alphabets, AC^1 for regular languages with arbitrary alphabets and in P for context-free languages. In the regular case, we calculate the maximum-weighted path in the DFA my using max-plus semiring; and in the context-free case we use dynamic programming to obtain the path of the CFG that has the maximum weight.

Then we use the existing result [1] to check for the inclusion $L \subseteq I_{\max}$ for regular languages in NL, and we show the inclusion can be checked in PSPACE for context-free languages by simply guessing an ideal representation of $I \in \mathcal{I}$ that does not embed in $I_{\max}$ and checking the whether I embeds in $I_{\max}$ atom by atom.

Lastly, we give a matching lowerbound for the context-free case. In particular, we reduce a known PSPACE-hard membership problem in straight-line programs to the problem of checking whether a language given by a CFG is contained in an ideal.

Literatur

[1] G. ZETZSCHE, The Complexity of Downward Closure Comparisons. In: I. CHATZIGIANNAKIS, M. MITZENMACHER, Y. RABANI, D. SANGIORGI (eds.), *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. LIPIcs 55, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 123:1–123:14.
https://doi.org/10.4230/LIPIcs.ICALP.2016.123

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 57–57.

RPTU

Longest Common Subsequence with Gap Constraints

Duncan Adamson[(A)] Maria Kosche[(A)] Tore Koß[(A)]
Florin Manea[(A)] Stefan Siemer[(A)]

[(A)]Department of Computer Science, University of Göttingen, Göttingen, Germany
stefan.siemer@cs.uni-goettingen.de

Abstract

We consider the longest common subsequence problem in the context of subsequences with gap constraints. In particular, following Day et al. 2022 [2], we consider the setting when the distance (i. e., the gap) between two consecutive symbols of the subsequence has to be between a lower and an upper bound (which may depend on the position of those symbols in the subsequence or on the symbols bordering the gap) as well as the case where the entire subsequence is found in a bounded range (defined by a single upper bound), considered by Kosche et al. 2022 [3]. In all these cases, we present efficient algorithms for determining the length of the longest common constrained subsequence between two given strings.

The paper on which this talk is based appeared in WORDS 2023 [1].

References

[1] D. ADAMSON, M. KOSCHE, T. KOSS, F. MANEA, S. SIEMER, Longest Common Subsequence with Gap Constraints. In: A. E. FRID, R. MERCAS (eds.), *Combinatorics on Words - 14th International Conference, WORDS 2023, Umeå, Sweden, June 12-16, 2023, Proceedings*. Lecture Notes in Computer Science 13899, Springer, 2023, 60–76.
https://doi.org/10.1007/978-3-031-33180-0_5

[2] J. D. DAY, M. KOSCHE, F. MANEA, M. L. SCHMID, Subsequences with Gap Constraints: Complexity Bounds for Matching and Analysis Problems. In: S. W. BAE, H. PARK (eds.), *33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea*. LIPIcs 248, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 64:1–64:18.
https://doi.org/10.4230/LIPIcs.ISAAC.2022.64

[3] M. KOSCHE, T. KOSS, F. MANEA, V. PAK, Subsequences in Bounded Ranges: Matching and Analysis Problems. In: A. W. LIN, G. ZETZSCHE, I. POTAPOV (eds.), *Reachability Problems - 16th International Conference, RP 2022, Kaiserslautern, Germany, October 17-21, 2022, Proceedings*. Lecture Notes in Computer Science 13608, Springer, 2022, 140–159.
https://doi.org/10.1007/978-3-031-19135-0_10

THEORIE-AG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 58–60.

RPTU

Concurrent Stochastic Lossy Channel Games

Daniel Stan[(A)]

[(A)]EPITA, Paris-Strasbourg
daniel.stan@epita.fr

Lossy channel systems is a classical model of infinite state space systems used for representing processes communicating through unbounded FIFO channels[2, 7, 13, 6, 1]. In this formalism, we assume the channels to be unreliable, that is to say, there is always a small fixed probability for a message to be dropped before being read. This assumption is crucial to regain decidability for simple problems such as reachability checking of a configuration/state, through the use of well-quasi-orderings [9, 12] and backward reachability techniques [3]. After a recall of these methods, we introduce an extension where two or more players control the transitions and operations on the channels, concurrently. This concurrent settings enables the players to play mixed strategies, which are more likely to interesting equilibrium concepts [14, 11, 10].

In this talk, we focus on the 2.5-player zero-sum case, and first revisit algorithms for solving such games with qualitative probabilistic objectives. For reachability and safety, we extend algorithms known in the finite case [8], to the infinite state space case. As opposed to previous work on turn-based games [4, 5], the presence of concurrent actions requires a more careful analysis of winning strategies, whose suitable classes are depicted in Figure 1. We further discuss the computation of winning regions for more complex objectives such as almost-sure Büchi/co-Büchi objectives, as well as conjunctions of qualitative objectives. These results are finally compared to the non-deterministic setting –that is to say "exists/for all" winning modes– which are surprisingly harder, or even undecidable [13].

Joint work with Parosh Aziz Abdullah, Anthony W. Lin and Muhammad Najib, submitted to CSL'2024.

Irgendein Projekt kann hier eingefügt werden.

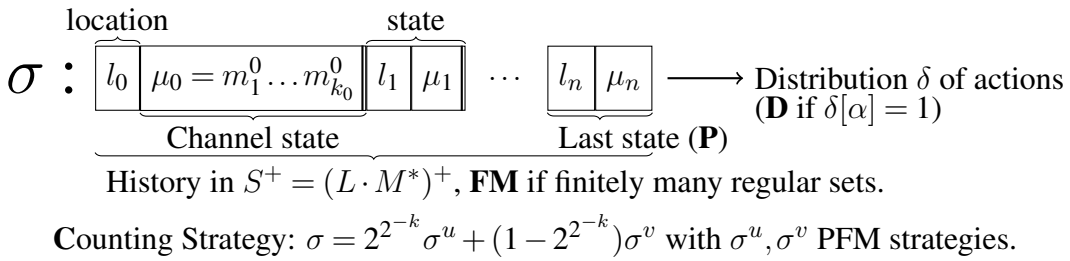


Figure 1: Summary of useful strategy classes.

References

[1] P. A. ABDULLA, C. AISWARYA, M. F. ATIG, Data Communicating Processes with Unreliable Channels. In: M. GROHE, E. KOSKINEN, N. SHANKAR (eds.), *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*. ACM, 2016, 166–175.
http://doi.acm.org/10.1145/2933575.2934535

[2] P. A. ABDULLA, N. BERTRAND, A. M. RABINOVICH, P. SCHNOEBELEN, Verification of probabilistic systems with faulty communication. *Inf. Comput.* **202** (2005) 2, 141–165.
https://doi.org/10.1016/j.ic.2005.05.008

[3] P. A. ABDULLA, K. CERANS, B. JONSSON, Y. TSAY, General Decidability Theorems for Infinite-State Systems. In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, 313–321.
https://doi.org/10.1109/LICS.1996.561359

[4] P. A. ABDULLA, L. CLEMENTE, R. MAYR, S. SANDBERG, Stochastic Parity Games on Lossy Channel Systems. *Log. Methods Comput. Sci.* **10** (2014) 4.
https://doi.org/10.2168/LMCS-10(4:21)2014

[5] P. A. ABDULLA, N. B. HENDA, L. DE ALFARO, R. MAYR, S. SANDBERG, Stochastic Games with Lossy Channels. In: R. M. AMADIO (ed.), *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 - April 6, 2008. Proceedings*. Lecture Notes in Computer Science 4962, Springer, 2008, 35–49.
https://doi.org/10.1007/978-3-540-78499-9_4

