BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2023
http://excel.fit.vutbr.cz

Regulated Language Operations and Their Use

David Chocholatý*

Abstract
The classical finite automaton has been extended in many different ways. One of the modifications is the *general jumping finite automata* which do not read the input string left to right in a standard way. Instead, jumps can be provided over the input tape and substrings are erased. This paper introduces and studies *erasing systems* as an alternative to general jumping finite automata. A significant difference compared to the given automata is a control regular language instead of state control. Along with the introduction of the new formal system, the paper demonstrates its properties and studies multiple applications.

*xchoch09@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The *jumping finite automata* were introduced in 2012 by Meduna and Zemek [1, 2], and in the past years, they were well studied, including their applications, for example, see [3, 4, 5, 6, 7]. However, regarding how *regulated finite automatons* use languages (see [2]), we can consider the new formal system, which uses regular language for controlling the erasions.

The main motivation for the creation of the mentioned system is the study of a formal model with similar jumping properties to the *general jumping finite automaton*, which uses state control in the form of general finite automata. The new formal system called *erasing system* leaves the string work on the input tape, while regular languages themselves can be accepted by classical finite automata.

At the same time, the paper demonstrates the relations with well-known language families, the family of shuffle languages, Dyck languages and closure properties.

Based on the formal specification of the erasing system, multiple applications in molecular biology, text editors and compositional chess are shown, including designing algorithms and presenting the implementation solution.

2. General Jumping Finite Automata

This section introduces the most general type of jumping finite automata — the general jumping finite automaton. The definitions are taken from [2].

2.1 Definitions and State Control

The definitions of the general jumping finite automaton are provided in the first section of the poster, see the following: $\boxed{\text{Definition 1.1.}}$, $\boxed{\text{Definition 1.2.}}$, and $\boxed{\text{Definition 1.3.}}$. The example of state control in the form of the general finite automaton is depicted in $\boxed{\text{Figure 1.1}}$.

In the rest of the paper, the general jumping automaton is denoted by GJFA and the family of languages accepted by general jumping finite automata **GJFA**.

3. Erasing Systems

This section introduces a completely new formal system called erasing system. The properties and relations of the family of languages accepted by erasing systems are shown.

3.1 Definitions and Examples

The definitions of the new formal system called erasing system are provided on the poster (see following: $\boxed{\text{Definition 2.1.}}$, $\boxed{\text{Definition 2.2.}}$, $\boxed{\text{Definition 2.3.}}$). The input string is accepted by the erasing system if and only if such a sequence of *erasing steps* exists that the final string, which arises by the concatenation of the used erasing string, belongs to the *regular language*.

The $\boxed{\text{Example 2.1.}}$ presents that the new formal system can accept the well-known context-sensitive non context-free language.

Throughout the rest of the paper, the erasing system is denoted by ES and the family of languages accepted by erasing systems by **ES**.

3.2 Relations with Specific Language Families

The second section provides the following theorems: $\boxed{\text{Theorem 3.1.}}$, $\boxed{\text{Theorem 3.2.}}$, and $\boxed{\text{Theorem 3.3.}}$ which shows that ES can accept every Dyck or semi-Dyck language (see more [8]). The last theorem shows that the family of the languages, which are generated by the shuffle expressions (see [5]), and **ES** are incomparable.

3.3 Relations with Well-Known Language Families

The next big part of the work is proof of the relations between **ES** and the language families from Chomsky's hierarchy (see more [9], [10]). Every relation is shown on the poster in $\boxed{\text{Figure 4.1}}$. The surprising result is, that the ESs can't accept every finite language.

3.4 Closure Properties

The other differences between GJFA and ES were proved for the closure properties of the families accepted by the specific formal model. The differences captures the table, $\boxed{\text{Table 5.1}}$. As can be seen, the **ES** is not closed under *union* and *reversal* as **GJFA**.

4. Applications of the Erasing Systems

Several applications of the ESs were provided concretely in the three following areas.

