Universality of Partially Ordered NFAs

Tomáš Masopust*

Institute of Mathematics, Czech Academy of Sciences, Brno and TU Dresden
masopust@math.cas.cz

Abstract

An automaton is partially ordered if the only cycles in its transition diagram are self-loops. The expressivity of partially ordered NFAs (poNFAs) can be characterized by the Straubing-Thérien hierarchy [18, 19]. The most studied level of the hierarchy is level 1, known as *piecewise testable languages*. Piecewise testable languages are recognized by *confluent, partially ordered DFAs* [16, 8]. Omitting confluence results in *partially ordered DFAs* (poDFAs) studied by Brzozowski and Fich [3], who showed that poDFAs recognize *$\mathcal{R}$-trivial languages*, a class of languages strictly between levels 1 and $\frac{3}{2}$. Lifting the notion from DFAs to NFAs, Schwentick, Thérien and Vollmer [15] showed that poNFAs recognize level $\frac{3}{2}$ and are thus more powerful than poDFAs. Languages of level $\frac{3}{2}$ are also known as *Alphabetical Pattern Constraints* [2], which are regular languages effectively closed under permutation rewriting. We showed that the increase of expressivity of poNFAs compared to poDFAs is caused by self-loop transitions involved in nondeterminism. Consequently, $\mathcal{R}$-trivial languages are characterized by *self-loop deterministic poNFAs (rpoNFAs)*. Our study further reveals that *complete, confluent and self-loop deterministic poNFAs (ptNFAs)* characterize piecewise testable languages; ptNFAs are thus a natural extension of confluent poDFAs to nondeterministic automata.

We studied the *universality* problem for these types of poNFAs. The problem asks whether a given automaton accepts all words over its alphabet. The problem is PSPACE-complete for NFAs [14].

Despite a rather low expressivity, the universality problem for poNFAs has the same worst-case complexity as for general NFAs, even if restricted to binary alphabets. This is because poNFAs possess a powerful nondeterminism; self-loops involved in nondeterminism admit an unbounded number of nondeterministic steps – the poNFA either stays in the same state or moves to another one. Forbidding such self-loops results in self-loop deterministic poNFAs where the number of nondeterministic steps is bounded by the number of states. This restriction affects the complexity – deciding universality of self-loop deterministic poNFAs is coNP-complete if the alphabet is fixed. However, the complexity remains PSPACE-complete if the alphabet may grow polynomially, since the growth of the alphabet compensates for the restriction on the number of nondeterministic steps. Surprisingly, the reduced complexity is also preserved by the weaker complete, confluent and self-loop deterministic poNFAs.

Héam [5] characterized level $\frac{1}{2}$ as languages recognized by saturated poNFAs (spoNFA), also called *shuffle ideals* or *upward closures*. A poNFA is *saturated* if it has a self-loop under every letter in every state. Deciding universality for spoNFAs is simple – find a state that is both initial and accepting. Therefore, complete, confluent and self-loop deterministic poNFAs are the simplest and natural kind of NFAs recognizing a well-known class of languages for which the universality problem is as difficult as for general NFAs. Our results are summarized in Table 1.

A consequence of our results is the worst-case complexity of inclusion and equivalence problems – the complexity of universality is a lower bound. Deciding inclusion or equivalence of two languages given as (complete, confluent and) self-loop deterministic poNFAs is thus PSPACE-complete in general, coNP-complete if the alphabet is fixed, and NL-complete if the alphabet is unary. Considering the inclusion $L(A) \subseteq L(B)$ for automata over a fixed alphabet, the problem is coNP-complete if B is a self-loop deterministic poNFA and A is of any other type. Our results further

*Joint work with Markus Krötzsch (TU Dresden) and Michaël Thomazo (INRIA)

	ST	$\lvert\Sigma\rvert = 1$	$\lvert\Sigma\rvert = k \geq 2$	Σ is growing
DFA		L-comp. [7]	NL-comp. [7]	NL-comp. [7]
spoNFA	$\frac{1}{2}$	AC^0 [9]	AC^0 [9]	AC^0 [9]
ptNFA	1	NL-comp. [9]	coNP-comp. [12]	PSPACE-comp. [9]
rpoNFA		NL-comp. [10]	coNP-comp. [10]	PSPACE-comp. [10]
poNFA	$\frac{3}{2}$	NL-comp. [10]	PSPACE-comp. [10]	PSPACE-comp. [1]
NFA		coNP-comp. [17]	PSPACE-comp. [1]	PSPACE-comp. [1]

Table 1: Complexity of deciding universality for poNFAs; ST stands for the corresponding level of the Straubing-Thérien hierarchy; Σ denotes the input alphabet

show that the k-piecewise testability problem for complete, confluent and self-loop deterministic poNFAs is PSPACE-complete. This problem is of interest in XML databases and separability [4, 6, 11].