[6] P. A. ABDULLA, B. JONSSON, Verifying Programs with Unreliable Channels. In: *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93), Montreal, Canada, June 19-23, 1993*. IEEE Computer Society, 1993, 160–170.
https://doi.org/10.1109/LICS.1993.287591

[7] N. BERTRAND, P. SCHNOEBELEN, Model Checking Lossy Channels Systems Is Probably Decidable. In: A. D. GORDON (ed.), *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*. Lecture Notes in Computer Science 2620, Springer, 2003, 120–135.
https://doi.org/10.1007/3-540-36576-1_8

[8] L. DE ALFARO, T. A. HENZINGER, O. KUPFERMAN, Concurrent reachability games. *Theor. Comput. Sci.* **386** (2007) 3, 188–217.
https://doi.org/10.1016/j.tcs.2007.07.008

[9] A. FINKEL, P. SCHNOEBELEN, Well-structured transition systems everywhere! *Theor. Comput. Sci.* **256** (2001) 1-2, 63–92.
https://doi.org/10.1016/S0304-3975(00)00102-X

[10] J. GUTIERREZ, S. KOWARA, S. KRAUS, T. STEEPLES, M. WOOLDRIDGE, Cooperative concurrent games. *Artificial Intelligence* **314** (2023), 103806.

[11] J. GUTIERREZ, M. NAJIB, G. PERELLI, M. J. WOOLDRIDGE, Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. *Artif. Intell.* **287** (2020), 103353. https://doi.org/10.1016/j.artint.2020.103353

[12] G. HIGMAN, Ordering by Divisibility in Abstract Algebras. In: *Proc. London Math. Soc.*. 2, 1952, 326–336.

[13] R. MAYR, Undecidable problems in unreliable computations. *Theor. Comput. Sci.* **297** (2003) 1-3, 337–354. https://doi.org/10.1016/S0304-3975(02)00646-1

[14] J. F. NASH, Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America* **36** (1950) 1, 48–49.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 61–66.

RPTU

Strictly Locally Testable and Resources Restricted Control Languages in Tree-Controlled Grammars

Bianca Truthe

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

Abstract

Tree-controlled grammars are context-free grammars where the derivation process is controlled in such a way that every word on a level of the derivation tree must belong to a certain control language. We investigate the generative capacity of such tree-controlled grammars where the control languages are special regular sets, especially strictly locally testable languages or languages restricted by resources of the generation (number of nonterminal symbols or production rules) or acceptance (number of states). Furthermore, the set theoretic inclusion relations of these subregular language families themselves are studied.

1. Introduction

In the monograph [1] by Jürgen Dassow and Gheorghe Păun, *Seven Circumstances Where Context-Free Grammars Are Not Enough* are presented. A possibility to enlarge the generative power of context-free grammars is to introduce some regulation mechanism which controls the derivation in a context-free grammar. In some cases, regular languages are used for such a regulation. They are rather easy to handle and, used as control, they often lead to context-sensitive or even recursively enumerable languages while the core grammar is only context-free.

One such control mechanism was introduced by Karel Čulik II and Hermann A. Maurer in [13] where the structure of derivation trees of context-free grammars is restricted by the requirement that the words of all levels of the derivation tree must belong to a given regular (control) language. This model is called tree-controlled grammar.

Gheorghe Păun proved that the generative capacity of such grammars coincides with that of context-sensitive grammars (if no erasing rules are used) or arbitrary phrase structure grammars (if erasing rules are used). Thus, the question arose to what extend the restrictions can be weakened in order to obtain 'useful' families of languages which are located somewhere between the classes of context-free and context-sensitive languages.

This is an extended abstract of a paper presented at AFL 2023 ([10]).

In [2, 3, 4, 5, 9, 11, 12], many subregular families of languages have been investigated as classes for the control languages. In this paper, we continue this research with further subregular language families, especially strictly locally testable languages or languages restricted by resources of the generation (number of non-terminal symbols or production rules) or acceptance (number of states). Furthermore, the set theoretic inclusion relations of these subregular language families themselves are studied.

2. Preliminaries

By *REG*, *CF* and *CS*, we denote the set of all regular, all context-free, and all context-sensitive languages, respectively. With a derivation of a terminal word by a context-free grammar, we associate a derivation tree which has the start symbol in its root and where every node with a non-terminal $A \in N$ has as children nodes with symbols which form, read from left to right, a word w such that $A \to w$ is a rule of the grammar (if $A \to \lambda$, then the node with A has only one child node and this is labelled with λ). Nodes with terminal symbols or λ have no children. With any derivation tree t of height k and any number $0 \leq j \leq k$, we associate the word of level j and the sentential form of level j which are given by all nodes of depth j read from left to right and all nodes of depth j and all leaves of depth less than j read from left to right, respectively. Obviously, if two words w and v are sentential forms of two successive levels, then $w \Longrightarrow^* v$ holds and this derivation is obtained by a parallel replacement of all non-terminal symbols occurring in the word w.

By *MAT*, we denote the family of all languages generated by matrix grammars with appearance checking and without erasing rules; by $\mathit{MAT}_{\mathit{fin}}$, we denote the family of all such languages where the matrix grammar is of finite index ([1], [8]). By *E0L* (*ET0L*), we denote the family of all languages generated by extended (tabled) interactionless Lindenmayer systems ([7]).

By restricting the resources needed for generating or accepting their elements, we define the following families:

$$\begin{aligned}
\mathit{RL}_n^V &= \{\, L \mid L \text{ is gen. by a right-lin. grammar with at most } n \text{ non-terminal symbols} \,\}, \\
\mathit{RL}_n^P &= \{\, L \mid L \text{ is gen. by a right-lin. grammar with at most } n \text{ production rules} \,\}, \\
\mathit{REG}_n^Z &= \{\, L \mid L \text{ is acc. by a DFA with at most } n \text{ states} \,\}.
\end{aligned}$$

Furthermore, we consider the following restrictions for regular languages. Let L be a language over an alphabet V. With respect to the alphabet V, the language L is said to be

- *monoidal* if and only if $L = V^*$,
- *nilpotent* if and only if it is finite or its complement $V^* \setminus L$ is finite,
- *combinational* if and only if it has the form $L = V^*X$ for some subset $X \subseteq V$,
- *definite* if and only if it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *suffix-closed* (or *fully initial* or *multiple-entry* language) if and only if, for any two words $x \in V^*$ and $y \in V^*$, the relation $xy \in L$ implies the relation $y \in L$,
- *ordered* if and only if the language is accepted by some deterministic finite automaton $A = (V, Z, z_0, F, \delta)$ with an input alphabet V, a finite set Z of states, a start state $z_0 \in Z$, a set $F \subseteq Z$ of accepting states and a transition mapping δ where $(Z, \preceq)$

is a totally ordered set and, for any input symbol $a \in V$, the relation $z \preceq z'$ implies $\delta(z,a) \preceq \delta(z',a)$,

- *commutative* if and only if it contains with each word also all permutations of this word,
- *circular* if and only if it contains with each word also all circular shifts of this word,
- *non-counting* (or *star-free*) if and only if there is a natural number $k \geq 1$ such that, for every three words $x \in V^*$, $y \in V^*$, and $z \in V^*$, it holds $xy^k z \in L$ if and only if $xy^{k+1}z \in L$,
- *power-separating* if and only if, there is a natural number $m \geq 1$ such that for every word $x \in V^*$, either $J_x^m \cap L = \emptyset$ or $J_x^m \subseteq L$ where $J_x^m = \{\, x^n \mid n \geq m \,\}$,
- *union-free* if and only if L can be described by a regular expression which is only built by product and star,
- *strictly locally k-testable* if and only if there are three subsets B, I, and E of V^k such that any word $a_1 a_2 \ldots a_n$ with $n \geq k$ and $a_i \in V$ for $1 \leq i \leq n$ belongs to the language L if and only if $a_1 a_2 \ldots a_k \in B$, $a_{j+1} a_{j+2} \ldots a_{j+k} \in I$ for every j with $1 \leq j \leq n-k-1$, and $a_{n-k+1} a_{n-k+2} \ldots a_n \in E$,
- *strictly locally testable* if and only if it is strictly locally k-testable for some natural number k.