4.1 Bioinformatics for Molecular Biology

The ESs in the field of bioinformatics found use especially for the processing of the sequence (see the first three points on the poster in the same section).

4.2 Text Editors

The main purpose of the use of the ESs is their properties with Dyck languages, which can represent the proper bracketing. Based on that, the ESs can verify if the brackets in the text are properly written.

4.3 The N-Queens Problem

The last application of the ESs is the check of the solution for the n-queens problem. If the solution is encoded to the string in the desired shape, the proposed algorithm can decide whether the input solution is the correct solution for the n-queens problem or not. For example, the solutions in the figures, $\boxed{\text{Figure 6.1}}$ and $\boxed{\text{Figure 6.2}}$, are not correct.

5. Implementation of the Erasing Systems Applications

This section presents the main algorithm of the ES operations and explains the use of the two types of quantifiers — *greedy* and *lazy* (see [11]).

5.1 Erasing System Algorithm

The main algorithm of the ES operations is provided by algorithm, $\boxed{\text{Algorithm 7.1}}$. A very interesting part is the transformation of ES behaviour. In the algorithm designing, the process of the erasings transforms to the searching the state space (see [12]).

5.2 Greedy and Lazy Quantifier

The two types of quantifiers are used. Briefly, the operators $*$ and $+$ try to include as many symbols as possible in the subexpression assigned to them. Lazy quantifiers do the same thing in the opposite way. The behaviour of the listed types of quantifiers is shown on the poster in $\boxed{\text{Figure 7.1}}$.

6. Conclusions

This work aimed to introduce a new formal system called the erasing system (denoted by ES). The paper together with the poster defines the ES itself. At the same time, it answers questions such as what is its accepting power or demonstrates closure properties. In the mentioned areas, surprising results were achieved.

Subsequently, the applications of the new formal system to real problems were introduced in the field of bioinformatics for molecular biology, in text editors, and to verify the correctness of the solution to a known n queens problem.

Acknowledgements

I would like to sincerely thank my supervisor prof. RNDr. Alexander Meduna, CSc. for his help and expert guidance. I also would like to thank Ing. Zbyněk Křivka, PhD for his advice in the area of current research on the theory of formal languages and Ing. Ivana Burgetová, PhD for her recommendations in the application of the formal system in bioinformatics.

References

[1] Alexander Meduna and Petr Zemek. Jumping finite automata. *International Journal of Foundations of Computer Science*, 23(7):1555–1578, 2012.

[2] Alexander Meduna and Petr Zemek. *Regulated Grammars and Automata*. Computer science. Springer, New York, 1 edition, 2014.

[3] Vojtěch Vorel. On basic properties of jumping finite automata. *CoRR*, abs/1511.08396, 11 2015.

[4] Vojtěch Vorel. Two results on discontinuous input processing. volume 9777, pages 205–216, 07 2016.

[5] Henning Fernau, Meenakshi Paramasivan, Markus L. Schmid, and Vojtěch Vorel. Characterization and complexity results on jumping finite automata. *Theoretical Computer Science*, 679:31–52, 2017. Implementation and Application of Automata.

[6] Radim Kocman, Benedek Nagy, Zbyněk Křivka, and Alexander Meduna. A jumping 5'->3' watson-crick finite automata model. In *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)*, books@ocg.at 332, pages 117–132. Austrian Computer Society, 2018.

[7] Stephen Obare, Abejide Ade-Ibijola, and George Okeyo. Jumping finite automata for tweet comprehension. pages 1–7, 11 2019.

[8] Phillip G. Bradford. Efficient exact paths for dyck and semi-dyck labeled path reachability. *CoRR*, abs/1802.05239, 2 2018.

[9] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[10] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[11] Jan Goyvaerts and Steven Levithan. *Regular Expressions Cookbook*. O'Reilly Media, Inc., Sebastopol, 2 edition, 8 2012.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence*. Prentice Hall, USA, 3 edition, 2010.