The presentation is based on the following papers:

[9] M. Krötzsch and T. Masopust. "Universality of Confluent, Self-Loop Deterministic Partially Ordered NFAs is Hard". In: Available online at `https://arxiv.org/abs/1704.07860`. 2017.

[10] M. Krötzsch, T. Masopust, and M. Thomazo. "On the Complexity of Universality for Partially Ordered NFAs". In: *MFCS*. Vol. 58. LIPIcs. 2016, 61:1–61:14.

[12] T. Masopust. "Piecewise Testable Languages and Nondeterministic Automata". In: *MFCS*. Vol. 58. LIPIcs. 2016, 67:1–67:14.

[13] T. Masopust and M. Thomazo. "On boolean combinations forming piecewise testable languages". In: *Theoretical Computer Science* 682 (2017), pp. 165–179.

Acknowledgment Supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 (HAEC) and in Emmy Noether grant KR 4381/1-1 (DIAMOND).

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] A. Bouajjani, A. Muscholl, and T. Touili. "Permutation rewriting and algorithmic verification". In: *Information and Computation* 205 (2007), pp. 199–224.

[3] J. A. Brzozowski and F. E. Fich. "Languages of R-Trivial Monoids". In: *Journal of Computer and System Sciences* 20 (1980), pp. 32–49.

[4] W. Czerwiński, W. Martens, and T. Masopust. "Efficient Separability of Regular Languages by Subsequences and Suffixes". In: *ICALP*. Vol. 7966. LNCS. 2013, pp. 150–161.

[5] P.-C. Héam. "On Shuffle Ideals". In: *Theoretical Informatics and Applications* 36 (2002), pp. 359–384.

[6] P. Hofman and W. Martens. "Separability by Short Subsequences and Subwords". In: *ICDT*. Vol. 31. LIPIcs. 2015, pp. 230–246.

[7] N. D. Jones. "Space-Bounded Reducibility among Combinatorial Problems". In: *Journal of Computer and System Sciences* 11 (1975), pp. 68–85.

[8] O. Klíma and L. Polák. "Alternative Automata Characterization of Piecewise Testable Languages". In: *DLT*. Vol. 7907. LNCS. 2013, pp. 289–300.

[9] M. Krötzsch and T. Masopust. "Universality of Confluent, Self-Loop Deterministic Partially Ordered NFAs is Hard". In: Available online at `https://arxiv.org/abs/1704.07860`. 2017.

[10] M. Krötzsch, T. Masopust, and M. Thomazo. “On the Complexity of Universality for Partially Ordered NFAs”. In: *MFCS*. Vol. 58. LIPIcs. 2016, 61:1–61:14.

[11] W. Martens et al. “BonXai: Combining the simplicity of DTD with the expressiveness of XML Schema”. In: *PODS*. 2015, pp. 145–156.

[12] T. Masopust. “Piecewise Testable Languages and Nondeterministic Automata”. In: *MFCS*. Vol. 58. LIPIcs. 2016, 67:1–67:14.

[13] T. Masopust and M. Thomazo. “On boolean combinations forming piecewise testable languages”. In: *Theoretical Computer Science* 682 (2017), pp. 165–179.

[14] A. R. Meyer and L. J. Stockmeyer. “The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space”. In: *SWAT*. 1972, pp. 125–129.

[15] T. Schwentick, D. Thérien, and H. Vollmer. “Partially-Ordered Two-Way Automata: A New Characterization of DA”. In: *DLT*. Vol. 2295. LNCS. 2001, pp. 239–250.

[16] I. Simon. “Hierarchies of Events with Dot-Depth One”. PhD thesis. University of Waterloo, Canada, 1972.

[17] L. J. Stockmeyer and A. R. Meyer. “Word Problems Requiring Exponential Time: Preliminary Report”. In: *STOC*. 1973, pp. 1–9.

[18] H. Straubing. “A Generalization of the Schützenberger Product of Finite Monoids”. In: *Theoretical Computer Science* 13 (1981), pp. 137–150.

[19] D. Thérien. “Classification of Finite Monoids: The Language Approach”. In: *Theoretical Computer Science* 14 (1981), pp. 195–208.