We remark that monoidal, nilpotent, combinational, definite, ordered, union-free, and strictly locally (k-)testable languages are regular, whereas non-regular languages of the other types mentioned above exist. Here, we consider among the commutative, circular, suffix-closed, non-counting, and power-separating languages only those which are also regular.

By *FIN*, *MON*, *NIL*, *COMB*, *DEF*, *SUF*, *ORD*, *COMM*, *CIRC*, *NC*, *PS*, *UF*, SLT_k (for any natural number $k \geq 1$), and *SLT*, we denote the families of all finite, monoidal, nilpotent, combinational, definite, regular suffix-closed, ordered, regular commutative, regular circular, regular non-counting, regular power-separating, union-free, strictly locally k-testable, and strictly locally testable languages, respectively.

For any natural number $n \geq 1$, let MON_n be the set of all languages that can be represented in the form $A_1^* \cup A_2^* \cup \cdots \cup A_k^*$ with $1 \leq k \leq n$ where all A_i ($1 \leq i \leq k$) are alphabets. Obviously,

$$\mathit{MON} = \mathit{MON}_1 \subset \mathit{MON}_2 \subset \cdots \subset \mathit{MON}_j \subset \cdots .$$

A strictly locally testable language characterized by three finite sets B, I, and E as above which includes additionally a finite set F of words which are shorter than those of the sets B, I, and E is denoted by $[B,I,E,F]$.

As the set of all families under consideration, we set

$$\begin{aligned}\mathfrak{F} = {} & \{\mathit{FIN}, \mathit{NIL}, \mathit{COMB}, \mathit{DEF}, \mathit{SUF}, \mathit{ORD}, \mathit{COMM}, \mathit{CIRC}, \mathit{NC}, \mathit{PS}, \mathit{UF}\} \\ & \cup \{\, \mathit{MON}_k \mid k \geq 1 \,\} \cup \{\mathit{SLT}\} \cup \{\, \mathit{SLT}_k \mid k \geq 1 \,\} \\ & \cup \{\, \mathit{RL}_n^V \mid n \geq 1 \,\} \cup \{\, \mathit{RL}_n^P \mid n \geq 1 \,\} \cup \{\, \mathit{REG}_n^Z \mid n \geq 1 \,\}.\end{aligned}$$

A tree-controlled grammar is a quintuple $G = (N,T,P,S,R)$ where

- (N,T,P,S) is a context-free grammar with a set N of non-terminal symbols, a set T of terminal symbols, a set P of context-free non-erasing rules (with the only exception that the rule $S \to \lambda$ is allowed if S does not occur on a right-hand side of a rule), and an axiom S,
- R is a regular set over $N \cup T$.

The language $L(G)$ generated by a tree-controlled grammar $G = (N, T, P, S, R)$ consists of all such words $z \in T^*$ which have a derivation tree t where z is the word obtained by reading the leaves from left to right and the words of all levels of t – besides the last one – belong to the regular control language R.

Let $\mathcal{F}$ be a subfamily of *REG*. Then, we denote the family of languages generated by tree-controlled grammars $G = (N, T, P, S, R)$ with $R \in \mathcal{F}$ by $\mathcal{TC}(\mathcal{F})$.

In [6] (see also [1]), it has been shown that a language L is generated by a tree-controlled grammar if and only if it is generated by a context-sensitive grammar. In subsequent papers, tree-controlled grammars have been investigated where the control language belongs to some subfamily of the class *REG* ([2, 3, 4, 5, 9, 11, 12]). In this paper, we continue this research with further subregular language families. From the definition follows that the subset relation is preserved under the use of tree-controlled grammars: if we allow more, we do not obtain less.

3. Results

A summary of all the inclusion relations obtained so far is given in Figure 1. An arrow from an entry X to an entry Y depicts the inclusion $X \subseteq Y$; a solid arrow means proper inclusion; a dashed arrow indicates that it is not known whether the inclusion is proper. If two families are not connected by a directed path, then they are not necessarily incomparable. An edge label in this figure refers to the paper where the respective inclusion is proved.

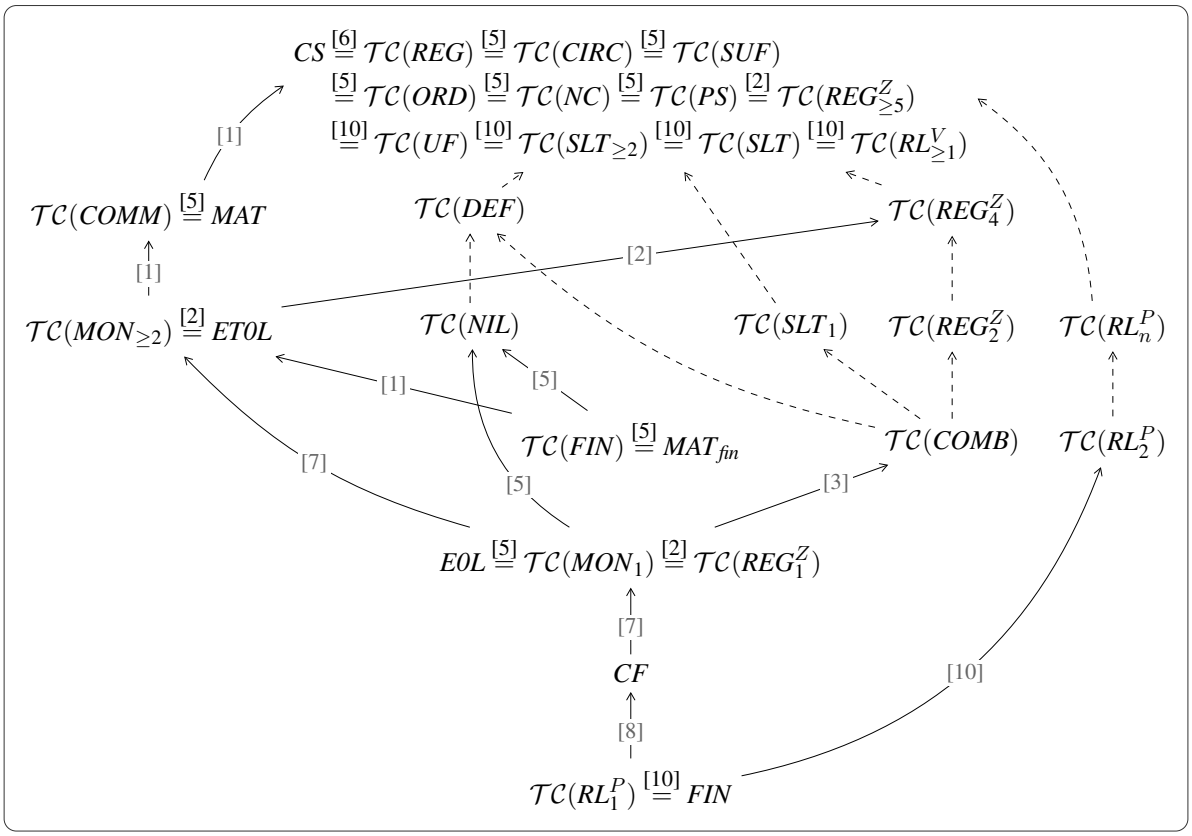


Figure 1: Hierarchy of subregularly tree-controlled language families

4. Future Research

There are several families of languages generated by tree-controlled grammars where we do not have a characterization by some other language class. The strictness of some inclusions and the incomparability of some families remain as open problems.

In the present paper, we have only considered tree-controlled grammars without erasing rules. For tree-controlled grammars where erasing rules are allowed, several results have been published already (see, e. g., [3, 11, 12]). Also in this situation, there are some open problems.

Another direction for future research is to consider other subregular language families or to relate the families of languages generated by tree-controlled grammars to language families obtained by other grammars/systems with regulated rewriting.

References

[1] J. DASSOW, GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. EATCS Monographs in Theoretical Computer Science 18, Springer-Verlag, 1989.

[2] J. DASSOW, R. STIEBE, B. TRUTHE, Two collapsing hierarchies of subregularly tree controlled languages. *Theoretical Computer Science* **410** (2009) 35, 3261–3271.

[3] J. DASSOW, R. STIEBE, B. TRUTHE, Generative capacity of subregularly tree controlled grammars. *International Journal of Foundations of Computer Science* **21** (2010) 5, 723–740.

[4] J. DASSOW, B. TRUTHE, On two hierarchies of subregularly tree controlled languages. In: C. CÂMPEANU, G. PIGHIZZINI (eds.), *Descriptional Complexity of Formal Systems, 10th International Workshop, Charlottetown, Prince Edward Island, Canada, July 16–18, 2008, Proceedings*. University of Prince Edward Island, 2008, 145–156.

[5] J. DASSOW, B. TRUTHE, Subregularly tree controlled grammars and languages. In: E. CSUHAJ-VARJÚ, Z. ÉSIK (eds.), *Automata and Formal Languages, 12th International Conference, AFL 2008, Balatonfüred, Hungary, May 27–30, 2008, Proceedings*. Computer and Automation Research Institute, Hungarian Academy of Sciences, 2008, 158–169.

[6] GH. PĂUN, On the generative capacity of tree controlled grammars. *Computing* **21** (1979), 213–220.

[7] G. ROZENBERG, A. SALOMAA, *The Mathematical Theory of L Systems*. Academic Press, 1980.

[8] G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[9] R. STIEBE, On the complexity of the control language in tree controlled grammars. In: J. DASSOW, B. TRUTHE (eds.), *Colloquium on the Occasion of the 50th Birthday of Victor Mitrana, Otto-von-Guericke-Universitäat Magdeburg, Germany, June 27, 2008, Proceedings*. Otto-von-Guericke-Universitäat Magdeburg, Germany, 2008, 29–36.

[10] B. TRUTHE, Strictly Locally Testable and Resources Restricted Control Languages in Tree-Controlled Grammars. In: ZS. GAZDAG, SZ. IVÁN, G. KOVÁSZNAI (eds.), *16th International Conference on Automata and Formal Languages, AFL 2023, Eger, Hungary, September 5–7, 2023, Proceedings*. EPTCS 386, Open Publishing Association, 2023, 253–268.

[11] S. TURAEV, J. DASSOW, F. MANEA, M. H. SELAMAT, Language classes generated by tree controlled grammars with bounded nonterminal complexity. *Theoretical Computer Science* **449** (2012), 134–144.

[12] GY. VASZIL, On the nonterminal complexity of tree controlled grammars. In: H. BORDIHN, M. KUTRIB, B. TRUTHE (eds.), *Languages Alive – Essays Dedicated to Jürgen Dassow on the Occasion of His 65th Birthday*. LNCS 7300, Springer, 2012, 265–272.

[13] K. ČULIK II, H. A. MAURER, Tree controlled grammars. *Computing* **19** (1977) 2, 129–139.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 67–70.

RPTU

Matching Patterns with Variables Under Simon's Congruence

Pamela Fleischmann[(A)] Sungmin Kim[(C)] Tore Koß[(B)]
Florin Manea[(B)] Dirk Nowotka[(A)] Stefan Siemer[(B)]
Max Wiedenhöft[(A)]

[(A)]Department of Computer Science, Kiel University, Germany
{fpa,dn,maw}@informatik.uni-kiel.de

[(B)]Department of Computer Science, University of Göttingen, Germany
{tore.koss,florin.manea,stefan.siemer}@cs.uni-goettingen.de

[(C)]Department of Computer Science, Yonsei University, Republic of Korea
rena_rio@yonsei.ac.kr

Abstract

We introduce and investigate a series of matching problems for patterns with variables under Simon's congruence and give a thorough picture of their computational complexity.

1. Introduction

A *pattern with variables* is a string $\alpha \in (\Sigma \cup \mathcal{X})^*$ consisting of *constant letters* (or *terminals*) from a finite alphabet $\Sigma = \{1, ..., \sigma\}$ of size $\sigma \geq 2$ and a potentially infinite set of *variables* $\mathcal{X}$ such that $\Sigma \cap \mathcal{X} = \emptyset$. Here, we assume σ to be bounded by a constant. A pattern is mapped by a *substitution* $h : (\Sigma \cup \mathcal{X})^* \to \Sigma^*$ which is a morphism that acts as the identity on Σ and maps each variable of $\mathcal{X}$ to a (potentially empty) string over Σ. For example, we can map the pattern $\alpha = xx\mathtt{abab}yy$ to the string of constants aaaaababbb by the substitution h with $h(x) = \mathtt{aa}$ and $h(y) = \mathtt{b}$ and by that $h(\alpha) = \mathtt{aaaaababbb}$. If a pattern α can be mapped to a string of constants w, we say that α *matches* w. The problem of deciding whether there exists a substitution h for a pattern α such that $h(\alpha) = w$ for a given word w is called the *(exact) matching problem*, Match. This heavily studied problem is NP-Complete in general [1], but a series of classes of patterns, defined by structural restrictions, for which Match is in P were identified [4]. Moreover, for most of the parameterised classes, Match is $W[1]$-hard [3] w.r.t. the structural parameters used to define the respective classes. Recently, Gawrychowski et. al. [7, 8] studied Match in an approximate setting. In general: given a pattern α and a word w, decide whether there exists a substitution h such that $h(\alpha)$ is similar to w w.r.t. some similarity measure. Thus, it seems natural to consider other string-equivalence relations as similarity measures. Here, we consider an approximate variant of Match using Simon's congruence $\sim_k$ [13].

Matching under Simon's Congruence: $\texttt{MatchSimon}(\alpha, w, k)$
Input: Pattern α, $|\alpha| = m$, word w, $|w| = n$, and number $k \in [n]$.
Question: Is there a substitution h with $h(\alpha) \sim_k w$?

A string u is a *subsequence* of a string w if u results from w by deleting some letters of w. Let $\mathbb{S}_k(w)$ be the set of all subsequences of a given string w up to length $k \in \mathbb{N}_0$. Two strings v and v' are k-Simon congruent iff $\mathbb{S}_k(v) = \mathbb{S}_k(v')$ [13]. Then, we write $v \sim_k v'$. As a similarity measure for strings, $\sim_k$ was optimally solved in [2, 6]. Thus, it seems natural to consider, in a general setting, the problem of checking whether one can map a given pattern α to a string which is similar to w w.r.t. $\sim_k$. One of the congruence-classes of Σ^* w.r.t. $\sim_k$ received much attention: the class of k-subsequene universal words [11, 2] which are those words which contain all k-length words as subsequences. Here, we consider the following problem, where $\iota(w)$ (universality index of w) is the largest integer ℓ for which w is ℓ-subsequence universal.

Matching a Target Universality: $\texttt{MatchUniv}(\alpha, k)$
Input: Pattern α, $|\alpha| = m$, and $k \in \mathbb{N}_0$.
Question: Is there a substitution h with $\iota(h(\alpha)) = k$?

Note that `MatchUniv` can be formulated in terms of `MatchSimon`. One very important difference, though, is that we are not explicitly given a target word w but instead, we are given the number k which represents the target more compactly (using only $\log k$ bits).

A well-studied extension of `Match` is the satifiability problem for word equations (e.g. see [10]). Here, we extend `MatchSimon` to the problem of solving word equations under $\sim_k$:

Word Equations under Simon's Congruence: $\texttt{WESimon}(\alpha, \beta, k)$
Input: Patterns α, β, $|\alpha| = m$, $|\beta| = n$, and $k \in [m+n]$.
Question: Is there a substitution h with $h(\alpha) \sim_k h(\beta)$?

We present a rather comprehensive picture of the problems' computational complexity, starting with `MatchUniv` and showing that it is `NP`-complete. Also, we present a series of structurally restricted classes of patterns for which it can be solved in polynomial time. Then, we discuss `MatchSimon` and show its `NP`-completeness. Finally, we discuss `WESimon` and its variants, characterise their computational complexity, and point to a series of future research directions.

2. The NP-Completeness of `MatchUniv` and `MatchSimon`

To show that `MatchUniv` is `NP`-hard, we reduce the `NP`-complete problem `3CNFSAT` (see [9, 5]) to `MatchUniv`. The idea is to construct several gadgets which allow us to encode a `3CNFSAT`-instance φ as a `MatchUniv` instance (α, k). Thus, we can find a substitution h for the instance (α, k) such that $\iota(h(\alpha)) = k$ iff φ is satisfiable. We recall `3CNFSAT`.

3-Satisfiability for formulas in conjunctive normal form, `3CNFSAT`.
Input: Clauses $\varphi := \{c_1, c_2, \ldots, c_m\}$, where $c_j = (y_j^1 \vee y_j^2 \vee y_j^3)$ for $1 \leq j \leq m$, and y_j^1, y_j^2, y_j^3 from a finite set of boolean variables $X := \{x_1, x_2, \ldots, x_n\}$ and their negations $\bar{X} := \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$.
Question: Is there an assignment for X, which satisfies all clauses of φ?

Further, we get `NP`-containment by using a slight variation of *subsequence universality signatures* [12] such that the maximal length of certificates is polynomial in the input.

Theorem 2.1 `MatchUniv` *is* `NP`*-complete.*

By restricting the input patterns, we get two classes of patterns such that MatchUniv can be solved in polynomial time.

Proposition 2.2 $\texttt{MatchUniv}(\alpha, k) \in \texttt{P}$ *if there exists a variable that occurs only once in* α. *So,* $\texttt{MatchUniv}(\alpha, k) \in \texttt{P}$ *for regular patterns (see e.g. [4])* α. *Also,* $\texttt{MatchUniv}(\alpha.k) \in \texttt{P}$ *if* $|\texttt{var}(\alpha)|$ *is constant.*

Further, we discuss the MatchSimon problem. In case of MatchSimon we are given a pattern α, a word w, and a natural number $k \leq |w|$ and we want to check the existence of a substitution h such that $h(\alpha) \sim_k w$. We immediately get that MatchSimon is NP-hard, because $\texttt{MatchSimon}(\alpha, w, |w|)$ is equivalent to $\texttt{Match}(\alpha, w)$ and Match is NP-complete. Notice that this result followed much easier than the corresponding lower bound for MatchUniv because in MatchSimon we only ask for $h(\alpha) \sim_k w$ and allow $h(\alpha) \sim_{k+1} w$, while in MatchUniv $h(\alpha)$ has to be strict k-universal but not $(k+1)$-universal. Thus, we consider the following problem.

Matching under Strict Simon's Congruence: $\texttt{MatchStrictSimon}(\alpha, w, k)$	
Input:	Pattern α, $\lvert\alpha\rvert = m$, word w, $\lvert w\rvert = n$, and $k \in [n]$.
Question:	Is there a substitution h with $h(\alpha) \sim_k w$ and $h(\alpha) \not\sim_{k+1} w$?

Adapting the reduction used for Theorem 2.1, we can show that MatchStrictSimon is NP-hard. For the NP-containment, we know that it is enough to only consider strings of length up to $O((k+1)^\sigma)$ as potential substitutions of the variables in a substitution h for a pattern α. Longer strings can be replaced with shorter ones which are $\sim_k$-congruent with the same impact on the sets $\mathbb{S}_k(h(\alpha))$.

Theorem 2.3 MatchSimon *and* MatchStrictSimon *are* NP*-complete.*

If the patterns are regular, note that MatchSimon and MatchStrictSimon are in P.

Proposition 2.4 $\texttt{MatchSimon}(\alpha, w, k), \texttt{MatchStrictSimon}(\alpha, w, k) \in \texttt{P}$ *if* α *is regular.*

3. An Analysis of WESimon

Finally, we address the WESimon problem, where we are given two patterns α and β and a natural number k and we want to check the existence of a substitution h with $h(\alpha) \sim_k h(\beta)$.

Theorem 3.1 WESimon *is* NP*-complete.*

To avoid trivial cases arising for WESimon, we also consider a stricter variant of this problem which, in contrast to WESimon, is NP-hard in all cases.

Word Equations under Strict Simon's Congruence: $\texttt{WEStrictSimon}(\alpha, \beta, k)$	
Input:	Patterns α, β, $\lvert\alpha\rvert = m$, $\beta = n$, and $k \in [m+n]$.
Question:	Is there a substitution h with $h(\alpha) \sim_k h(\beta)$ and $h(\alpha) \not\sim_{k+1} h(\beta)$?

Lemma 3.2 WEStrictSimon *is* NP*-hard, even if both patterns contain variables.*

Regarding the NP-membership, if k is upper bounded by a polynomial function in $|\alpha| + |\beta|$, we get that $\texttt{WEStrictSimon} \in \texttt{NP}$. Otherwise, the question of the NP-membership remains open.

Theorem 3.3 WEStrictSimon *is* NP*-complete for all* $k \leq |\alpha| + |\beta|$.

4. Conclusion

We considered the problem of matching patterns with variables under Simon's congruence. Specifically, we considered the three main problems MatchUniv, MatchSimon, WESimon, strict variations MatchStrictSimon and WEStrictSimon, and have given a comprehensive image of their computaitonal complexity. In general, these problems are NP-complete, but have interesting particular cases which are in P. Interestingly, our NP and P algorithms work in (nondeterministic) polynomial time only in the case of a constant input alphabet. A characterisation of the parameterised complexity of these problems w.r.t. the parameter σ might be interesting. Another paramter of interest could be the number of variables of the considered patterns. We conjecture that the problems are $W[1]$-hard with respect to both of these parameters.

References

[1] D. ANGLUIN, Finding Patterns Common to a Set of Strings. *J. Comput. Syst. Sci.* **21** (1980) 1, 46–62.

[2] L. BARKER, P. FLEISCHMANN, K. HARWARDT, F. MANEA, D. NOWOTKA, Scattered Factor-Universality of Words. In: *DLT 2020, Proceedings*. LNCS 12086, Springer, 2020, 14–28.

[3] R. G. DOWNEY, M. R. FELLOWS, *Parameterized Complexity*. Monographs in Computer Science, Springer, 1999.

[4] H. FERNAU, F. MANEA, R. MERCAS, M. L. SCHMID, Pattern Matching with Variables: Efficient Algorithms and Complexity Results. *ACM Trans. Comput. Theory* **12** (2020) 1, 6:1–6:37.

[5] M. R. GAREY, D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[6] P. GAWRYCHOWSKI, M. KOSCHE, T. KOSS, F. MANEA, S. SIEMER, Efficiently Testing Simon's Congruence. In: *STACS 2021*. LIPIcs 187, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 34:1–34:18.

[7] P. GAWRYCHOWSKI, F. MANEA, S. SIEMER, Matching Patterns with Variables Under Hamming Distance. In: *46th ISMFCS, MFCS 2021*. LIPIcs 202, 2021, 48:1–48:24.

[8] P. GAWRYCHOWSKI, F. MANEA, S. SIEMER, Matching Patterns with Variables Under Edit Distance, Springer, 2022, 275–289.

[9] R. M. KARP, Reducibility Among Combinatorial Problems. The IBM Research Symposia Series, Plenum Press, New York, 1972, 85–103.

[10] M. LOTHAIRE, *Combinatorics on Words*. Cambridge University Press, 1997.

[11] P. SCHNOEBELEN, P. KARANDIKAR, The height of piecewise-testable languages and the complexity of the logic of subwords. *Logical Methods in Computer Science* **15** (2019).

[12] P. SCHNOEBELEN, J. VERON, On Arch Factorization and Subword Universality for Words and Compressed Words. In: *WORDS 2023, Proceedings*. Lecture Notes in Computer Science 13899, 2023, 274–287.

[13] I. SIMON, Piecewise testable events, Springer, 1975, 214–222.

THEORIE-TAG 2023

A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 71–74.

RPTU

α-β-Factorisation and the Binary Case of Simon's Congruence

Pamela Fleischmann[(A)] Jonas Höfer[(B)] Annika Huch[(A)]
Dirk Nowotka[(A)]

[(A)]Kiel University, Kiel, Germany
fpa@informatik.uni-kiel.de.de,stu216885@mail.uni-kiel.de,dn@informatik.uni-kiel.de

[(B)]University of Gothenburg, Sweden
jonas.hofer@gu.se

Abstract

Based on the arch factorisation (Hébrard 1991), first the notion of k-richness and later the one of k-universality - both measure words by their scattered factors - were introduced. In 2022 Fleischmann et al. presented a generalisation by intersecting the arch factorisations of a word and its reverse. Here we use this α-β-factorisation in order to characterise the Simon congruence of k-universal words in terms of 1-universal words and apply these results to binary words obtaining a full characterisation of the index of the congruence.

1. Introduction

A *scattered factor*, *subsequence*, or *scattered subword* of a word w is a word that is obtained by deleting letters from w while preserving the order of the remaining ones, e.g., `tea` and `thora` are both scattered factors of `theorietag`. In contrast to a factor, like `eta`, a scattered factor is not necessarily contiguous. Here, we focus on Simon's congruence [8] $\sim_k$ for $k \in \mathbb{N}_0$: $u \sim_k v$ iff u, v share all scattered factors up to length k. A long outstanding question, posed by Sakarovitch and Simon [7], is the exact structure of the congruence classes of $\sim_k$ and the index of the relation. Currently, no exact formula is known. One approach for studying scattered factors in words is based on the notion of *scattered factor universality* [1, 2, 3]. A word w is called *ℓ-universal* if it contains all words of length ℓ as scattered factors. For instance, the word `alfalfa`[1] is 2-universal since it contains all words of length two over the alphabet $\{\mathtt{a}, \mathtt{l}, \mathtt{f}\}$ as scattered factors. A main tool in this line of research is the *α-β-factorization* [3]. Kosche et al. [6] implicitly used this factorisation to determine shortest absent scattered factors in words.

Our Contribution. We investigate the α-β-factorization and give necessary and sufficient conditions for the congruence of words in terms of their factors. We characterise $\sim_k$ in terms of 1-universal words through their $\alpha\beta\alpha$-factors. We use these results to characterize the classes of binary words and their cardinality, as well as, to calculate the index in this special case. Lastly, we start to transfer the previous results to the ternary alphabet.

[1] Alfalfa (Medicago sativa) is plant whose name means *horse food* in Old Persian

2. Preliminaries

Let $\mathbb{N} = \{1,2,\ldots\}$ and set $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$, $[m] = \{1,\ldots,m\}$, and $[m]_0 = \{0\} \cup [m]$. For the standard definitions of combinatorics on words, we refer to [7]. We abbreviate an alphabet of cardinality $i \in \mathbb{N}$ by Σ_i. If $w = xy$ we write $x^{-1}w$ for y and wy^{-1} for x. A word $u \in \Sigma^*$ of length $n \in \mathbb{N}_0$ is called a *scattered factor* of $w \in \Sigma^*$ if there exist $v_0,\ldots,v_n \in \Sigma^*$ with $w = v_0u[1]v_1\cdots v_{n-1}u[n]v_n$. Let $\mathrm{ScatFact}(w), \mathrm{ScatFact}_k$, and $\mathrm{ScatFact}_{\leq k}$ denote the sets of all, exactly of length k, up to length k resp. scattered factors of w. For comparing words w.r.t. their scattered factors, Simon introduced a congruence relation nowadays known as *Simon's congruence* [8]: two words $u, v \in \Sigma^*$ are called *Simon k-congruent ($u \sim_k v$)* iff $\mathrm{ScatFact}_{\leq k}(u) = \mathrm{ScatFact}_{\leq k}(v)$ for some $k \in \mathbb{N}$. A word $w \in \Sigma^*$ is called *k-universal* w.r.t. Σ if $\mathrm{ScatFact}_k(w) = \Sigma^k$. The maximal k such that w is k-universal is denoted by $\iota(w)$ and called w's *universality index*. For a word $w \in \Sigma^*$ the *arch factorisation* is given by $w = \mathrm{ar}_1(w)\cdots\mathrm{ar}_k(w)\,\mathrm{re}(w)$ for $k \in \mathbb{N}_0$ with $\mathrm{alph}(\mathrm{ar}_i(w)) = \Sigma$ for all $i \in [k]$, the last letter of $\mathrm{ar}_i(w)$ occurs exactly once in $\mathrm{ar}_i(w)$ for all $i \in [k]$, and $\mathrm{alph}(\mathrm{re}(w)) \subset \Sigma$. The words $\mathrm{ar}_i(w)$ are called *arches* and $\mathrm{re}(w)$ is the *rest* of w. Define the *modus of* w as $\mathrm{m}(w) = \mathrm{ar}_1(w)[|\mathrm{ar}_1(w)|]\cdots\mathrm{ar}_k(w)[|\mathrm{ar}_k(w)|] \in \Sigma^k$. Set $\mathrm{ar}_{i..j}(w) = \mathrm{ar}_i(w)\cdots\mathrm{ar}_j(w)$. The α-β-factorisation was introduced in [3] inspired by [6]. Define for the arch factorisation of w^R (read left to right) the i^{th} *reverse arch* $\overleftarrow{\mathrm{ar}}_i(w) = (\mathrm{ar}_{\iota(w)-i+1}(w^R))^R$, the *reverse rest* $\overleftarrow{\mathrm{re}}(w) = (\mathrm{re}(w^R))^R$, and set $\overleftarrow{\mathrm{m}}(w)$ as $\mathrm{m}(w^R)^R$ for the *reverse modus*.

Definition 2.1 *For $w \in \Sigma^*$ define w's* α-β-factorisation *by $w = \alpha_0\beta_1\alpha_1\cdots\alpha_{\iota(w)-1}\beta_{\iota(w)}\alpha_{\iota(w)}$ with $\mathrm{ar}_i(w) = \alpha_{i-1}\beta_i$ and $\overleftarrow{\mathrm{ar}}_i(w) = \beta_i\alpha_i$ for all $i \in [\iota(w)]$, $\overleftarrow{\mathrm{re}}(w) = \alpha_0$, as well as $\mathrm{re}(w) = \alpha_{\iota(w)}$. Define $\mathrm{core}_i = \beta_i[2..|\beta_i|-1]$ or ε if $|\beta_i| \leq 2$.*

Note that the α-β-factorisation is left-right-symmetric and that the i^{th} reverse arch always starts inside the i^{th} arch. We finish this section with three results from [4, 5, 8]

Lemma 2.2 *Let $u, v \in \Sigma^*$, $u', v' \in \Sigma^+$, and $\mathrm{x} \in \Sigma$.*
(1) If $u \sim_k v$ then $w_1uw_2 \sim_{\iota(w_1)+k+\iota(w_2)} w_1vw_2$.
(2) $u'v' \sim_k u'$ iff $u' = u'_1\cdots u'_k$ such that $\mathrm{alph}(u'_1) \supseteq \ldots \supseteq \mathrm{alph}(u'_k) \supseteq \mathrm{alph}(v')$.
(3) $uv \sim_k u\mathrm{x}v$ iff there exist $p, p' \in \mathbb{N}_0$ with $p + p' \geq k$ and $u\mathrm{x} \sim_p u$ and $\mathrm{x}v \sim_{p'} v$.

3. α-β-Factorisation

In this section, we investigate the α-β-factorisation based on results of [4]. The main result states that it suffices to look at 1-universal words in order to gain the information about the $\sim_k$ congruence classes. First, we show that *cutting of ℓ* arches from two k-congruent words each, leads to $(k-\ell)$-congruence. Then we connect the congruence of words to the congruence of their α factors, leading to a characterisation by the congruence of $\alpha\beta\alpha$-factors.

Lemma 3.1 *Let $w, \tilde{w} \in \Sigma^*$ with $w \sim_k \tilde{w}$ and $\iota(w) = \iota(\tilde{w}) < k$, then $\mathrm{ar}_1^{-1}(w)\cdot w \sim_{k-1} \mathrm{ar}_1^{-1}(\tilde{w})\cdot\tilde{w}$ and $\alpha_i\beta_{i+1}\alpha_{i+1}\cdots\alpha_j \sim_{k-\iota(w)+j-i} \tilde{\alpha}_i\tilde{\beta}_{i+1}\tilde{\alpha}_{i+1}\cdots\tilde{\alpha}_j$ for all $0 \leq i \leq j \leq \iota(w)$.*

Proposition 3.2 *For all $w, \tilde{w} \in \Sigma^*$ with $m = \iota(w) = \iota(\tilde{w}) < k$ such that $\beta_i = \tilde{\beta}_i$ for all $i \in [m]$, we have $w \sim_k \tilde{w}$ iff $\alpha_i \sim_{k-m} \tilde{\alpha}_i$ for all $i \in [m]_0$. Thus, $w \sim_k \tilde{w}$ iff $\alpha_i \sim_{k-m} \tilde{\alpha}_i$ for all $i \in [m]_0$ and for $w' = \alpha_0\tilde{\beta}_1\alpha_1\cdots\tilde{\beta}_m\alpha_m$ we have $w \sim_k w'$.*

Theorem 3.3 *Let $w, \tilde{w} \in \Sigma^*$ with $m = \iota(w) = \iota(\tilde{w}) < k$. Then, $w \sim_k \tilde{w}$ iff $\alpha_{i-1}\beta_i\alpha_i \sim_{k-m+1} \tilde{\alpha}_{i-1}\tilde{\beta}_i\tilde{\alpha}_i$ for all $i \in [m]$.*

In the light of Theorem 3.3, in the following, we consider some special cases of these triples w.r.t. the alphabet of the both involved α. Hence, let $w, \tilde{w} \in \Sigma^*$ with $1 = \iota(w) = \iota(\tilde{w})$.

Proposition 3.4 *(1) Let $\alpha_0 = \alpha_1 = \tilde{\alpha}_0 = \tilde{\alpha}_1 = \varepsilon$. Then $w \sim_k \tilde{w}$ iff $k = 1$ or $k \geq 2$, $\mathrm{m}(w) = \mathrm{m}(\tilde{w})$, $\overleftarrow{\mathrm{m}}(w) = \overleftarrow{\mathrm{m}}(\tilde{w})$, and $\mathrm{core}_1 \sim_k \widetilde{\mathrm{core}}_1$.*
(2) Let $\mathrm{alph}(\alpha_i) = \mathrm{alph}(\tilde{\alpha}_i) \in \binom{\Sigma}{|\Sigma|-1}$, then $w \sim_k \tilde{w}$ iff $\alpha_i \sim_{k-1} \tilde{\alpha}_i$ for all $i \in [1]_0$.

The last proposition does not hold if not both $\mathrm{m}(w)$ and $\overleftarrow{\mathrm{m}}(w)$ are identical: consider $w = \mathtt{ababeabab} \cdot \mathtt{abecd} \cdot \mathtt{cdcdcd} \sim_4 \mathtt{ababeabab} \cdot \mathtt{baedc} \cdot \mathtt{cdcdcd} = \tilde{w}$ with $\mathrm{m}(w) = \mathtt{d} \neq \mathtt{c} = \mathrm{m}(\tilde{w})$ and $\overleftarrow{\mathrm{m}}(w) = \mathtt{a} \neq \mathtt{b} = \overleftarrow{\mathrm{m}}(\tilde{w})$. In the next proposition, we give a necessary condition for the α-factors.

Proposition 3.5 *Let $w \in \Sigma^*$ with $\iota(w) = 1$, $k \in \mathbb{N}$, and $\overleftarrow{\mathrm{M}} = \{\overleftarrow{\mathrm{m}}(\tilde{w})[1] \mid \tilde{w} \in [w]_{\sim_k}\}$. If $|\overleftarrow{\mathrm{M}}| \geq 2$ then there exists a factorisation $\alpha_0 = u_1 \cdots u_{k-1}$ with $\mathrm{alph}(u_1) \supseteq \ldots \supseteq \mathrm{alph}(u_{k-1}) \supseteq \overleftarrow{\mathrm{M}}$.*

4. The Binary and Ternary Case of Simon's Congruence

First, we apply our results to the binary alphabet. Note that for a given w with $\iota(w) \leq k$, we have $|\{\overleftarrow{\mathrm{m}}(\tilde{w}) \mid \tilde{w} \in [w]_{\sim_k}\}| = 1$.

Proposition 4.1 *For all $w \in \Sigma_2^*$, we have for all $i \in [\iota(w)]$, $\beta_i \in \{\mathtt{a}, \mathtt{b}, \mathtt{ab}, \mathtt{ba}\}$. If $\beta_i = \mathtt{x}$, then $\alpha_{i-1}, \alpha_i \in \overline{\mathtt{x}}^+$ with $\mathtt{x} \in \Sigma_2$ and if $\beta_i = \mathtt{x}\overline{\mathtt{x}}$, then $\alpha_{i-1} \in \mathtt{x}^*$ and $\alpha_i \in \overline{\mathtt{x}}^*$ with $\mathtt{x} \in \Sigma_2$.*

In the binary case the k-congruence of two words with identical $\iota < k$ leads to the same modi and same β giving a characterisation of $\sim_k$ for binary words in terms of unary words.

Lemma 4.2 *Let $w, w' \in \Sigma_2^*$ with $w \sim_k w'$ and $m = \iota(w) = \iota(w') < k$, then $\mathrm{m}(w) = \mathrm{m}(w')$ and thus, $\beta_i = \beta_i'$ for all $i \in [m]$.*

Theorem 4.3 *Let $w, w' \in \Sigma_2^*$ such that $m = \iota(w) = \iota(w') < k$, then $w \sim_k w'$ iff $\beta_i = \beta_i'$ for all $i \in [m]$ and $\alpha_i \sim_{k-m} \alpha_i'$ for all $i \in [m]_0$.*

Theorem 4.3 implies if $|[w]_{\sim_k}| = \infty$, then $\mathtt{x}^k \in \mathrm{ScatFact}_k(w)$ for some $\mathtt{x} \in \Sigma$ (the contrary is generally not true: $v = \mathtt{bbabb}$ w.r.t. $\sim_4$). The following theorem characterises the binary case.

Theorem 4.4 *Let $w \in \Sigma_2^*$, then $|[w]_{\sim_k}| < \infty$. We have $|[w]_{\sim_k}| = 1$ iff $\iota(w) < k$ and $|\alpha_i| < k - \iota(w)$ for all $i \in [\iota(w)]_0$.*

We present a formula for the precise value of $|\Sigma_2^*/\sim_k|$ counting classes based on the valid combinations of β-factors and number of classes for each α-factors and giving the index.

Theorem 4.5 *The number of congruence classes of $\Sigma_2^*/\sim_k$ of words with $m < k$ arches is given by $\| \binom{k-m\ \ k-m\ \ k-m}{1\ \ \ 2\ \ \ 1\ \ \ } \binom{k-m}{1}^m \cdot \binom{k-m}{k-m} \|_1 = c_k^m$ where $c_k^{-1} = 1$, $c_k^0 = 2k+1$, and $c_k^m = 2 \cdot (k-m+1) \cdot c_{k-1}^{m-1} - 2 \cdot (k-m) \cdot c_{k-2}^{m-2}$ where $\|\cdot\|_1$ denotes the 1-norm.*

Corollary 4.6 *For $k \in \mathbb{N}_0$ we have $|\Sigma_2^*/\sim_k| = 1 + \sum_{m=0}^{k-1} c_k^m$.*

Now, we consider Σ_3. Note that if $\mathrm{m}_1(w) = \overleftarrow{\mathrm{m}}_1(w)$ then $\mathrm{core}_1 = \varepsilon$. If $\mathrm{m}_1(w) \neq \overleftarrow{\mathrm{m}}_1(w)$, then $\mathrm{core}_1 \in (\Sigma \setminus \{\mathrm{m}_1(w), \overleftarrow{\mathrm{m}}_1(w)\})^*$, i.e., the cores are unary and denoted by $\mathrm{y} \in \Sigma_3$. Define for a boolean predicate P, $\delta_{P(x)} = 1$ if $P(x)$ is true and 0 otherwise. We assume $k \geq 2$ (we characterise 1-universal words) and $w, \tilde{w} \in \Sigma_3^*$ with $1 = \iota(w) = \iota(\tilde{w})$.

Lemma 4.7 *Let $\mathrm{m}(w) = \mathrm{m}(\tilde{w})$ and $\overleftarrow{\mathrm{m}}(w) = \overleftarrow{\mathrm{m}}(\tilde{w})$, we have $w \sim_k \tilde{w}$ iff $\alpha_i \sim_{k-1} \tilde{\alpha}_i$ for all $i \in [1]_0$ and $\mathrm{core}_1 \sim_{k-c} \widetilde{\mathrm{core}}_1 \in \mathrm{y}^*$ where $c \coloneqq \iota(\alpha_0) + \delta_{\mathrm{y} \preceq \mathrm{re}(\alpha_0)} + \iota(\alpha_1) + \delta_{\mathrm{y} \preceq \overleftarrow{\mathrm{re}}(\alpha_1)}$.*

We finish with a characterisation in the ternary case.

Theorem 4.8 *For $w, \tilde{w} \in \Sigma_3^*$ we have $w \sim_k \tilde{w}$ iff $\alpha_i \sim_{k-1} \tilde{\alpha}_i$ for all $i \in [1]_0$, and*
(1) $|\mathrm{alph}(\alpha_i)| = 2$, $\mathrm{alph}(\alpha_{1-i}) \cap \mathrm{alph}(\alpha_i) = \emptyset$, and $\iota(\alpha_i) \geq k-1$ for some $i \in [1]_0$, or
(2) $\mathrm{m}(w) = \mathrm{m}(\tilde{w})$, $\overleftarrow{\mathrm{m}}(w) = \overleftarrow{\mathrm{m}}(\tilde{w})$, and $\mathrm{core} \sim_{k-c} \widetilde{\mathrm{core}}$ where $c \coloneqq \iota(\alpha_0) + \delta_{\mathrm{y} \preceq \alpha_0} + \iota(\alpha_1) + \delta_{\mathrm{y} \in \alpha_1}$.

5. Conclusion

In this paper, we investigated the α-β-factorisation (cf. [6, 3]) as an object of intrinsic interest. This leads to a result characterising k-congruence of m-universal words in terms of their 1-universal $\alpha\beta\alpha$-factors. In the case of the binary and ternary alphabet, we fully characterised the congruence of words in terms of their single factors. Extending this idea of the α-β-factorisation to lower layers (arches w.r.t. some $\Omega \subset \Sigma$), is left as future work.

References

[1] L. Barker, P. Fleischmann, K. Harwardt, F. Manea, D. Nowotka, Scattered factor-universality of words. In: *DLT*. Springer, 2020, 14–28.

[2] P. Fleischmann, S. Germann, D. Nowotka, Scattered Factor Universality–The Power of the Remainder. *preprint arXiv:2104.09063 (published at RuFiDim)* (2021).

[3] P. Fleischmann, L. Haschke, A. Huch, A. Mayrock, D. Nowotka, Nearly k-universal words-investigating a part of simon's congruence. In: *DCFS*. 2022, 57–71.

[4] P. Karandikar, M. Kufleitner, P. Schnoebelen, On the index of Simon's congruence for piecewise testability. *Inf. Process. Lett.* **115** (2015) 4, 515–519.

[5] P. Karandikar, P. Schnoebelen, The height of piecewise-testable languages and the complexity of the logic of subwords. *LICS* **15** (2019) 2.

[6] M. Kosche, T. Koss, F. Manea, S. Siemer, Absent subsequences in words. In: *RP*. Springer, 2021, 115–131.

[7] M. Lothaire, *Combinatorics on Words*. Cambridge Mathematical Library, Cambridge University Press, 1997.

[8] I. Simon, Piecewise testable events. In: *Autom. Theor. Form. Lang., 2nd GI Conf.*. LNCS 33, Springer, 1975, 214–222.